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, fix a domain V of values (not necessarily finite) together with a collection of binary relations $R_i \subseteq V \times V$. Given a set of variables and asserted contraints between them, we want to determine if there exists a valuation of the variables that satisfies all the constraints. Throughout this section the relational reasoning does not depend on finiteness; we will explicitly mark the places later where finite carriers become important for algorithmic purposes.

In general this is an 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.

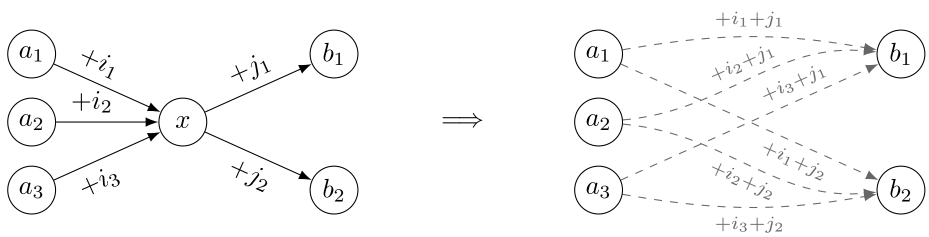
Equivalently, we can solve the problem by variable elimination: we take a variable x and all adjacent constraints on it. We assert all transitive constraints (where R_i composes with R_j to produce R_{i+j}) and repeat until all variables are eliminated. If, during this process, we ever see a constraint 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 $q \in \{1, 2\}$:



Before elimination After elimination

Figure 1: Difference constraints generate informative transitives.

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 $\max(a_1 + i_1, a_2 + i_2, a_3 + i_3) \le x \le \min(b_1 + j_1, b_2 + j_2)$. 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 $k \ge 3$, because if we have $a \ne x \ne b$, 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 $k \ge 3$: for $\ne$ constraints, eliminating x produces no useful transitives; every neighbour pair remains unconstrained:

The OxCaml mode solver has constraints of the form $x \le G(y)$ 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

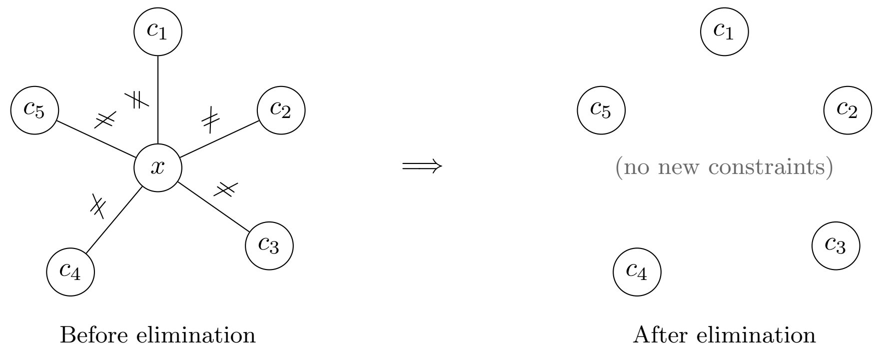


Figure 2: Inequality constraints add no new relations between neighbours.

now analyze precisely when this property holds, and provide an easily checkable criterion for it.

Eliminating x yields edges $R_{ij} = R_i^\top R_0^* R_j$ directed from y_j to y_i for all $i \geq j$, including self-loops R_{ii}. If there were existing relations between the y_i then we intersect them with these. Figure 3 illustrates the local transformation on the "star" centered at x.

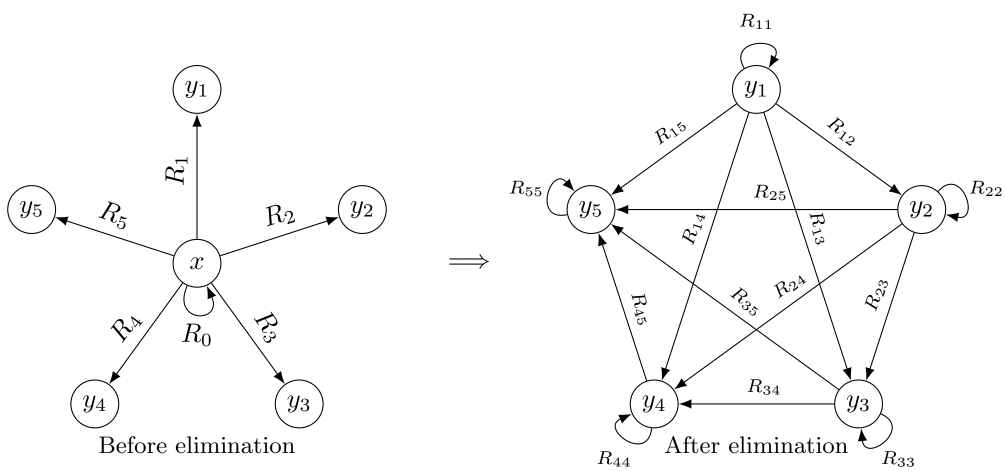


Figure 3: Eliminating x in the radial layout yields edges R_{ji} (with $R_{ji} \stackrel{\text{def}}{=} R_j^\top R_0^* R_i$) directed from y_j to y_i for all $i \geq j$, as well as self loops R_{ii}.

