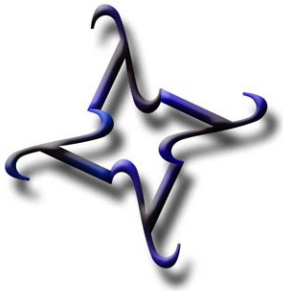
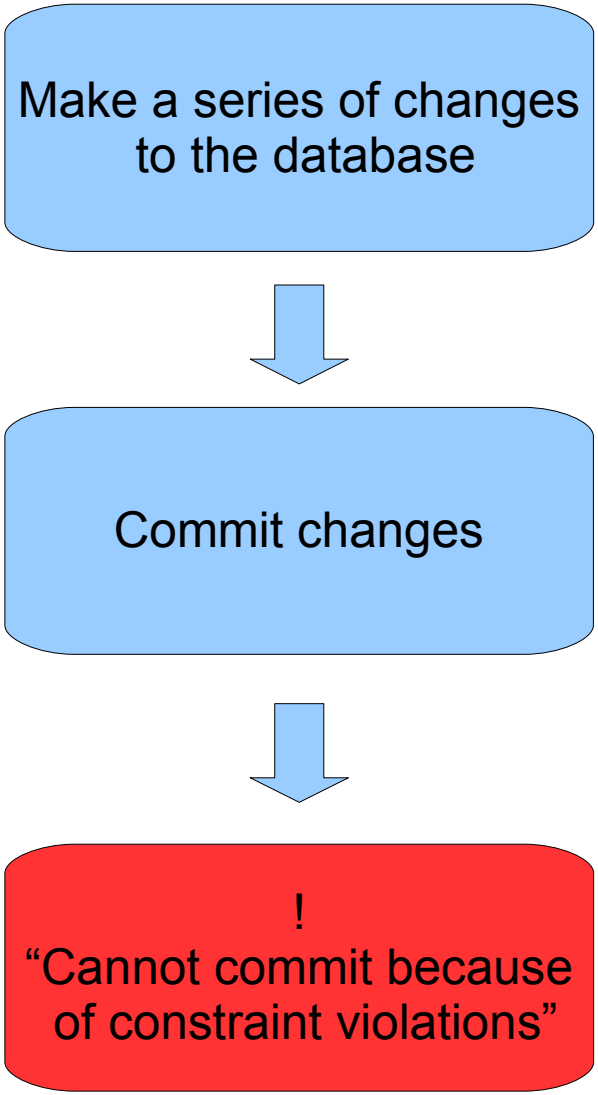
Repair guidance in a WYSIWYG data editor



