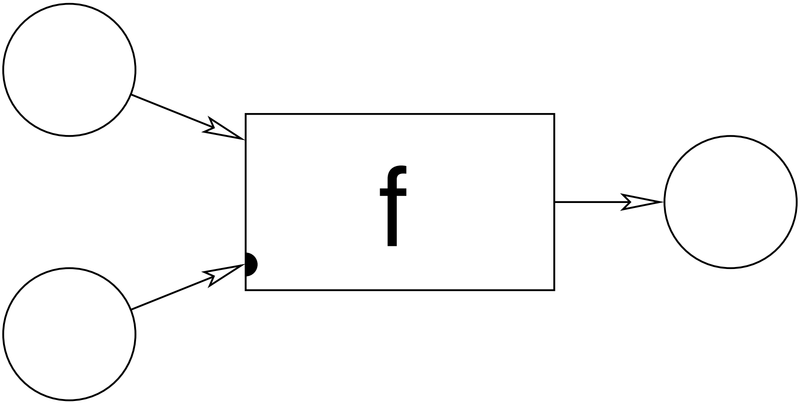
# Propagator Networks

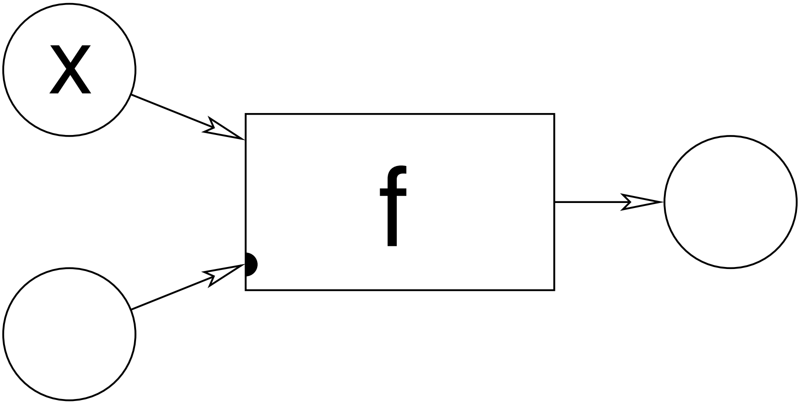
Alexey Radul

March 12, 2009, DIG Seminar



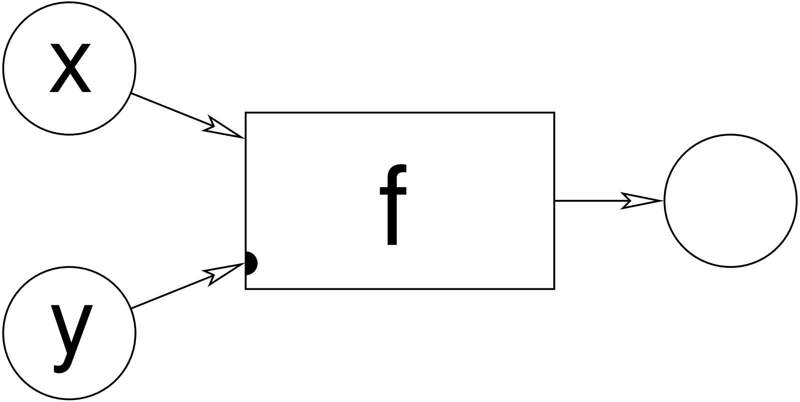
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



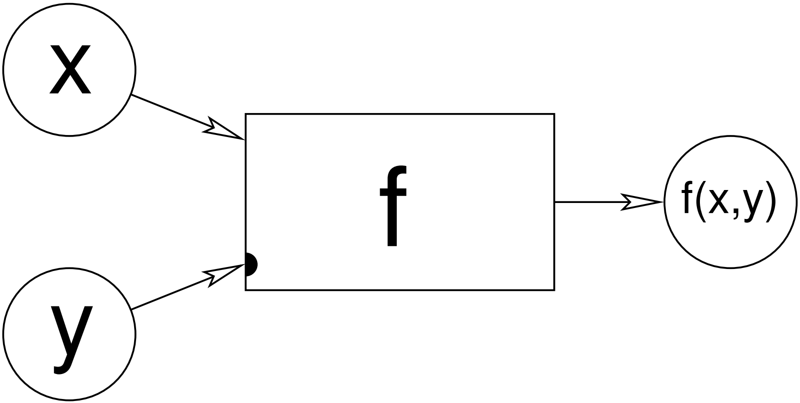
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



