

# Relational Solver Notes

Jules Jacobs

November 6, 2025

## 1 Roadmap

This document sketches the relational solver that accompanies the Mox type system notes. The goal is to develop a mode solver that is powerful enough to handle the mode constraints produced by the type checker.

At a high level, we will develop a solver for binary constraints between modes. We will first analyze a class of constraints that can be solved in polynomial time. Then we will develop a more powerful solver that can handle that class of constraints.

## 2 Constraint Language

As a first step, assume we have a finite domain  $V$  of values, and a set of binary relations  $R_i \subseteq V \times V$ . Given a set of variables and asserted constraints between them, we want to determine if there exists a valuation of the variables that satisfies all the constraints.

In general this is a NP-complete problem: consider  $V = \{0, 1, 2, \dots, k\}$  and  $R_1 = \{(a, b) \mid a \neq b\} \subseteq V \times V$ . Given a graph we can use this constraints to encode the  $k$ -coloring problem: we want to assign a color to each vertex such that no two adjacent vertices have the same color. The  $k$ -coloring problem is NP-complete, so this problem is also NP-complete.

However, certain classes of constraints can be solved in polynomial time. Consider the set of constraints  $R_i = \{(a, b) \mid b \geq a + i\} \subseteq V \times V$ . Given a graph of variables and constraints, we can solve this problem in polynomial time using the Floyd-Warshall algorithm.

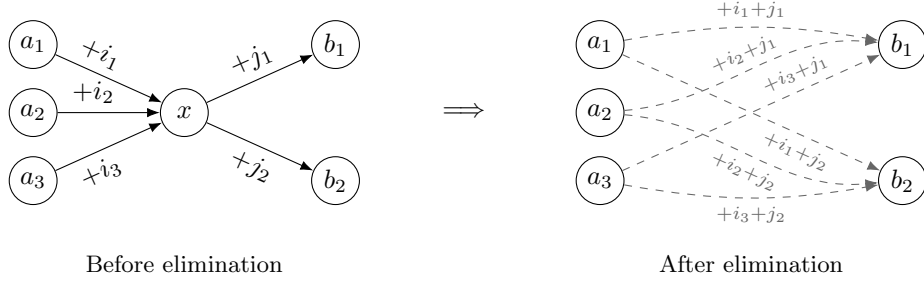
Equivalently, we can solve the problem by variable elimination: we take a variable  $x$  and all adjacent constraints on it. We assert all transitive constraints (where  $R_i$  composes with  $R_j$  to produce  $R_{i+j}$ ) and repeat until all variables are eliminated. If, during this process, we ever see a constraint between a variable and itself with  $i \neq 0$ , then the constraints are unsatisfiable.

Why does this elimination strategy work for this class of constraints, but not for the general case, and in particular for the  $k$ -coloring problem with inequality constraints?

Consider a variable  $x$  with:

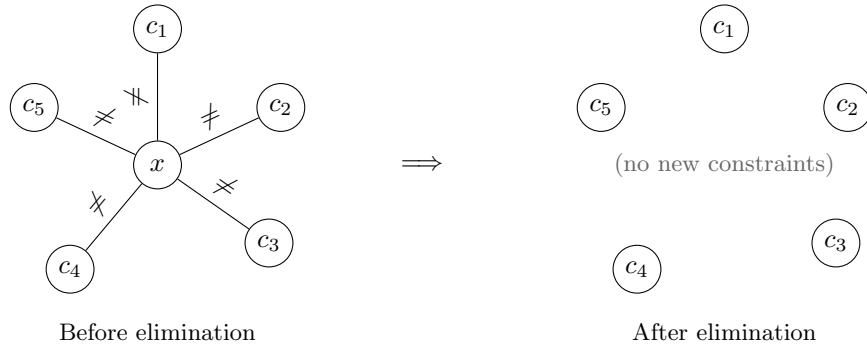
- a set of predecessors  $a_1, a_2, a_3$  with  $R_{i_1}(a_1, x), R_{i_2}(a_2, x), R_{i_3}(a_3, x)$  constraints on them,
- a set of successors  $b_1, b_2$  with  $R_{j_1}(x, b_1), R_{j_2}(x, b_2)$  constraints on them,

When eliminating  $x$ , we assert every transitive constraint  $R_{i_p+j_q}(a_p, b_q)$  for  $p \in \{1, 2, 3\}$  and  $q \in \{1, 2\}$ :



I claim that if there is a solution for the neighboring variables that satisfies all of those transitive constraints, then there is a solution for  $x$  that satisfies the original constraints: we can simply set  $x$  to any value in the interval  $\max(a_1 + i_1, a_2 + i_2, a_3 + i_3) \leq x \leq \min(b_1 + j_1, b_2 + j_2)$ . This interval is guaranteed to be non-empty if the transitive constraints hold.

The key blocker for  $k$ -coloring is that this property does not hold for inequality constraints. Suppose for example we have a vertex  $x$  and  $k+1$  neighbors with inequality constraints. The transitive constraints are trivial if  $k \geq 3$ , because if we have  $a \neq x \neq b$ , then for a given value of  $a$ , all values of  $b$  are still possible, by choosing a particular value for  $x$ . Thus, the strategy of variable elimination does not work for  $k$ -coloring, for  $k \geq 3$ : for  $\neq$  constraints, eliminating  $x$  produces no useful transitives; every neighbour pair remains unconstrained:



The OCaml mode solver has constraints of the form  $x \leq G(y)$  where  $G$  are modalities with left adjoints. Like the interval constraints we considered above, these constraints can be solved in polynomial time using variable elimination, and for the same reason.

### **2.1 Which constraints can be solved by elimination?**

For a set of constraints to be solvable by elimination, we need that the original problem has a solution iff the transformed problem has a solution. We will now analyze precisely when this property holds, and provide an easily checkable criterion for it.

## **3 Solver Architecture**

## **4 Open Questions**