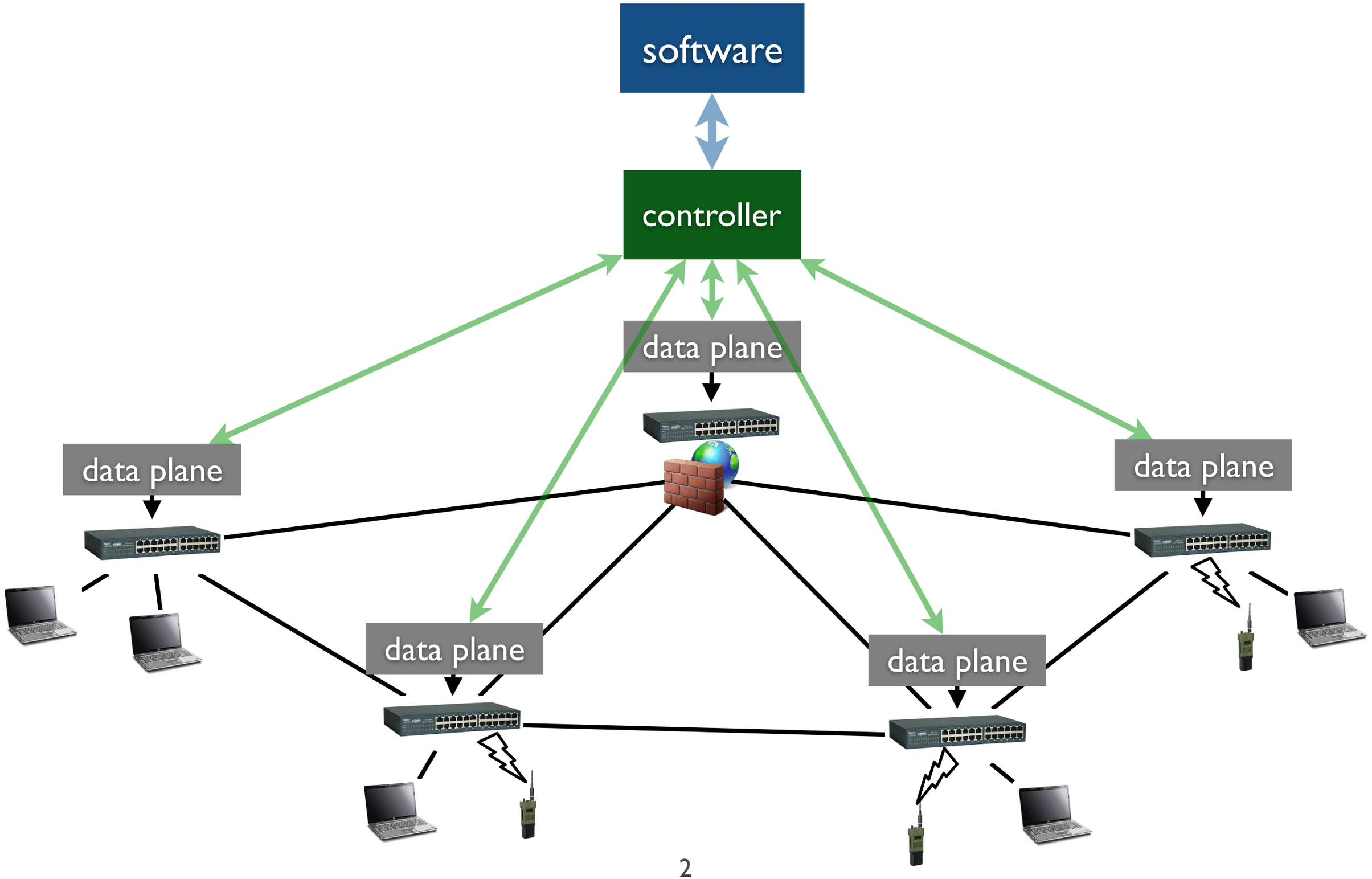
(Ir)relevance reasoning for software-defined network

Anduo Wang[*] Jason Croft[†] Xueyuan Mei[†]
Matthew Caesar[†] Brighten Godfrey[†]

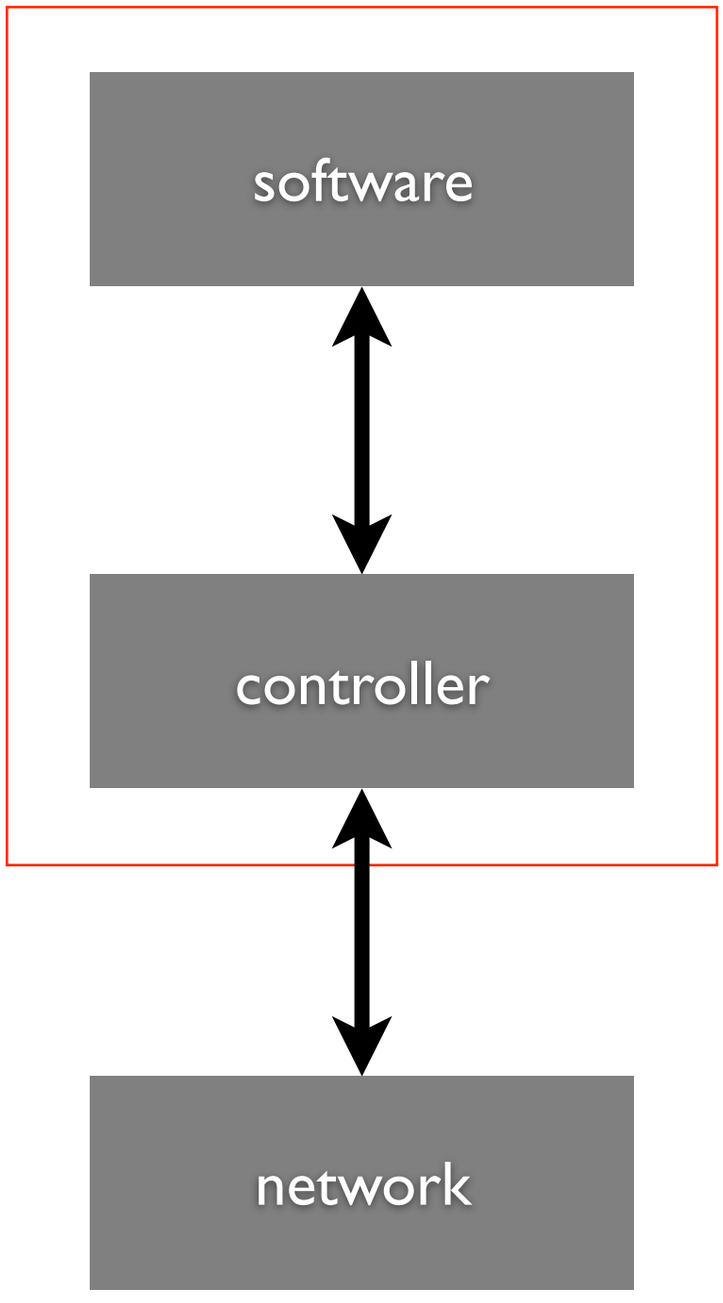
[*]*Temple University* [†]*University of Illinois Urbana-Champaign*

ExCAPE PI Meeting, May 9, 2016

software-defined networking (SDN)