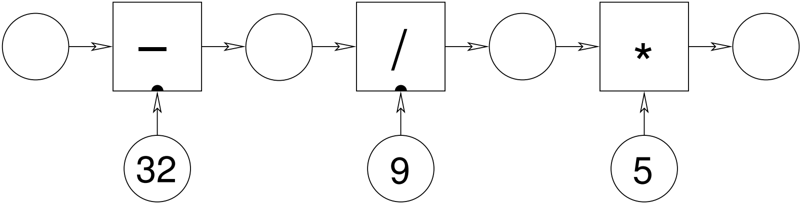
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



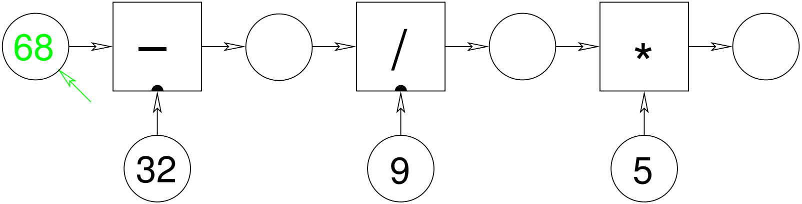
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



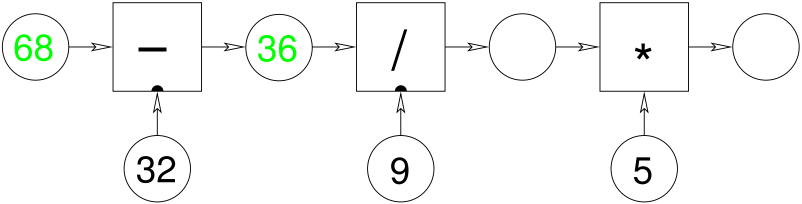
# Network them, and values propagate

this distributes naturally



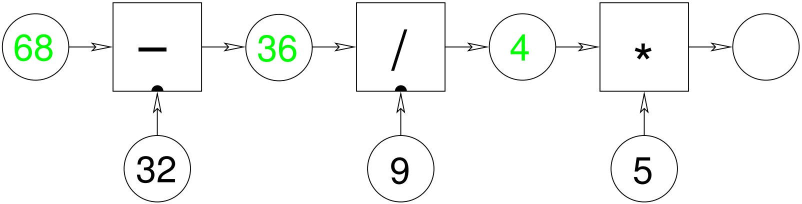
# Network them, and values propagate

this distributes naturally



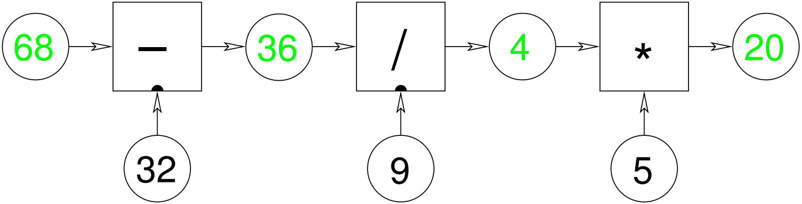
# Network them, and values propagate

this distributes naturally



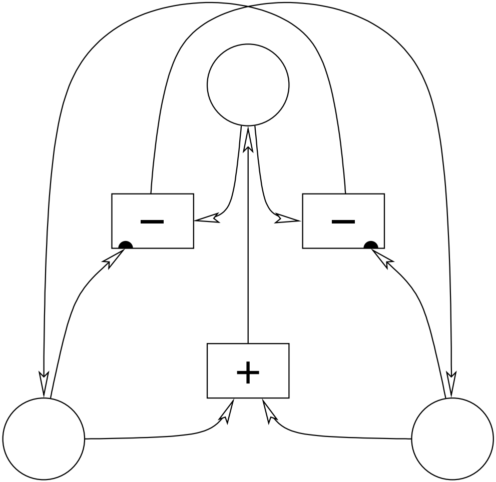
# Network them, and values propagate

this distributes naturally

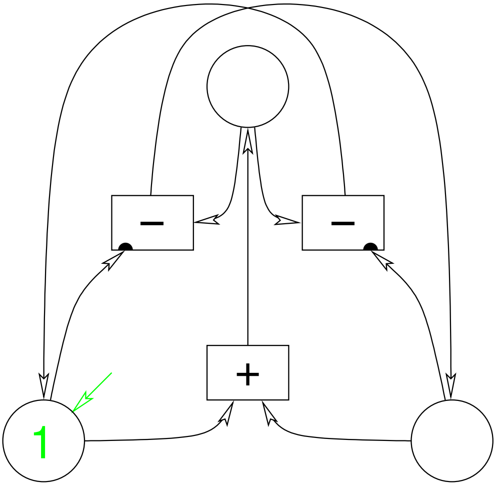