The key question is: for which families of relations Rel does this algorithm work?

Answer. Intuitively, the elimination step is sound if every time we eliminate a variable x, the constraints we generate between its neighbors are "complete": any solution of the projected constraints on the neighbors can be extended to a value of x. The obstruction is exactly the phenomenon we saw with $\neq$-constraints: locally consistent projections need not extend.

We now characterize when elimination is complete.

2.2 Closure and slices

Fix a domain V (possibly infinite) and a family of binary relations $\mathsf{Rel} \subseteq \mathcal{P}(V \times V)$. The elimination algorithm only ever uses relations obtained from Rel by:

- composition: $RS = \{(a, c) \mid \exists b.\ (a, b) \in R \wedge (b, c) \in S\}$,
- intersection: $R \cap S$,
- converse: $R^\top = \{(b, a) \mid (a, b) \in R\}$,
- restriction to the diagonal (from self-loops): $R^* = \{(a, a) \mid (a, a) \in R\}$.

Let $\overline{\mathsf{Rel}}$ be the smallest set of relations containing Rel and closed under these four operations. When V or Rel are infinite, $\overline{\mathsf{Rel}}$ may itself be infinite; this is fine for the structural results below, and we only exploit finiteness later when we want to enumerate these relations effectively.

For a relation $R \in \overline{\mathsf{Rel}}$ and an element $b \in V$ we write:

$$R[\_, b] \;=\; \{a \in V \mid (a, b) \in R\} \qquad \text{and} \qquad R[a, \_] \;=\; \{b \in V \mid (a, b) \in R\}$$

for its *column* and *row* slices. These are exactly the unary constraints on x induced by fixing the other endpoint.

Let $\mathcal{F}$ be the family of all such slices:

$$\mathcal{F} \;=\; \{\, R[\_, b], R[a, \_] \mid R \in \overline{\mathsf{Rel}},\ a, b \in V \,\} \;\subseteq\; \mathcal{P}(V).$$

Because $\overline{\mathsf{Rel}}$ is closed under intersection, so is $\mathcal{F}$: for any $A, B \in \mathcal{F}$ there exists $C \in \mathcal{F}$ with $C = A \cap B$. This intersection-closure is crucial below.

2.3 A Helly-style condition

The key combinatorial property is a Helly-type condition on $\mathcal{F}$.

Definition 1 (Helly-2 for slices)**.** *We say that* Rel *has the* Helly-2 slice property *if the following holds:*

For every finite subfamily $\{F_1, \ldots, F_k\} \subseteq \mathcal{F}$, if all pairwise intersections are non-empty,

$$F_i \cap F_j \neq \varnothing \quad \textit{for all } i \neq j,$$

then the total intersection is non-empty,

$$\bigcap_{i=1}^{k} F_i \neq \varnothing.$$

Because $\mathcal{F}$ is intersection-closed, this is equivalent to the absence of a *bad triple*:

Lemma 2 (Triples suffice for intersection-closed families)**.** *Assume $\mathcal{F}$ is closed under intersection. Then $\mathcal{F}$ has the Helly-2 slice property if and only if there do not exist $A, B, C \in \mathcal{F}$ such that*

$$A \cap B \neq \varnothing, \quad B \cap C \neq \varnothing, \quad C \cap A \neq \varnothing, \quad A \cap B \cap C = \varnothing.$$

Proof. ($\Rightarrow$) Immediate: such a triple would violate Helly-2.

($\Leftarrow$) Suppose Helly-2 fails. Pick a counterexample $\{F_1, \ldots, F_m\} \subseteq \mathcal{F}$ of minimal size: all pairs intersect, but $\bigcap_i F_i = \varnothing$. Since $\mathcal{F}$ is intersection-closed, for each $i \geq 2$ the set $A_i := F_1 \cap F_i$ lies in $\mathcal{F}$, is non-empty, and $\bigcap_{i=2}^m A_i = \bigcap_{i=1}^m F_i = \varnothing$. If some $A_i \cap A_j$ were empty, then F_1, F_i, F_j would form a bad triple. Otherwise $\{A_2, \ldots, A_m\}$ is a smaller counterexample, contradicting minimality. □

Thus in our setting we can detect failure of the Helly-2 slice property by a finite search for such bad triples among slices in $\overline{\mathsf{Rel}}$.