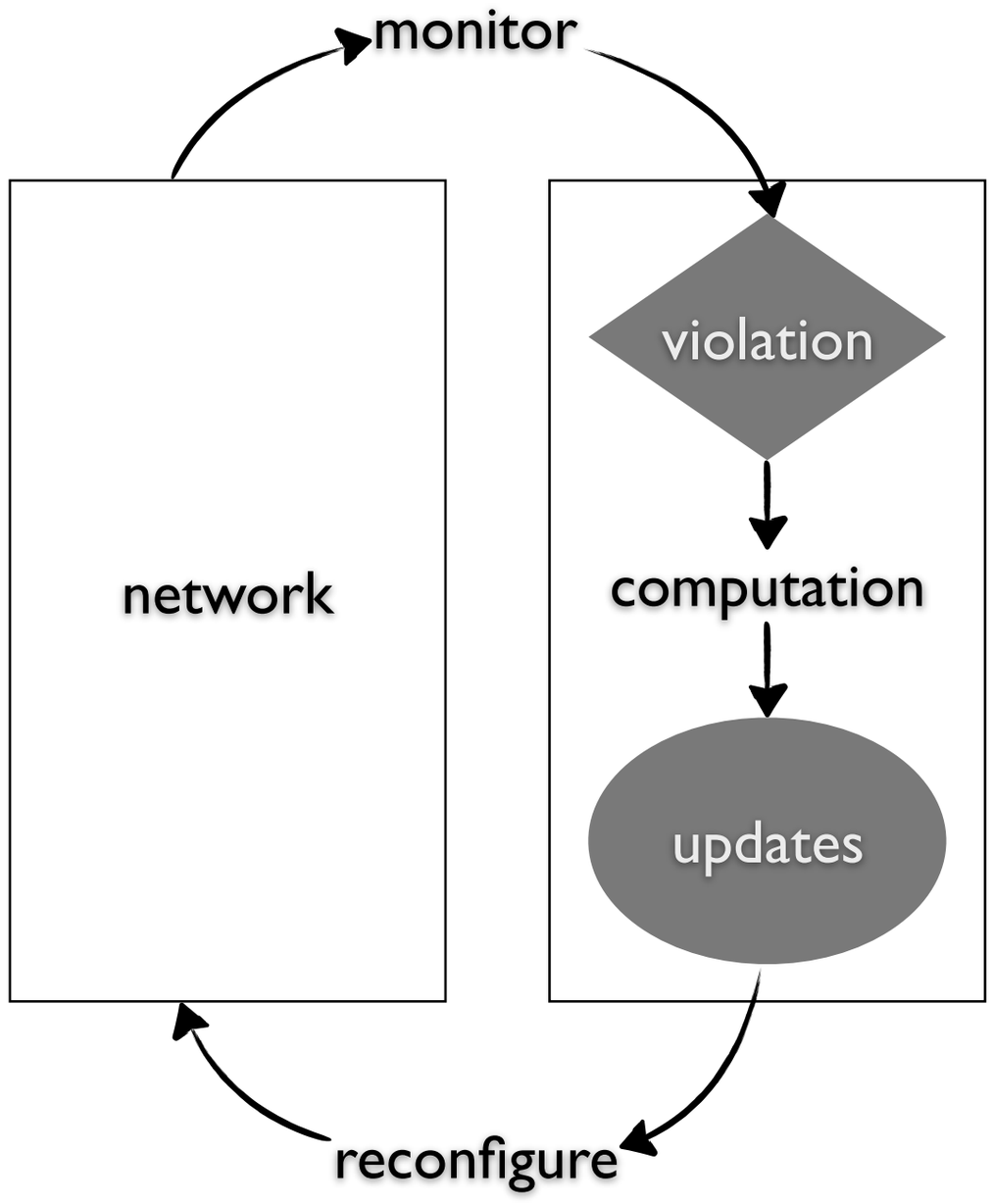


software-defined networking (SDN)



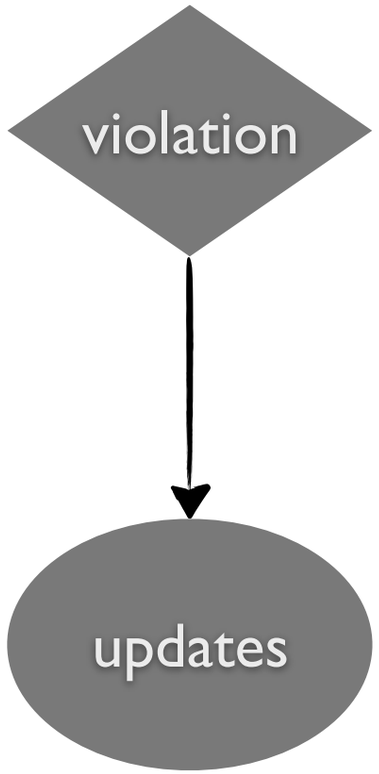
SDN moves complexity to
control software:
an opportunity and challenge