# Network them, and values propagate

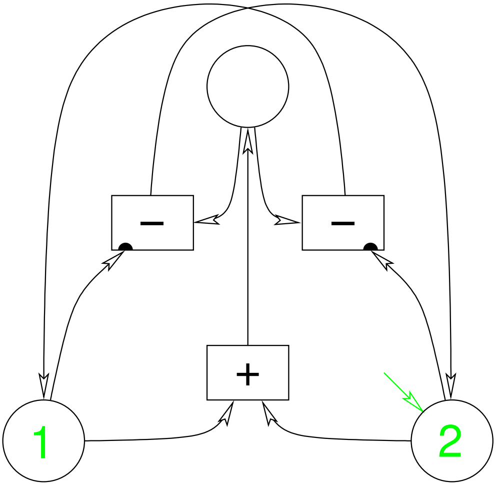
this distributes naturally



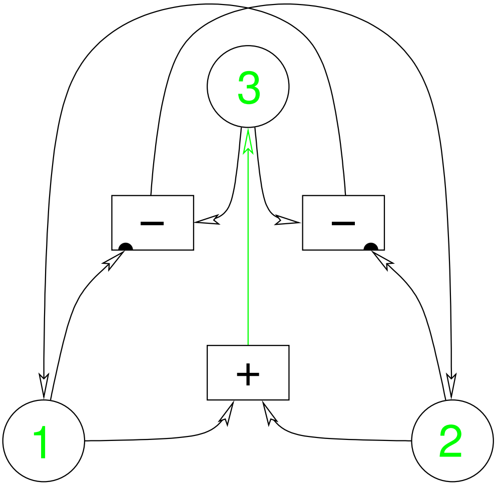
Win: Constraints are just piles of mutually inverse propagators



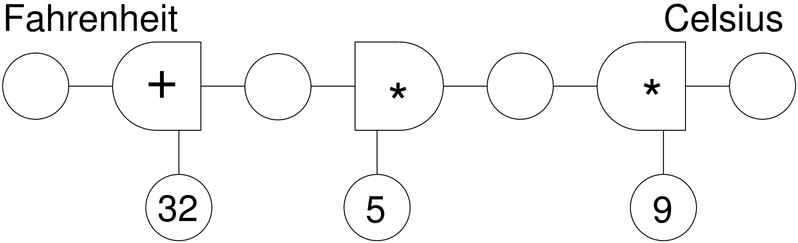
Win: Constraints are just piles of mutually inverse propagators



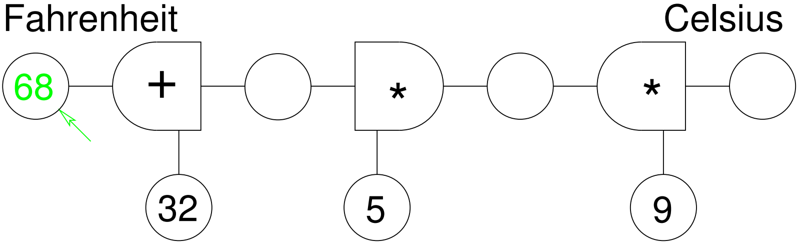
Win: Constraints are just piles of mutually inverse propagators



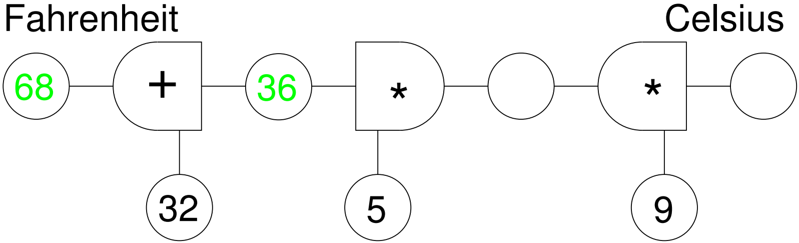
Win: Constraints are just piles of mutually inverse propagators



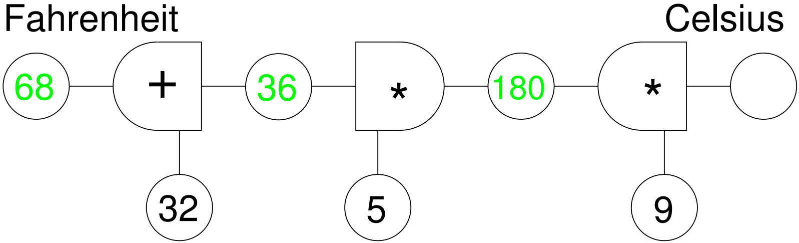
Win: Constraints compose into multidirectional computations