Notice that the Helly-2 slice property and the elimination argument below only rely on these relational intersections existing; the sets involved may be infinite. Consequently, the structural characterization applies equally well to infinite carriers. We will only appeal to finiteness when we want to enumerate $\overline{\mathsf{Rel}}$ or to mechanize the Helly check.

2.4 Correctness of elimination

We now state the main correctness criterion.

Theorem 3 (Characterization of admissible constraint families)**.** *Let V be a domain (possibly infinite) and $\mathsf{Rel} \subseteq \mathcal{P}(V \times V)$. Consider the following variable-elimination algorithm: at each step, pick a variable x, and for every two neighbors $y_i \xleftarrow{R_i} x \xrightarrow{R_j} y_j$ insert the composed edge $y_i \xleftarrow{R_i R_0^* R_j} y_j$ (where R_0^* is the current self-loop on x, possibly Id), intersecting with any existing edge between y_i and y_j, then delete x. If at any point a self-loop on some variable becomes empty on the diagonal (i.e. forbids all (a, a)), report unsatisfiable.*

Then the following are equivalent:

1. *For every finite constraint graph over* Rel*, this elimination algorithm decides satisfiability correctly: the final instance is satisfiable iff the original instance is satisfiable.*

2. Rel *has the Helly-2 slice property.*

Proof sketch. (2) $\Rightarrow$ (1) (*completeness of elimination*). It is enough to show that a *single* elimination step is complete, and then argue by induction on the number of variables.

Consider eliminating x from its star. Fix an assignment to the neighbors $\{y_i\}$ that satisfies all constraints generated by the algorithm between the y_i.

For each incident edge, this assignment induces a slice $S_i \in \mathcal{F}$ of admissible values for x (e.g., for $y_i \xrightarrow{R_i} x$ with y_i fixed to b_i we get $S_i = R_i[\_, b_i]$). By construction, every binary constraint added between y_i and y_j enforces that S_i and S_j intersect: if they did not, the corresponding composed relation would have removed the chosen pair (b_i, b_j). Hence the family $\{S_i\}$ is pairwise-intersecting. By the Helly-2 slice property, $\bigcap_i S_i \neq \varnothing$. Picking any a in this intersection and setting $x := a$ extends the neighbor assignment to a full solution of the original constraints around x. Inductively, every model of the final instance lifts to a model of the original one.

Soundness of the algorithm (it never introduces spurious solutions) holds because every new edge relation is logically implied by the existence of an intermediate x satisfying its incident constraints.

(1) $\Rightarrow$ (2) (*necessity*). Assume the Helly-2 slice property fails. By Lemma 2 and intersection-closure, there exist $S_1, S_2, S_3 \in \mathcal{F}$ with all pairwise intersections non-empty but empty triple intersection. By definition of $\mathcal{F}$, we can realize each S_i as the admissible values for some variable x given a fixed neighbor y_i and a relation $R_i \in \overline{\mathsf{Rel}}$. We construct a constraint star with center x and leaves y_1, y_2, y_3 so that, for the chosen values of the y_i, the corresponding slices are precisely S_i. By pairwise intersection, every pair of leaves admits a value of x; hence the elimination algorithm, which only asserts pairwise compositions, accepts this partial assignment to y_1, y_2, y_3. But by emptiness of $S_1 \cap S_2 \cap S_3$, no value of x satisfies all three constraints simultaneously, so the original instance is unsatisfiable. Thus elimination is not complete, contradicting (1). □

Corollary 4 (Finite enumerability check)**.** *When V and Rel are finite, the Helly-2 slice property for Rel is decidable:*

1. *compute $\overline{\mathsf{Rel}}$ (finite);*
2. *form all slices $R[\_, b], R[a, \_]$;*
3. *check that no triple of slices has non-empty pairwise intersections and empty triple intersection.*

If no such triple exists, variable elimination as above is sound and complete for all constraint graphs over Rel. When V or Rel are infinite the theorem still applies, but this finite search procedure may no longer be available.

In particular, difference constraints and the OxCaml-style modal constraints $x \leq G(y)$ with adjoints fall into this framework: their induced slices are (discrete) intervals, which satisfy Helly-2, so elimination is complete. By contrast, $\neq$-constraints generate non-convex slices, violate Helly-2, and are correctly rejected by this criterion.

2.5 Many-sorted constraint families

We now generalize the previous discussion from a single carrier set V to a many-sorted setting. This is the natural abstraction for type/mode systems where variables, modes, or resources live in different universes and constraints relate values across sorts.