SDN control software

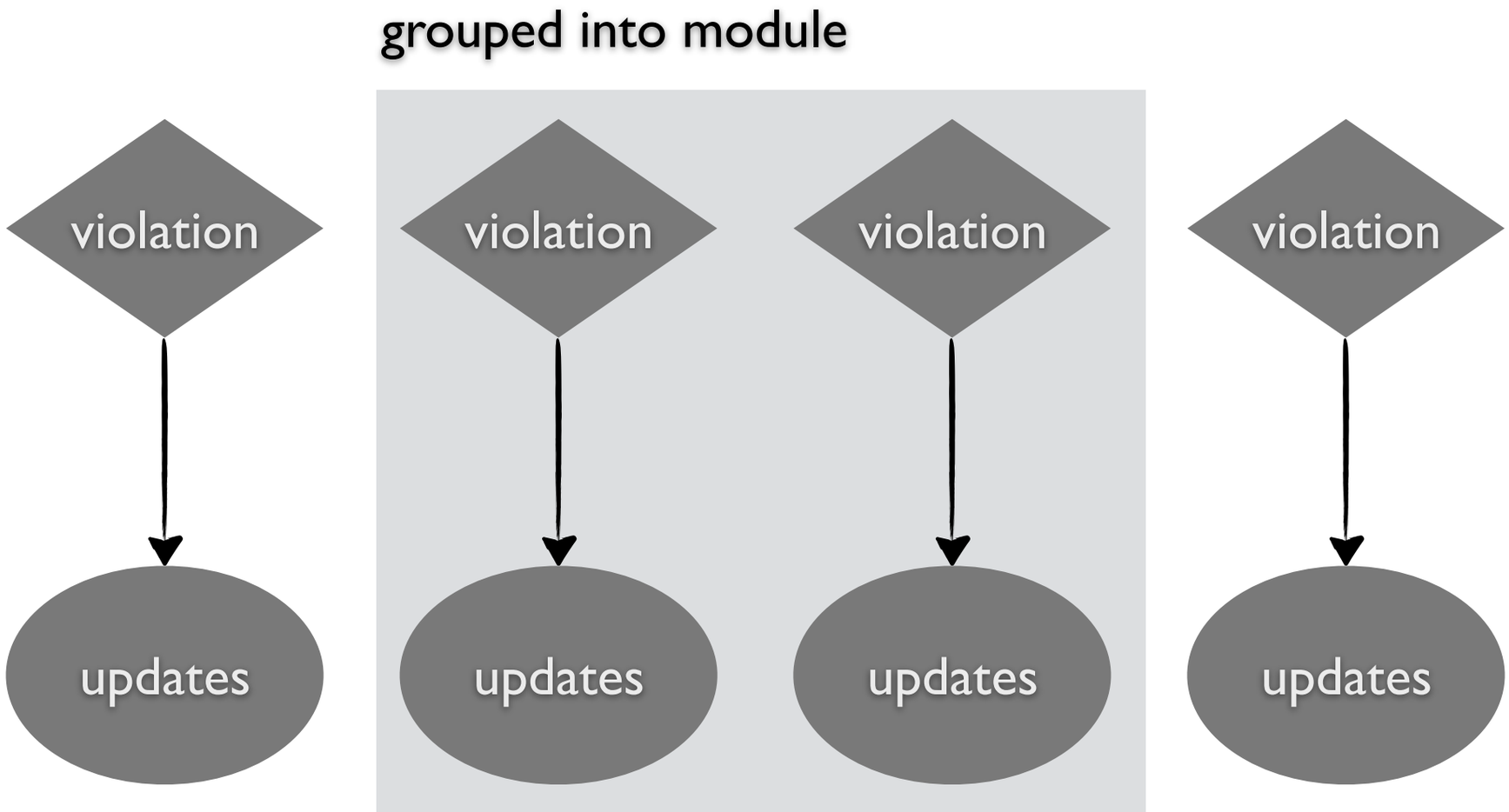


SDN control software

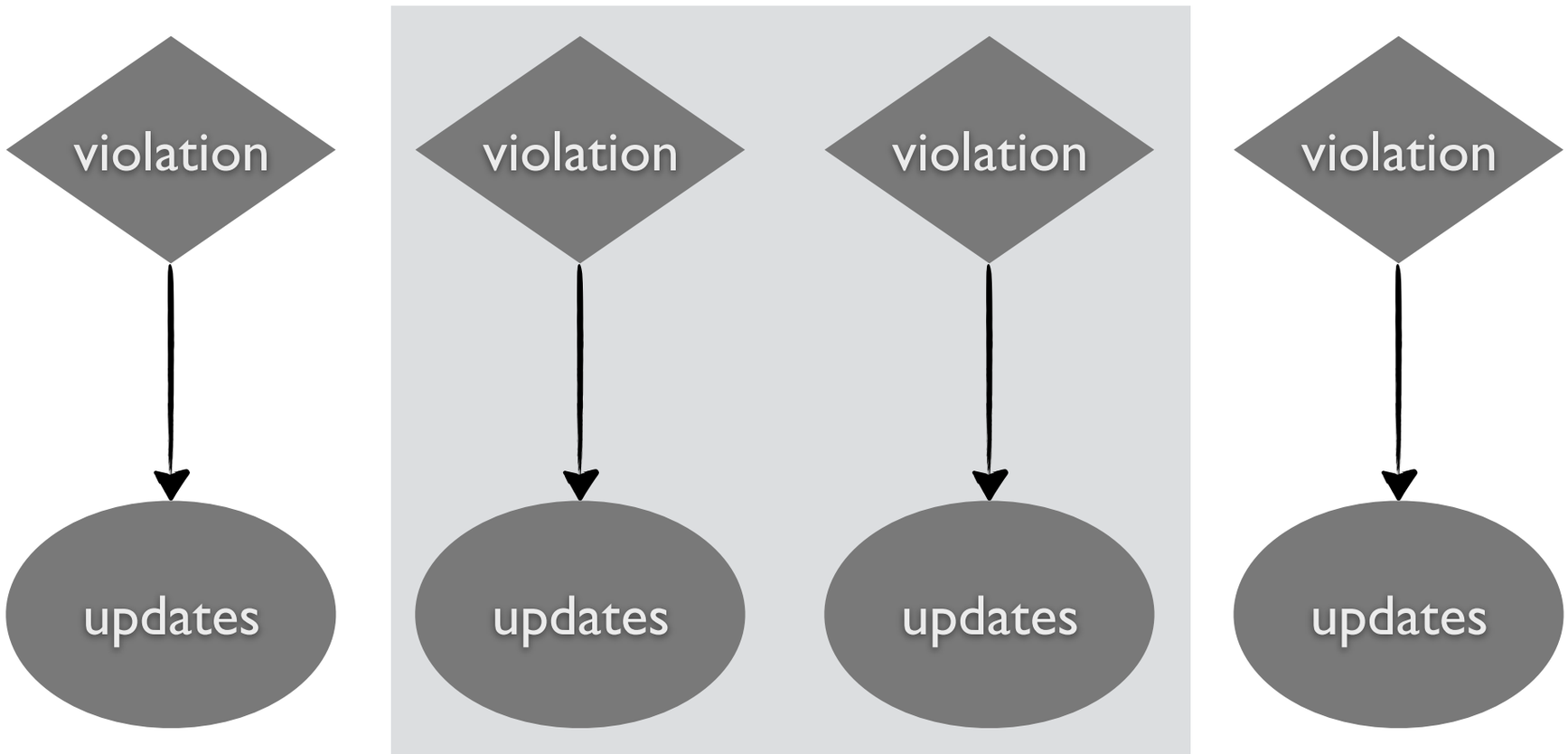
an individual control operation



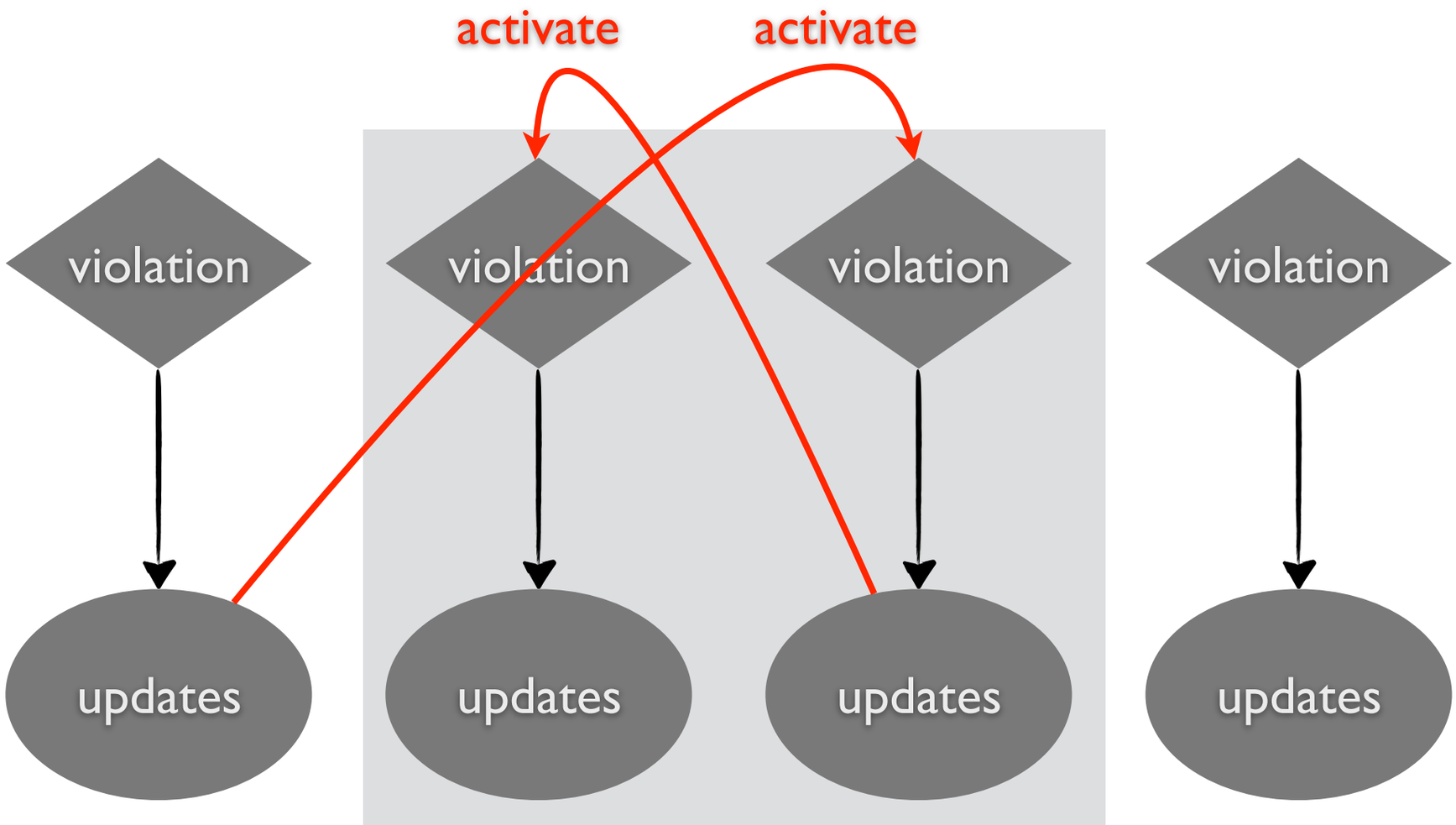
SDN control software



SDN control software



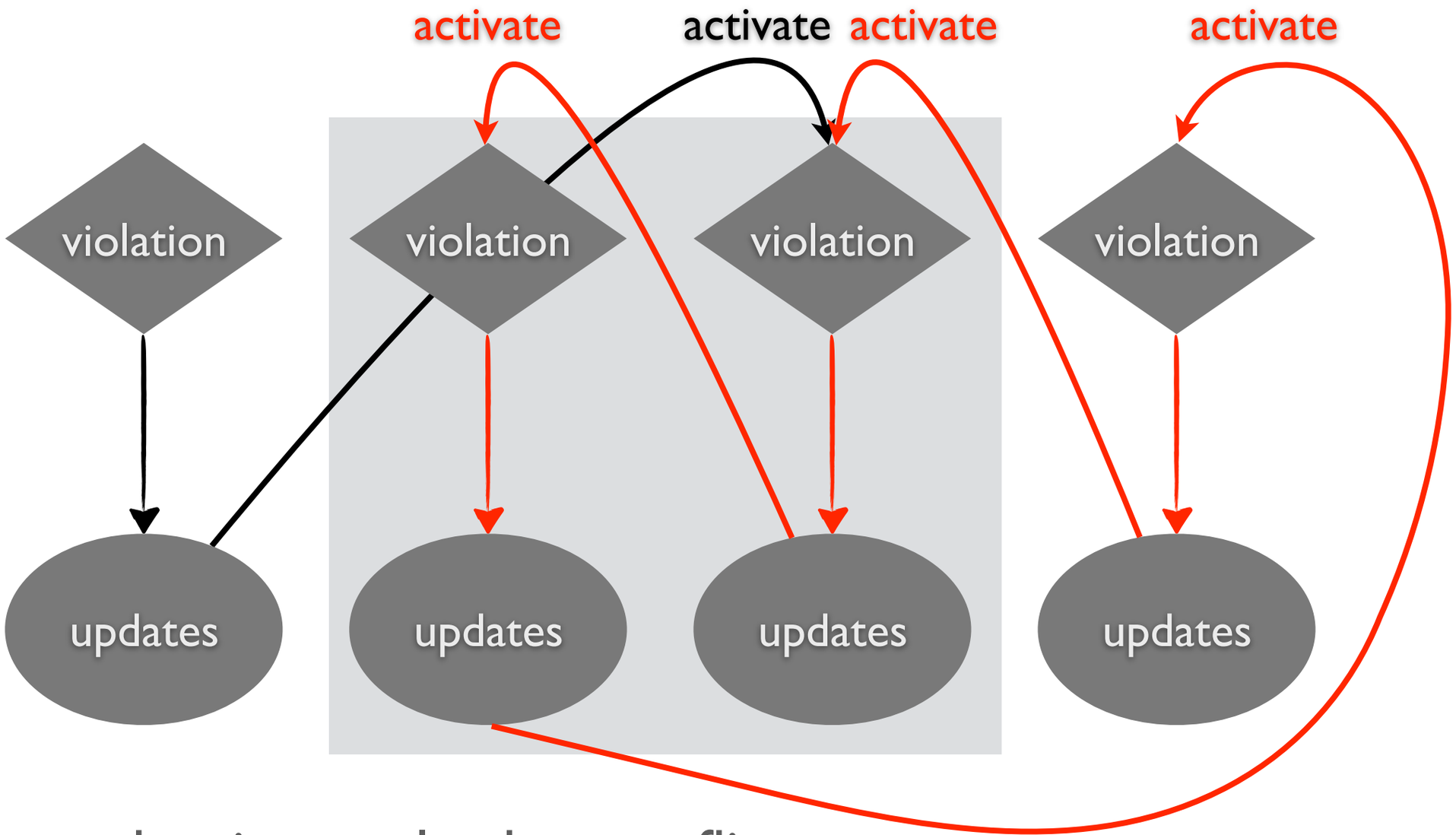
managing complexity in control software



dependency within and across modules