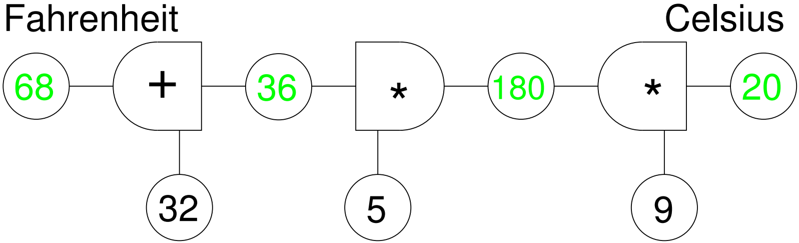
Win: Constraints compose into multidirectional computations



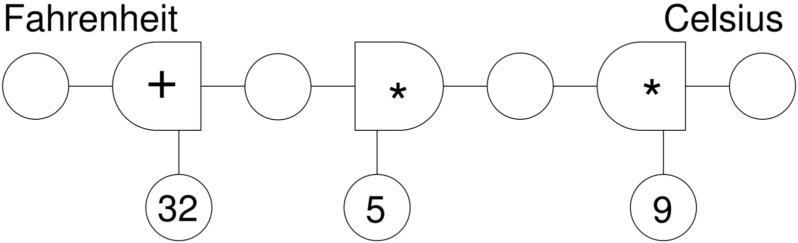
Win: Constraints compose into multidirectional computations



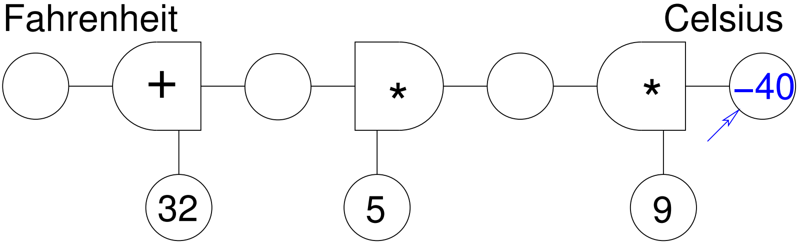
Win: Constraints compose into multidirectional computations



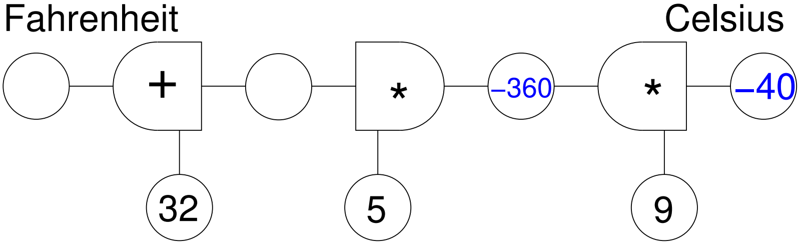
Win: Constraints compose into multidirectional computations



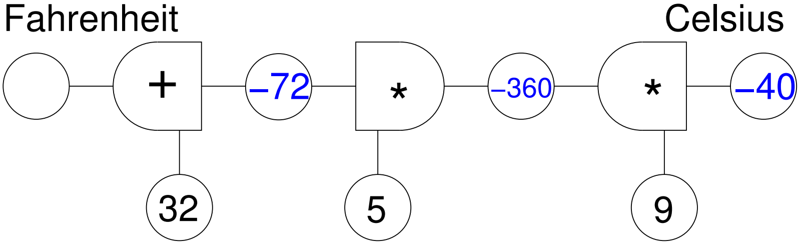
Win: Constraints compose into multidirectional computations



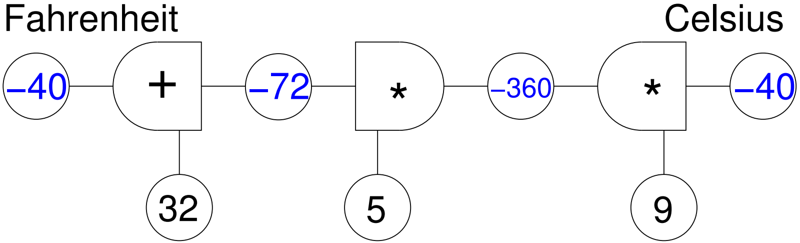
Win: Constraints compose into multidirectional computations



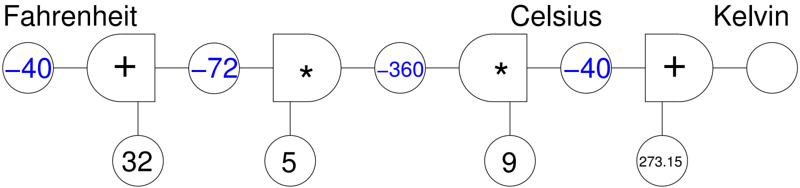
Win: Constraints compose into multidirectional computations