Student: Eric Kao
Advisor: Michael Genesereth
Logic Group

Traditional data editing

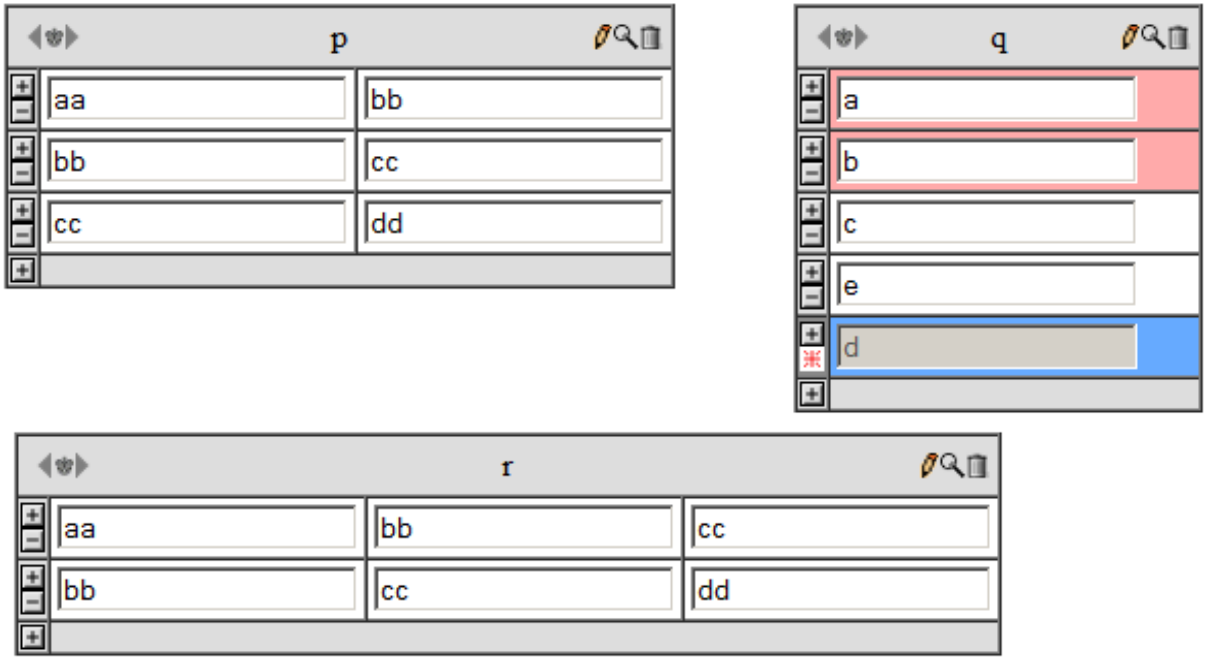


- Hard to understand how to avoid constraint violation

Restricted Interface

- Ask a series of questions in a fixed order
- Disallow choices that lead to violations
- Inflexible
- Hard to “navigate” between states

DataWYZ



- Unrestricted editing
- Visually rendered violations
- Visual repair guidance

Formal Representation

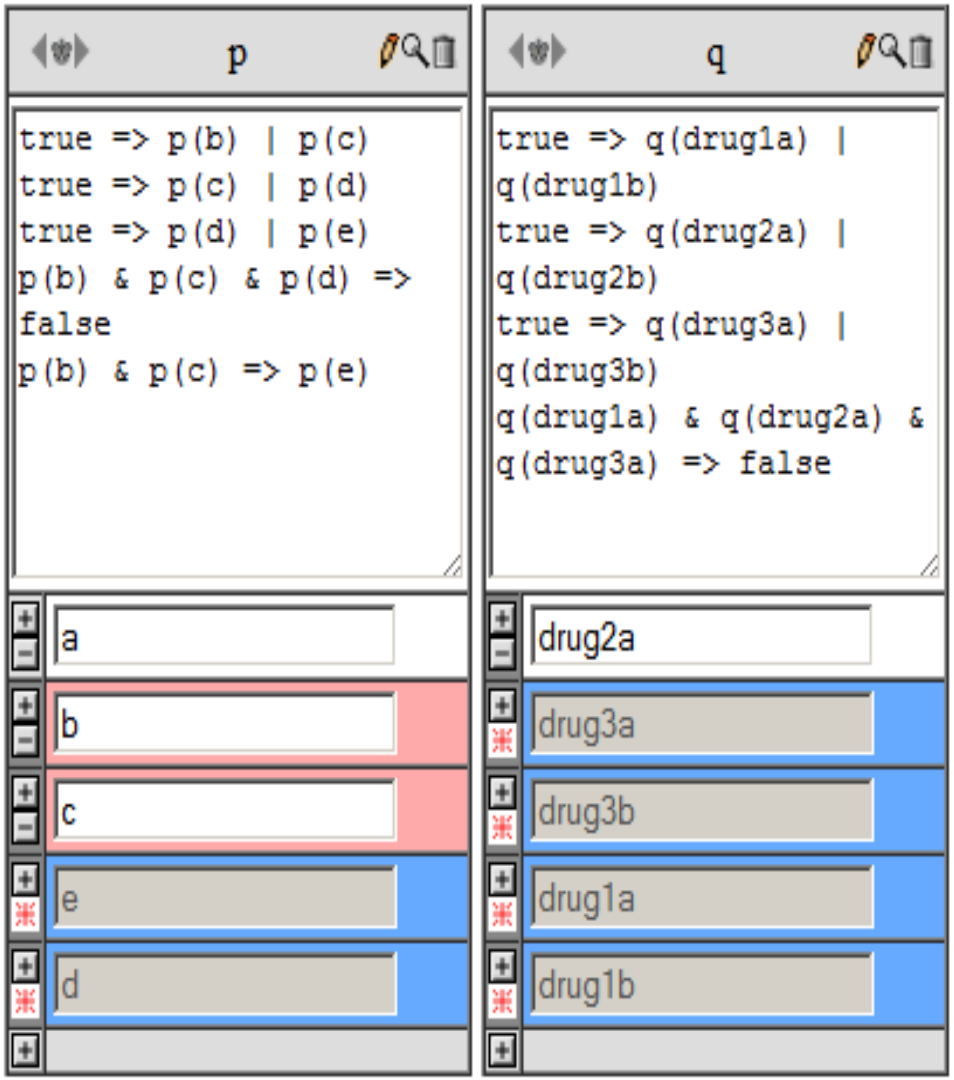
- D : Data
  - e.g., { use(drug1), use(drug2), use(drug3), ~use(drug4) }
- C : Constraints
  - e.g., {
    use(drug1) => use(drug2),
    use(drug2) => ~use(drug3),
    use(drug1) => ~use(drug3)
    }