- programming abstraction: modular programming, instrumentation by master programs
- limitation: manual, requires understanding of module internals

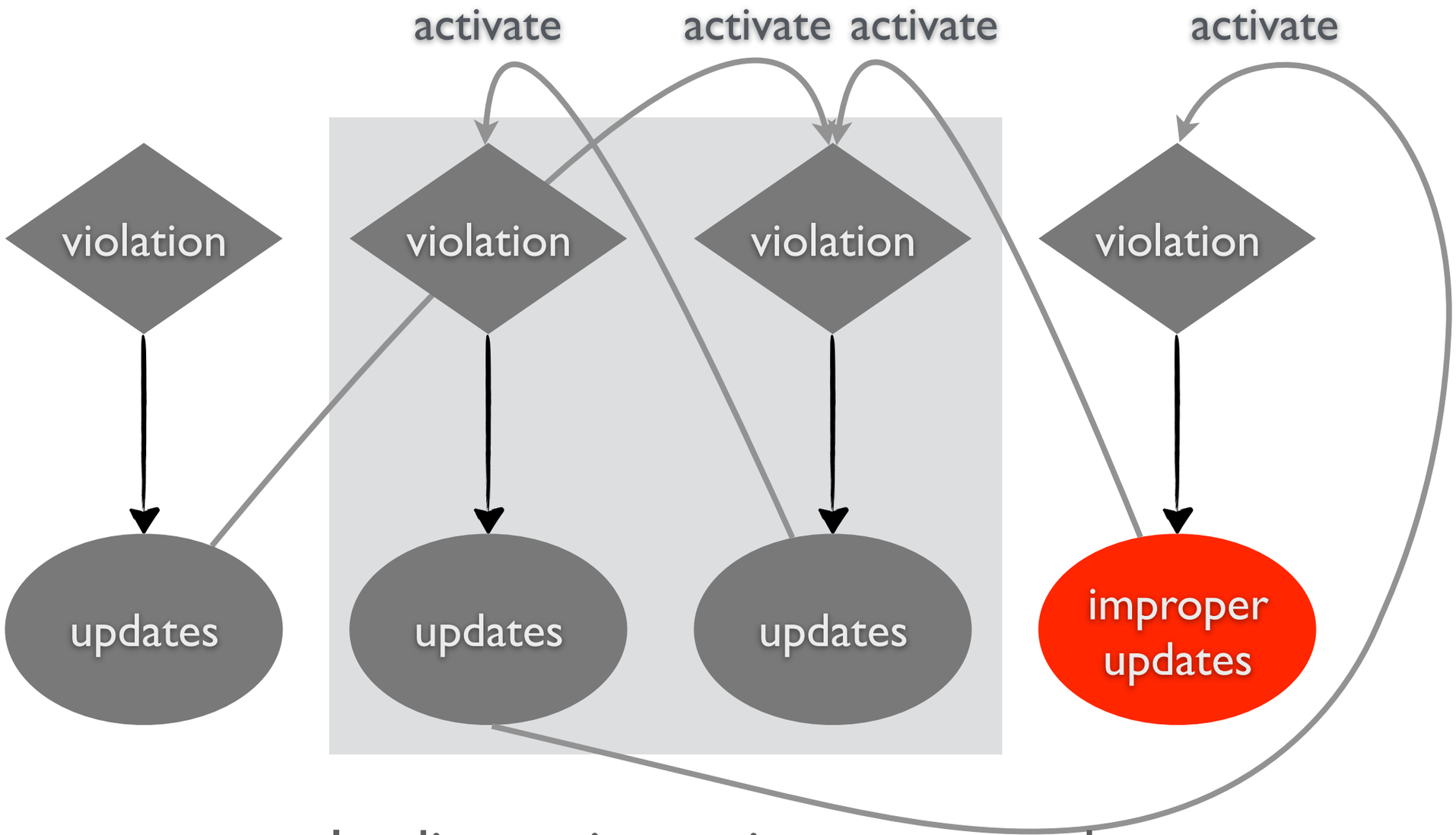
managing complexity in control software



multiple dependencies can lead to conflict

- conflict resolution: module-level priority, module composition
- limitation: coarse-grained, manual