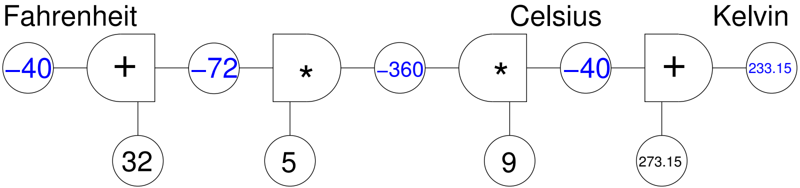
Win: Constraints compose into multidirectional computations



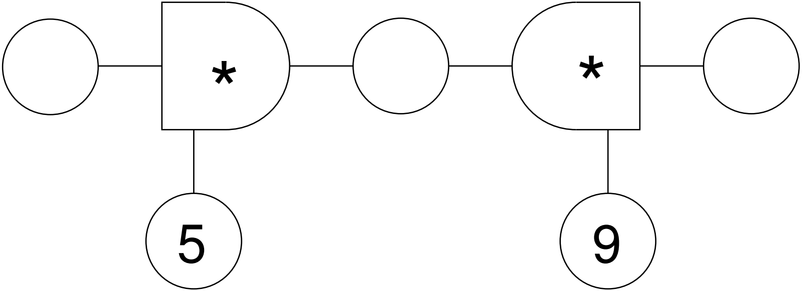
Win: Constraints compose into multidirectional computations



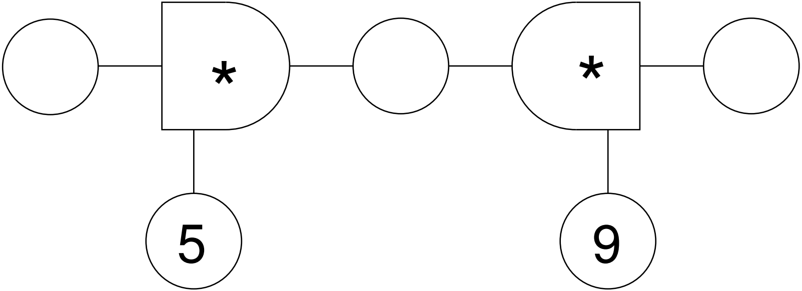
which can grow incrementally without adjusting explicit controls



which can grow incrementally without adjusting explicit controls



But: A cell can get stuff from multiple sources



But: A cell can get stuff from multiple sources

Is this bad?

# “Old View”: Cells hold values

# “Old View”: Cells hold values

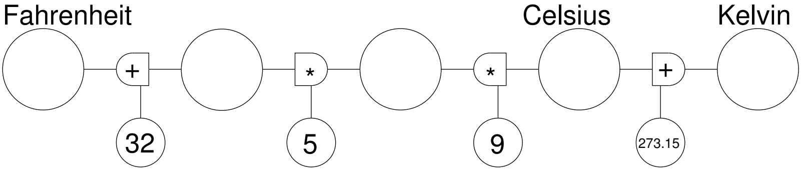
Leads to all kinds of trouble

- precedence
- overwriting
- infinite reactions and fights
- . . .

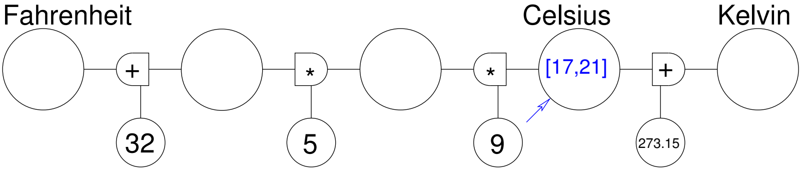
New View: Cells hold **information about** values

# New View: Cells hold **information about** values

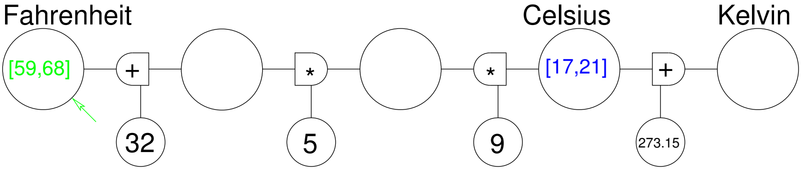
and merge it as it comes in from many sources



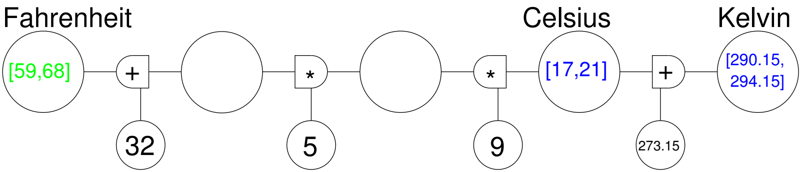
E.g. interval arithmetic is partial information