Violation Pinpointing

- Pinpointing: highlight every minimal subset D' of D such that D' violates C.
  - e.g.,
    mv(D,C) = { { use(drug1), use(drug3) } }
  - Note: { use(drug1), use(drug2), use(drug3) } is not minimal
- Naïve method has exponential running time
- Precompile constraints

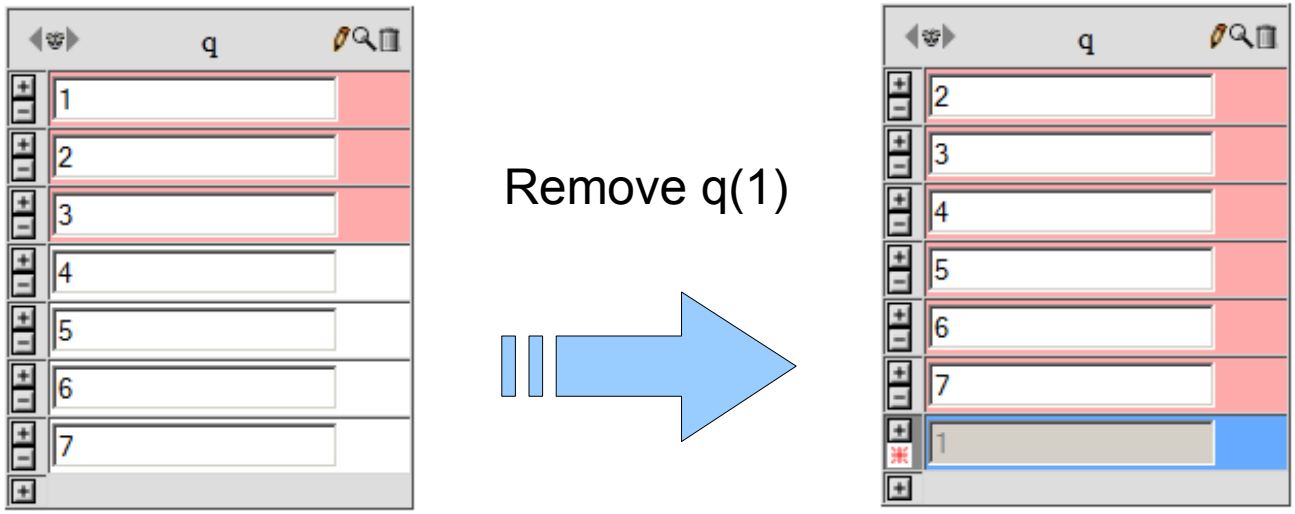
Basic repair guidance

- Suggest which tuples to add or remove
- Suggest adding {neg(S) : S ∈ mv(D,C) }
- Suggest removing {pos(S) : S ∈ mv(D,C) }