Sorted carriers and typed relations. Let $\mathcal{O}$ be a set of sorts. For each $A \in \mathcal{O}$, fix a carrier V_A (again, not necessarily finite). A binary relation is *typed* by its source and target sorts, so a relation $R \in \mathsf{Rel}(A, B)$ is a subset of $V_A \times V_B$.

Given two typed relations $R \in \mathsf{Rel}(A, B)$ and $S \in \mathsf{Rel}(B, C)$ with matching middle sort B, their composition is the typed relation $(RS)_{A,C} = \{(a, c) \mid \exists b \in V_B.\ (a, b) \in R \wedge (b, c) \in S\}$. Intersection is defined only for relations with the same type: $R \cap S \in \mathsf{Rel}(A, B)$. Converse flips the type: $(R)^\top = R^\top \in \mathsf{Rel}(B, A)$. For a same-sort relation $R \in \mathsf{Rel}(A, A)$, diagonal restriction is $R^* = \{(a, a) \mid (a, a) \in R\} \in \mathsf{Rel}(A, A)$.

Let $\overline{\mathsf{Rel}}$ be the least family of typed relations containing Rel and closed under these typed operations (composition, intersection, converse, and diagonal restriction where defined).

Slices by sort. For a typed relation $R \in \overline{\mathsf{Rel}}(A, B)$ and element $b \in V_B$, define the slice

$$R[\_, b] \;=\; \{a \in V_A \mid (a, b) \in R\}$$

For each sort A, collect all slices into the sorted slice family

$$\mathcal{F}_A \;=\; \{R[\_, b] \mid R \in \overline{\mathsf{Rel}}(A, B),\ b \in V_B\}.$$

As in the single-sorted case, each $\mathcal{F}_A$ is closed under intersection because $\overline{\mathsf{Rel}}$ is closed under intersection of like-typed relations.

Definition 5 (Helly-2 by sort)**.** *We say that a typed family* Rel *has the* Helly-2 slice property *if for every sort $A \in \mathcal{O}$ the slice family $\mathcal{F}_A$ satisfies Helly-2: for every finite subfamily $\{F_1, \ldots, F_k\} \subseteq \mathcal{F}_A$, pairwise non-empty intersections imply a non-empty total intersection.*

By intersection-closure, Lemma 2 applies sortwise: for each A it is enough to rule out *bad triples* $A, B, C \in \mathcal{F}_A$ with all pairwise intersections non-empty but empty triple intersection.

Typed elimination. Constraint graphs are now many-sorted: each variable $x : A$ has sort $A \in \mathcal{O}$ and each edge $x \xrightarrow{R} y$ carries a typed relation $R \in \overline{\mathsf{Rel}}(A, B)$. Self-loops on x carry same-sort relations $R \in \overline{\mathsf{Rel}}(A, A)$.

When eliminating a variable $x : A$, we first normalize the local star so that all incident non-loop edges are oriented *outward* from x: we replace any incoming edge $y \xrightarrow{S} x$ by the outgoing edge $x \xrightarrow{S^\top} y$. Let the outgoing edges be $x : A \xrightarrow{R_i} y_i : B_i$ and the self-loop be $R_0 \in \overline{\mathsf{Rel}}(A, A)$. For each pair $i \leq j$ we insert the composed edge from y_j to y_i with typed relation

$$R_{ij} \;=\; (R_i)^\top\, R_0^*\, R_j \quad \in \overline{\mathsf{Rel}}(B_i, B_j),$$

intersecting with any existing relation between y_i and y_j. If some same-sort self-loop becomes empty on the diagonal, report unsatisfiable.

Theorem 6 (Many-sorted characterization)**.** *For carriers $\{V_A\}_{A\in\mathcal{O}}$ (possibly infinite) and a typed family* Rel*, the typed variable-elimination algorithm above is sound and complete for all many-sorted constraint graphs over* Rel *if and only if* Rel *has the Helly-2 slice property by sort.*

Proof sketch. The argument is verbatim the single-sorted proof, carried out for each eliminated sort. $\square$

A computational enumeration of $\overline{\mathsf{Rel}}$ now requires each relevant sort to have a finite carrier and only finitely many primitive relations of that type; otherwise the structural theorem still applies but the finite search of Corollary 4 is unavailable.

3 Solver Architecture

3.1 Further optimizations

- Keep domain constraints (diagonally restricted relations) up to date.
- Remove constraints that are redundant given the domain constraints.
- Remove diagonal constraints by union-find on the variables that have to be equal.
- 2-SAT

3.2 Incremental solving

3.3 Generalization with levels

4 Open Questions