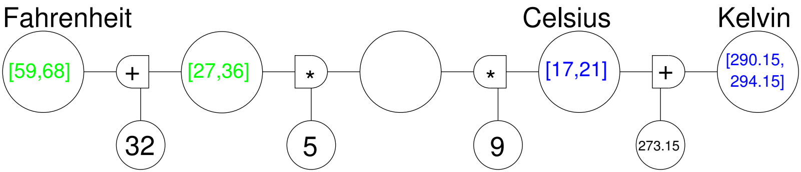
E.g. interval arithmetic is partial information



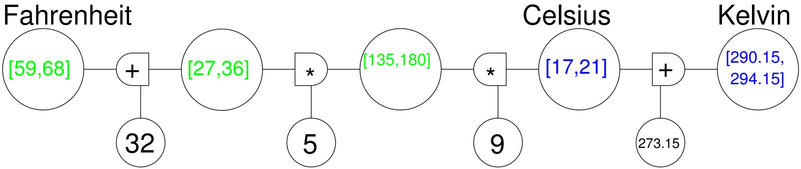
E.g. interval arithmetic is partial information



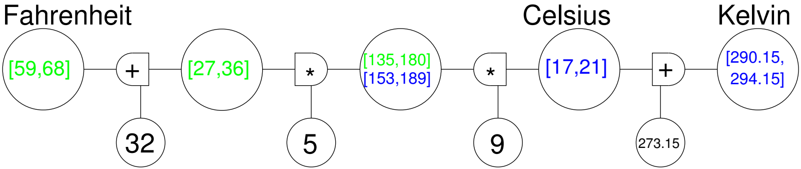
E.g. interval arithmetic is partial information



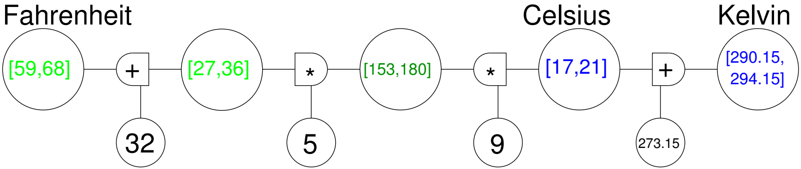
E.g. interval arithmetic is partial information



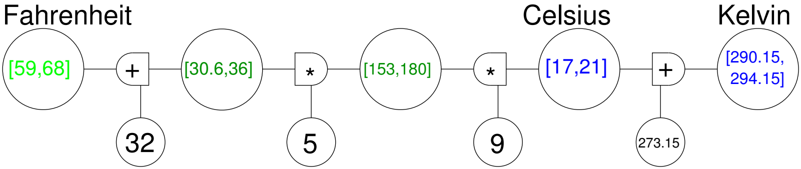
E.g. interval arithmetic is partial information



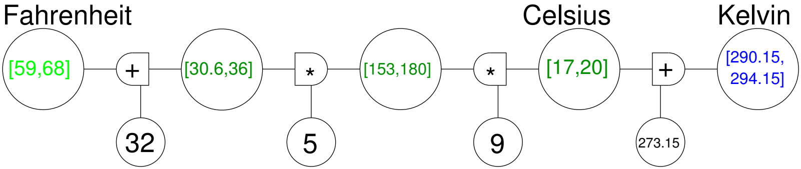
E.g. interval arithmetic is partial information



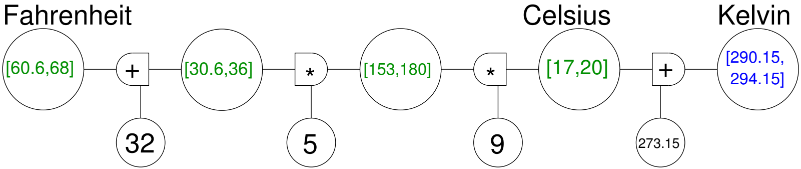
E.g. interval arithmetic is partial information



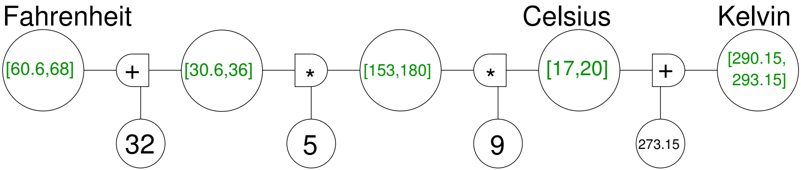
E.g. interval arithmetic is partial information