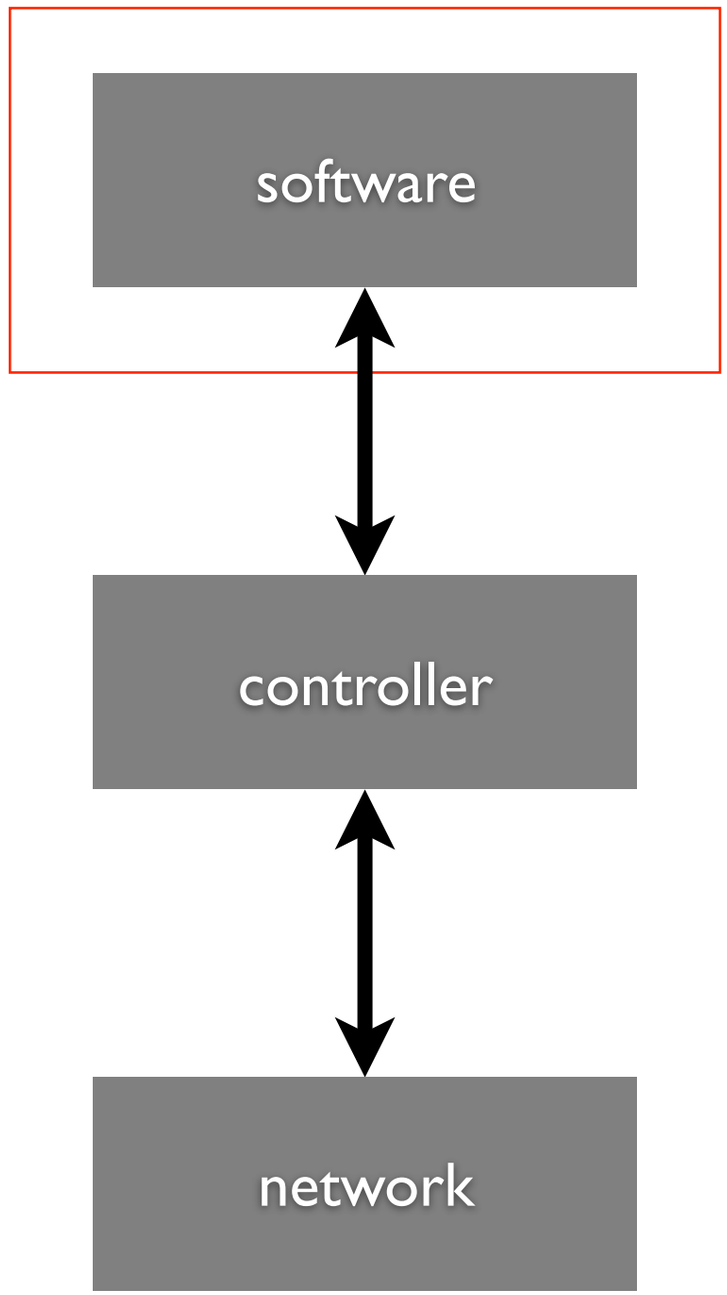
managing complexity in control software



updates may go wrong, leading to inconsistent network states

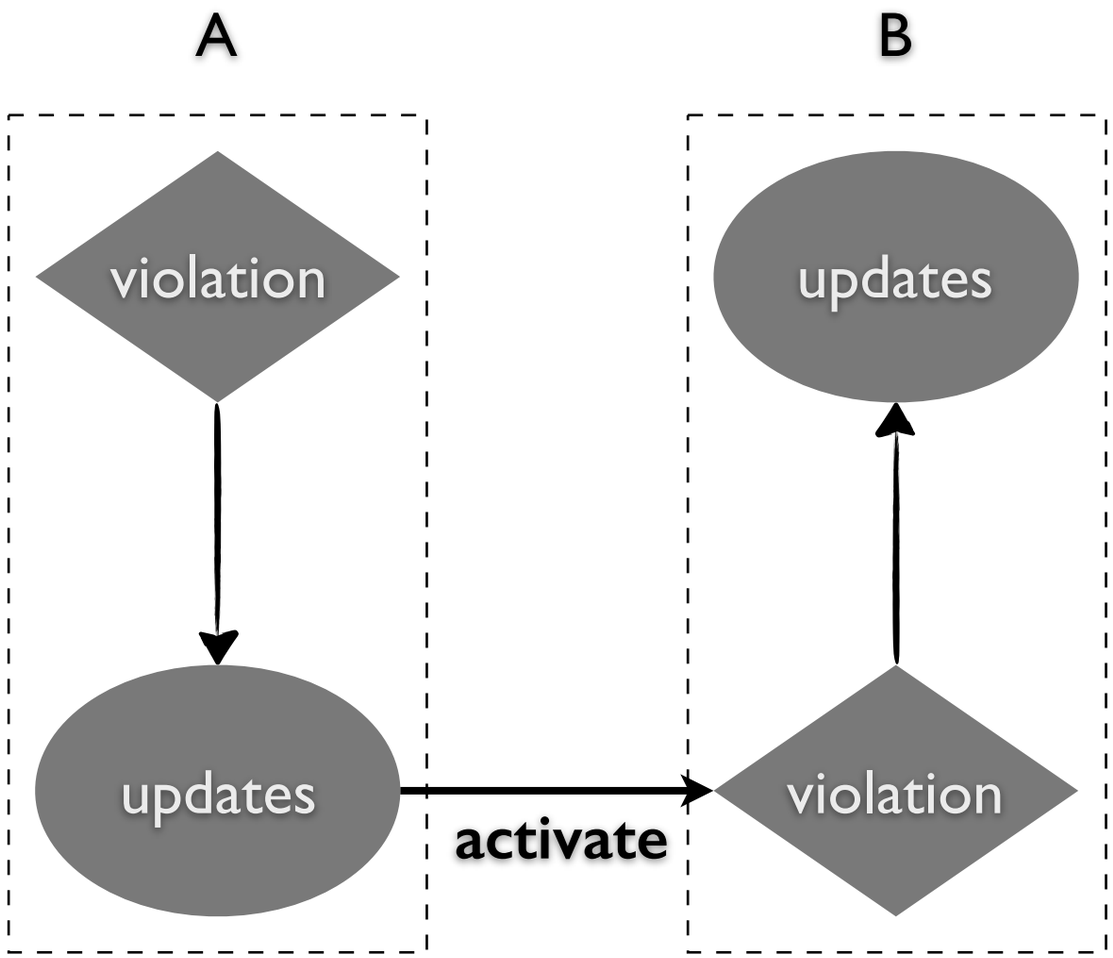
- **debugging and verification**: detect inconsistent states, locate events leading to an error
- **limitation**: post-mortem, not revealing incorrect control logic embedded in the software

approach: reasoning support



- **automated:** reduce human involvement with formal tool (SMT solver)
- **finer-grained:** operation-level
- **static:** prior-to deployment,
- **logic based reasoning:** derive proper interactions among controls