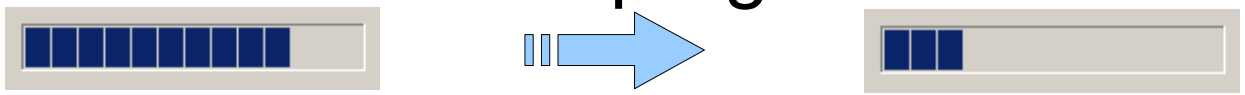


Problem: non-monotone progress

- Choosing certain changes lead to “bigger” violations
  - q(1) & q(2) => false
  - q(2) & q(3) => false
  - q(3) & q(4) & q(5) & q(6) & q(7) => q(1)

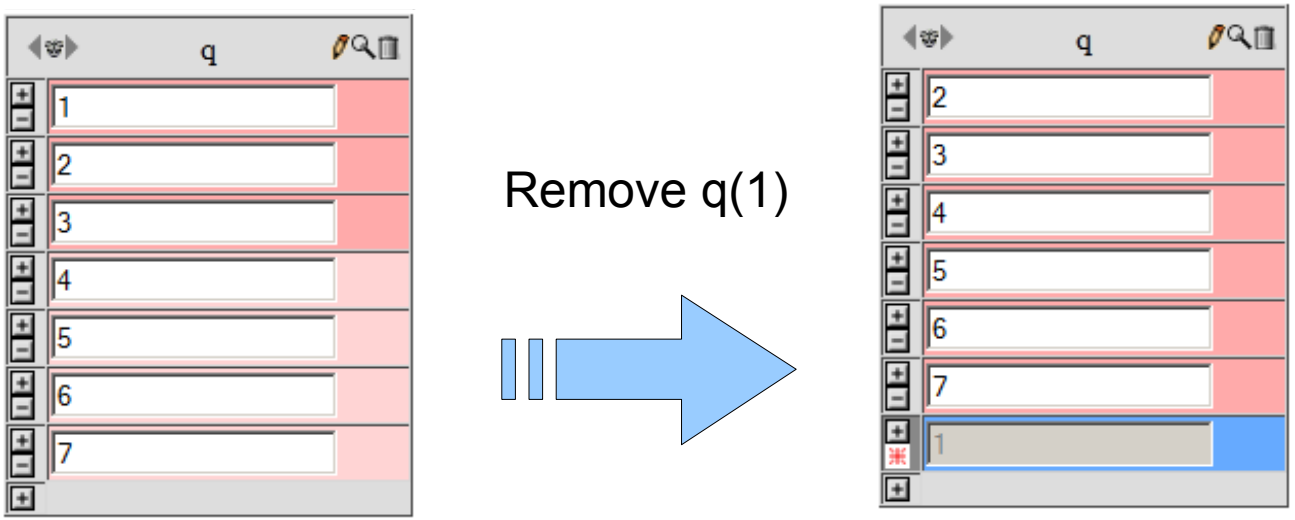


- No intuitive sense of progress.



Potentially affected

- Highlight all potentially affected tuples (PA)
- The set of highlighted tuple monotonically decrease