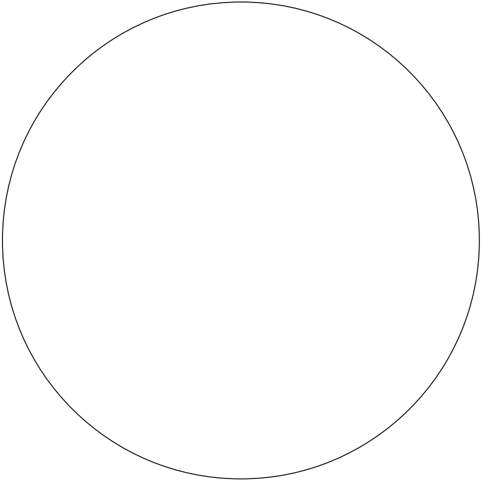
E.g. interval arithmetic is partial information



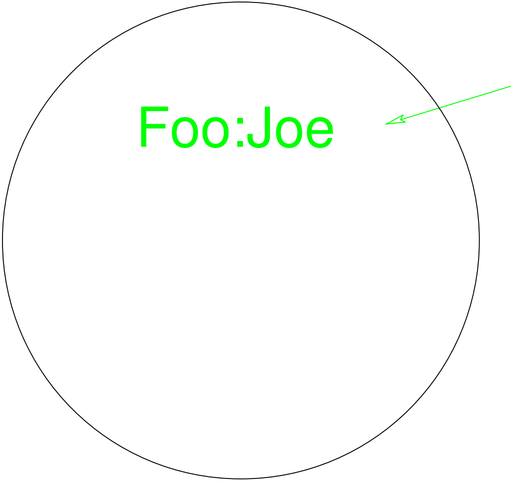
E.g. interval arithmetic is partial information



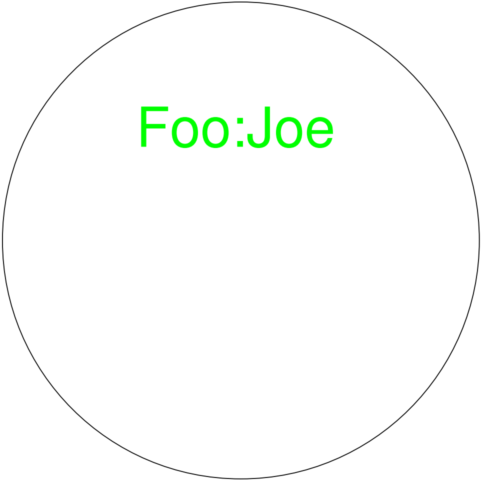
E.g. interval arithmetic is partial information



Win: Truth maintenance is partial information

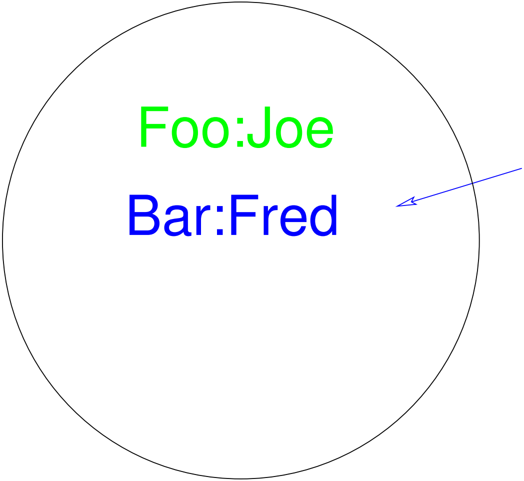


Win: Truth maintenance is partial information



Foo:Joe

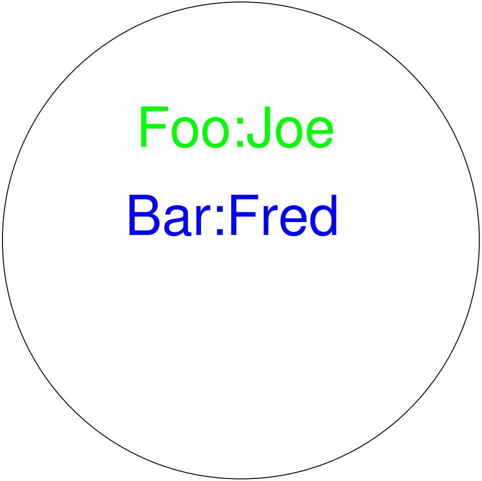
Win: Truth maintenance is partial information



Foo:Joe

Bar:Fred

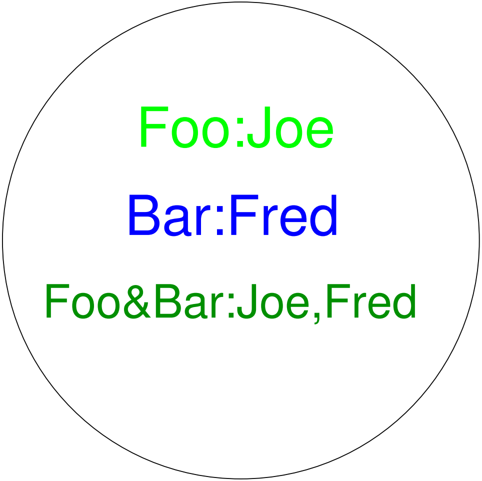
Win: Truth maintenance is partial information