dependency

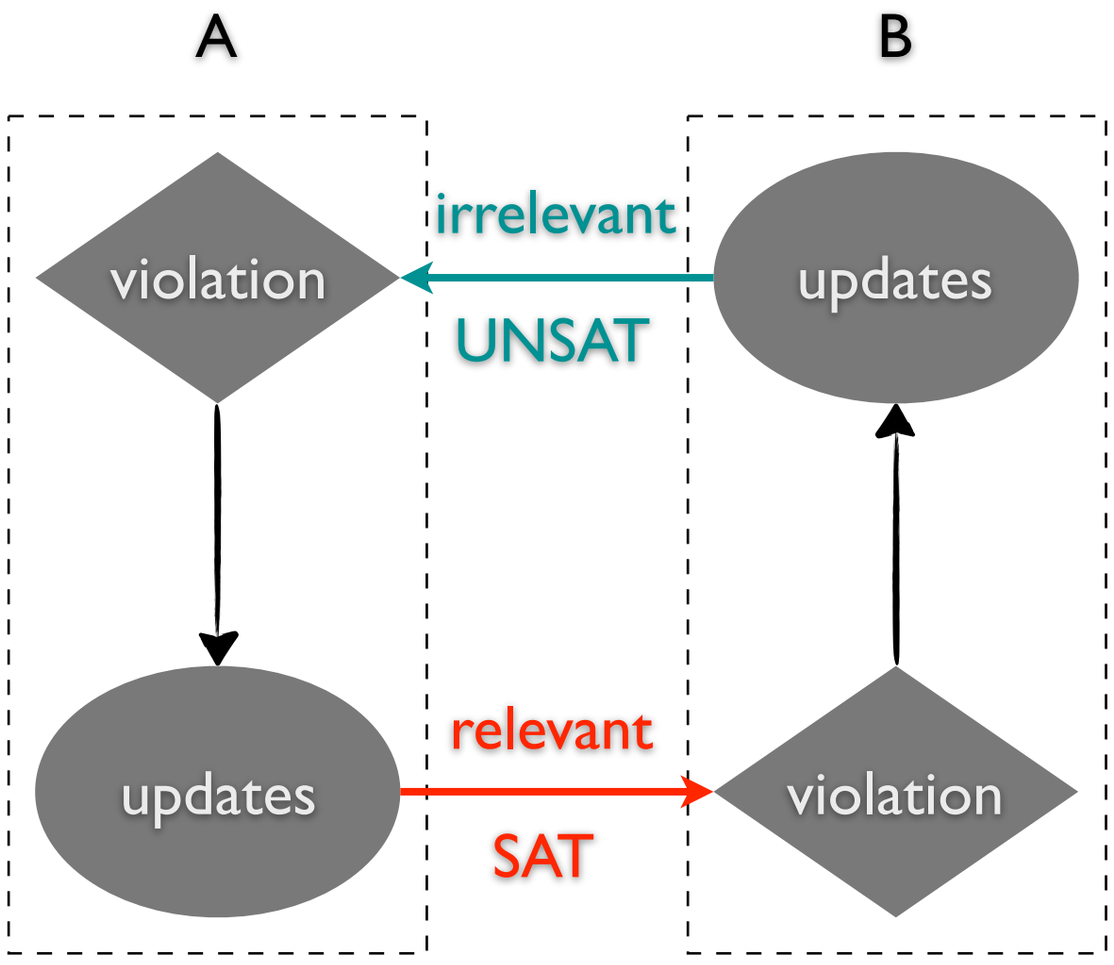


operation A depends on B

(1) A update can activate B

(2) B update never activates A

dependency by (ir)relevance reasoning



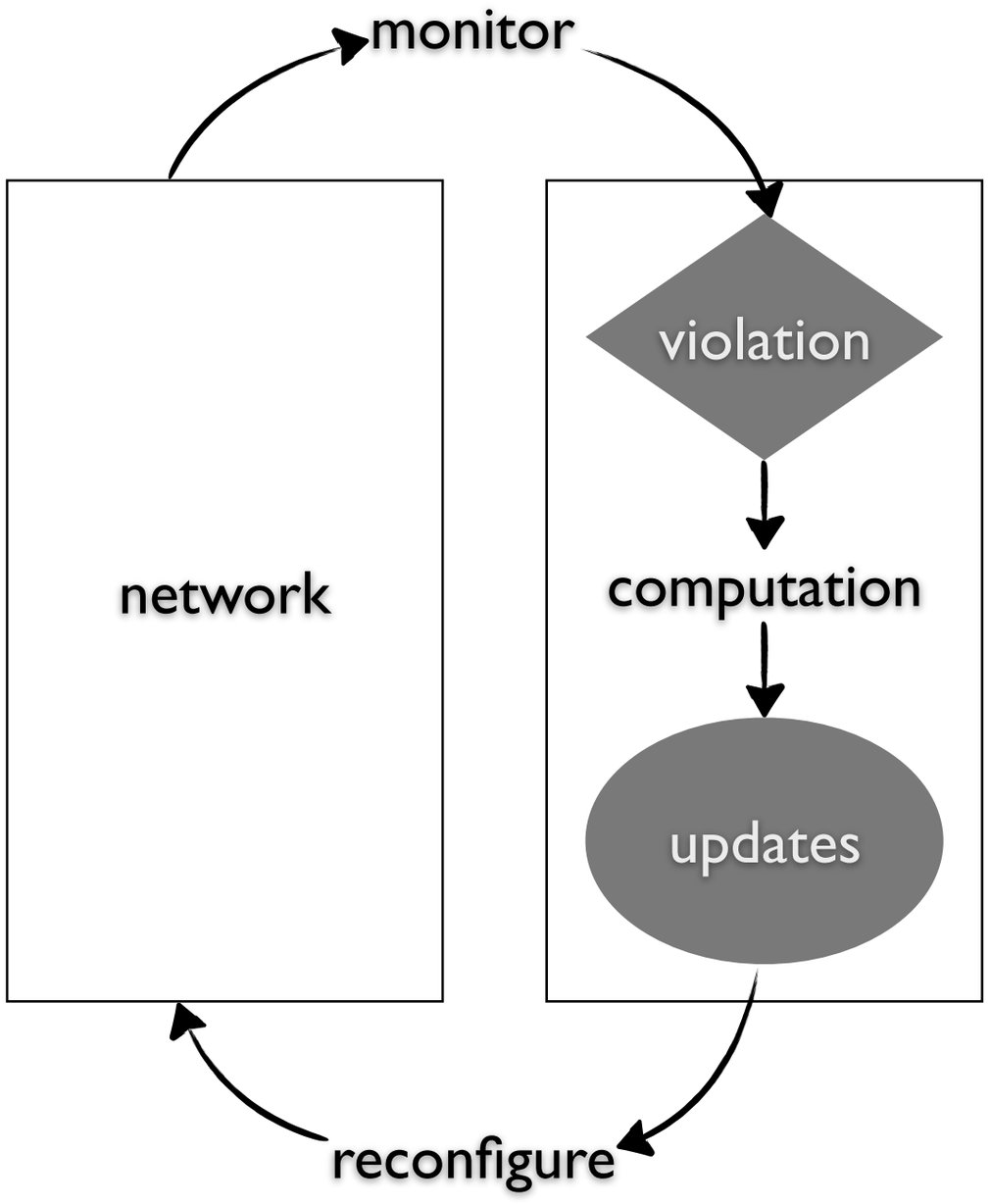
operation A depends on B

(1) A is relevant to B: can find a B update such that violates A

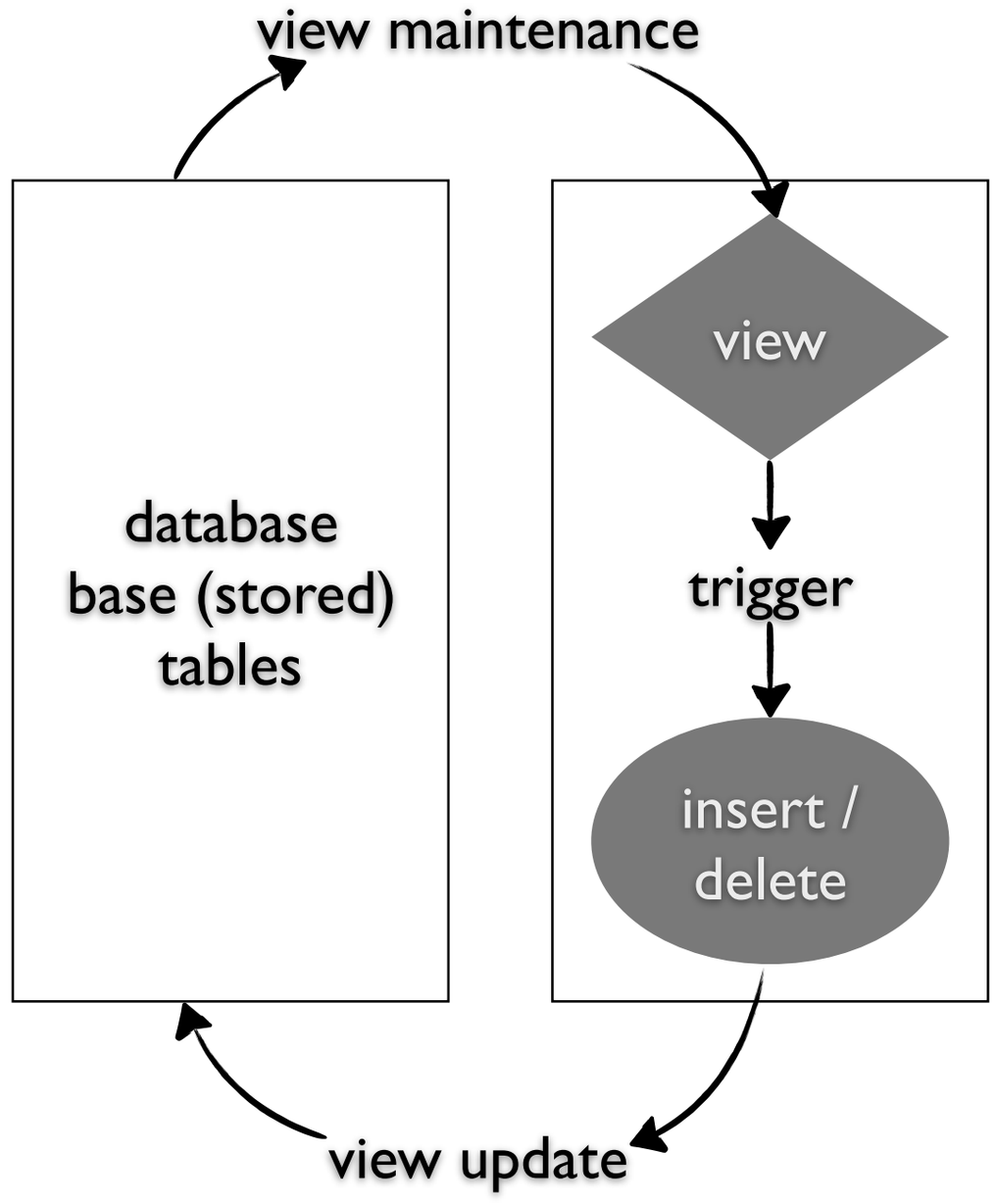
(2) B is irrelevant to A: cannot find a B update that violates A