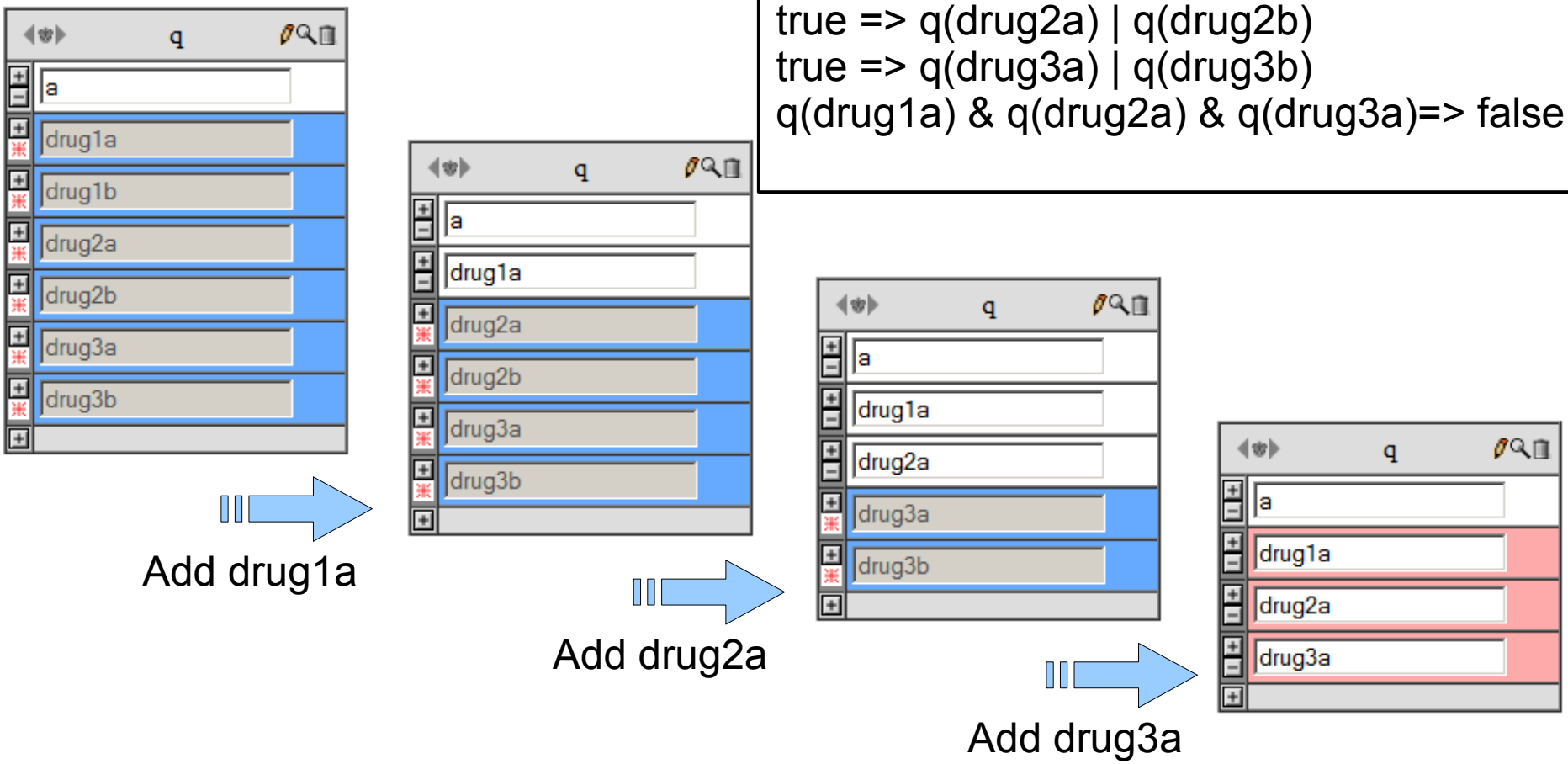


- Naïve procedure: simulate all repair choices
- Precompile constraints to compute PA quickly

Back-tracking

- In following repair guidance, the user may make choices that

true => q(drug1a) | q(drug1b)
true => q(drug2a) | q(drug2b)
true => q(drug3a) | q(drug3b)
q(drug1a) & q(drug2a) & q(drug3a)=> false



Back-track-free guidance

- When back-track free guidance is enabled, DataWYZ hides choices that would lead to back-tracking

```
true => q(drug1a) | q(drug1b)
true => q(drug2a) | q(drug2b)
true => q(drug3a) | q(drug3b)
q(drug1a) & q(drug2a) & q(drug3a)=> false
```