formal model

SDN control loop



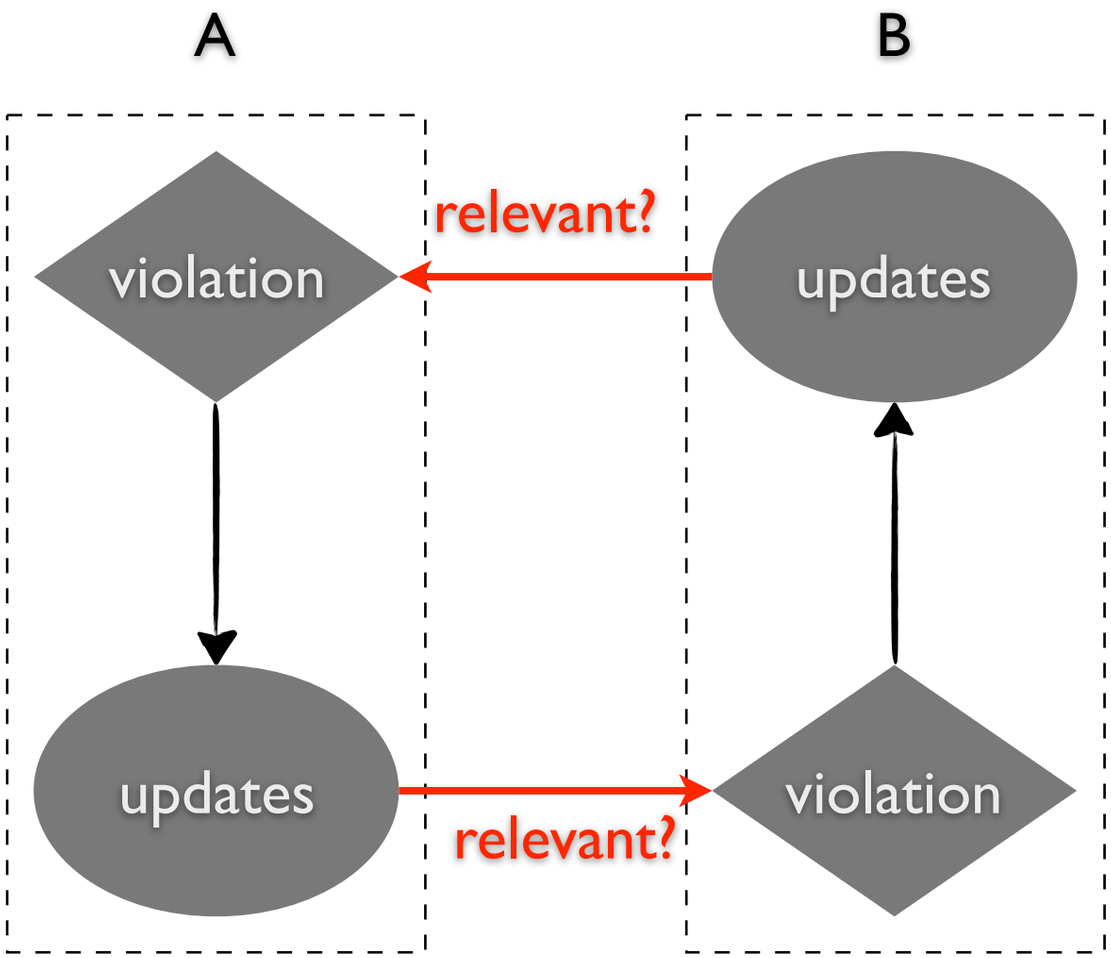
a unified database representation



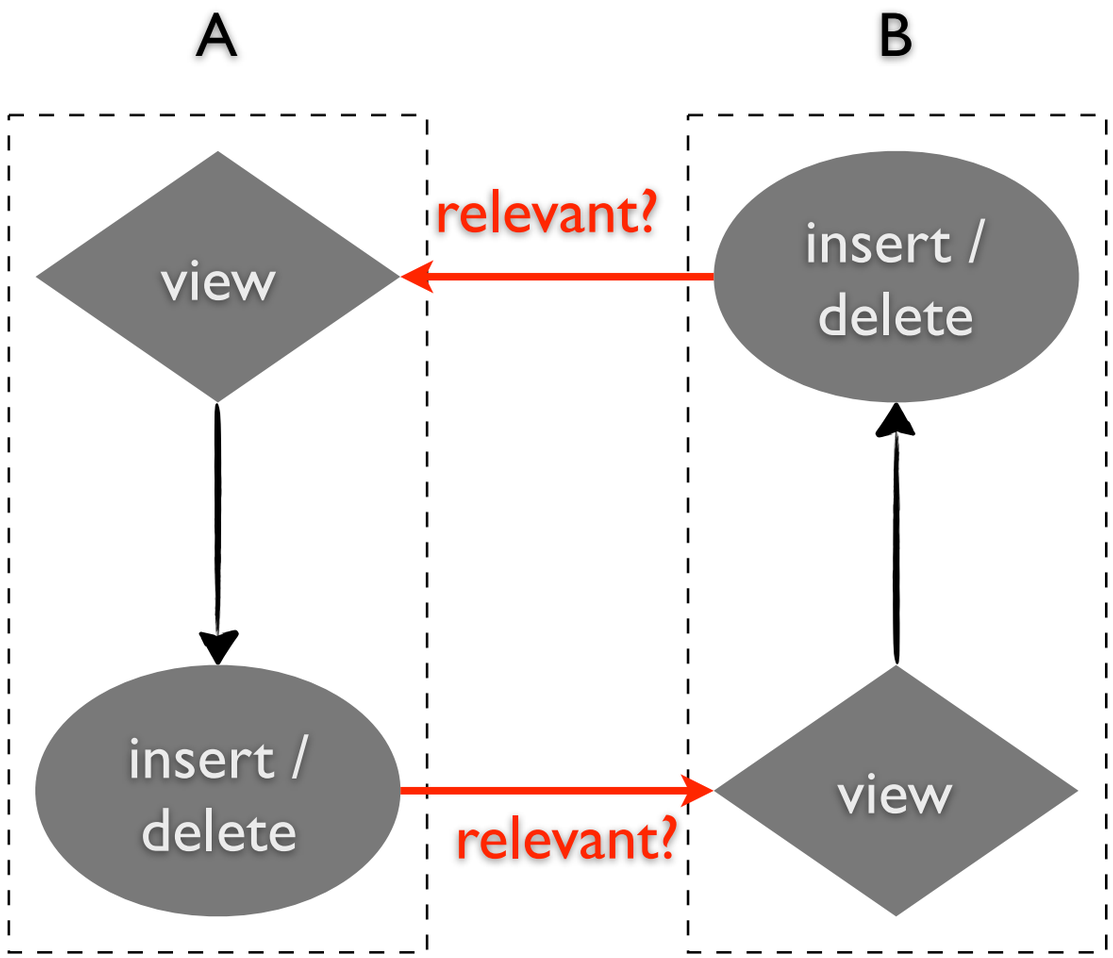
ravel: a database-defined network [SOSR'16]
ravel-net.org

database irrelevance reasoning

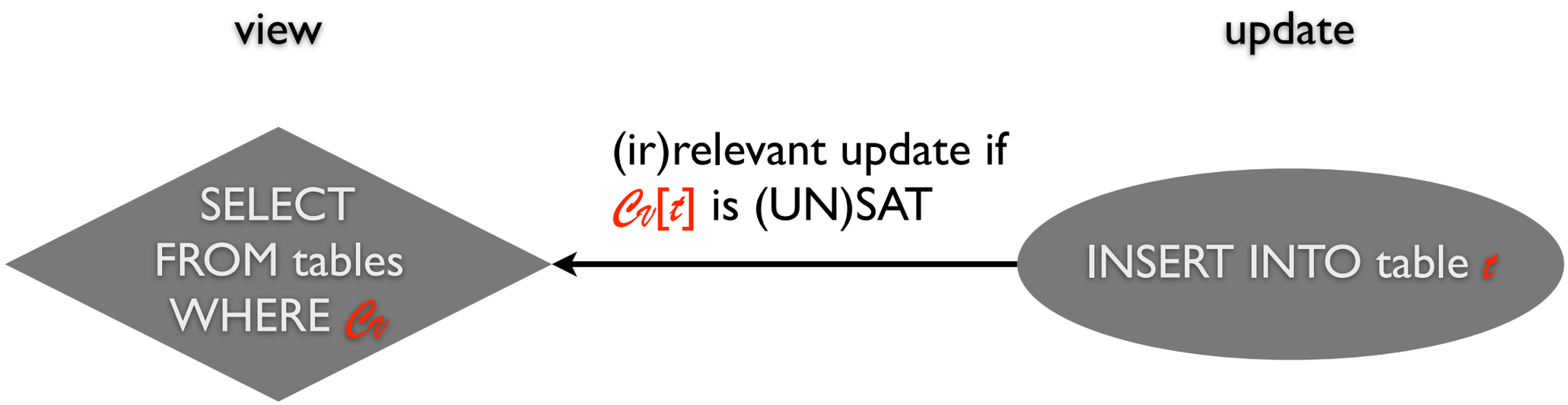
irrelevance reasoning for SDN

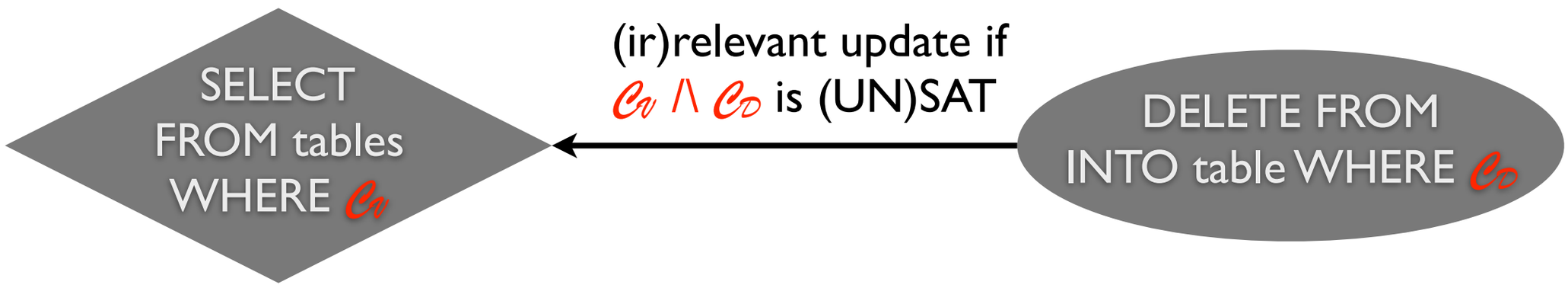


detect irrelevant database updates

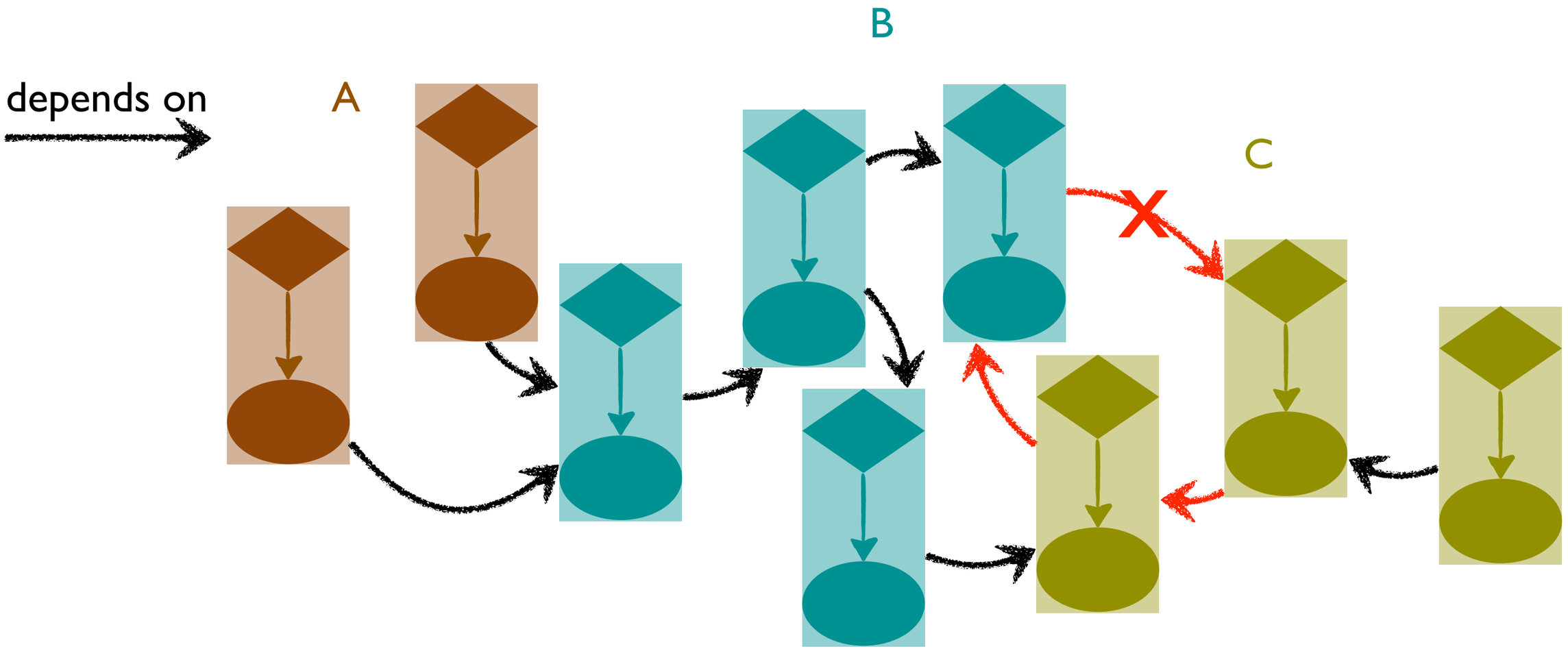


(ir)relevant database update





usage: synthesize orchestrator



construct dependency graph

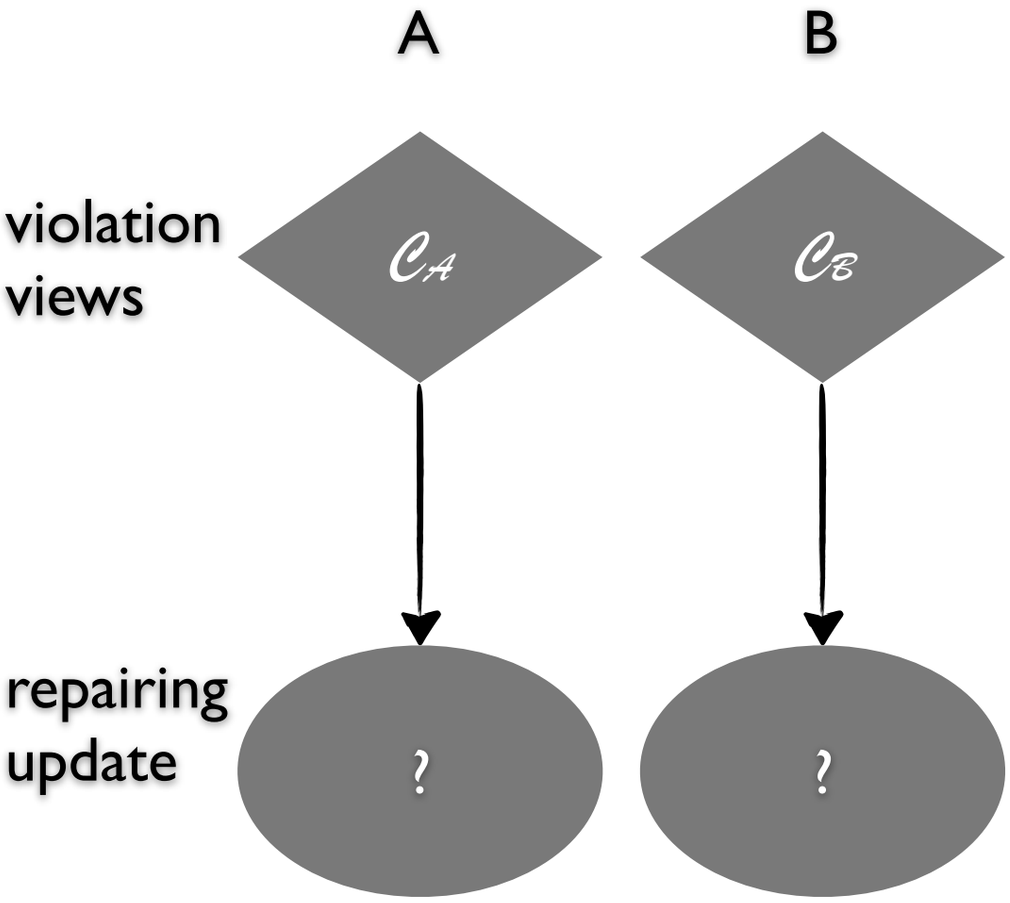
topological sort

- remove conflicts with user guidance
- assign each update a stratum number

synthesize a master orchestrator

- activate an update only when all updates with smaller stratum numbers have completed

usage: reasoning with partial information



conflict-free guarantee

if $\neg C_A \supset \neg C_B$, then A is guaranteed to be irrelevant to B

(corollary: synthesize conflict-free updates for A regarding B by rewriting C_A to $C_A \vee C_B$)

feasibility of conflict-free update

if $\neg C_A \wedge \neg C_B$ is SAT, there exists some A update that is irrelevant to B

infeasibility of conflict-free updates

if $\neg C_A \wedge \neg C_B$ is UNSAT, no A update exists that is irrelevant to B

thank you

backup

open questions

obtain the database representation

- use *Ravel*, a database-defined control platform
  - ravel-net.org

extract the database representation from arbitrary control software

- manually construct a map between data objects and database tables
- automatically convert data updates to DB write with conditions?
- extract view condition?

limitation

distribution and concurrency

- the network data plane is a distributed system with concurrent updates
- SDN relies on multiple controller for scalability

combine DB concurrency control and irrelevance reasoning?