Foo:Joe

Bar:Fred

Win: Truth maintenance is partial information



Foo:Joe

Bar:Fred

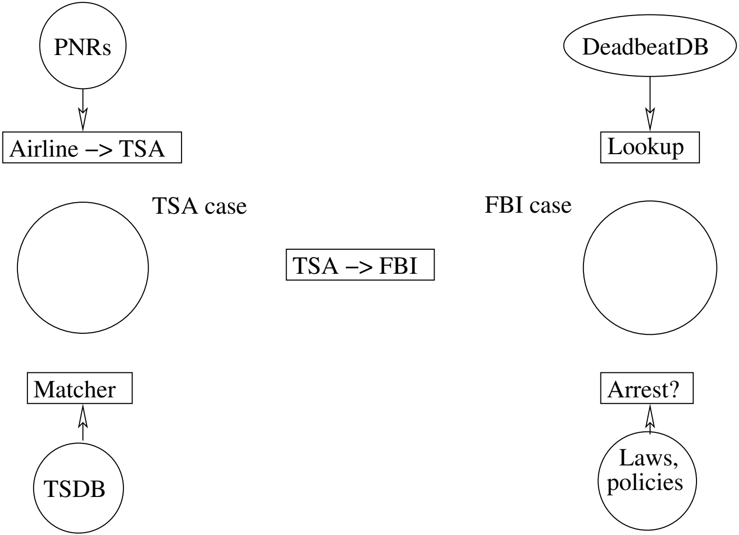
Foo&Bar:Joe,Fred

Win: Truth maintenance is partial information

Win: Making `merge` generic decouples the accident of kind of accumulator from the essence of propagation

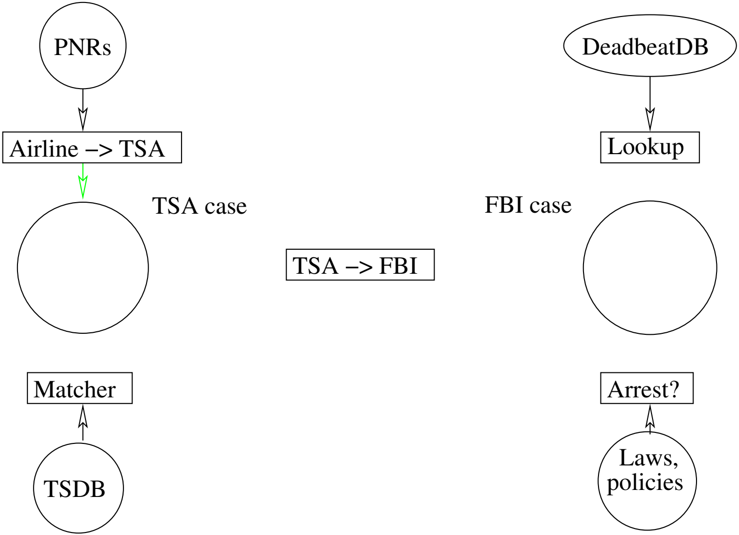
Win: Making `merge` generic decouples the accident of kind of accumulator from the essence of propagation

and now we can use many different kinds of accumulators



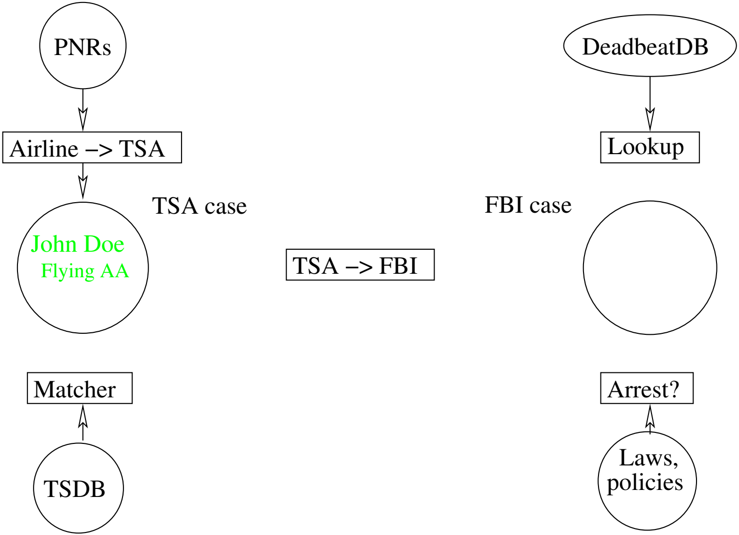
# And much can look like propagation

if you squint



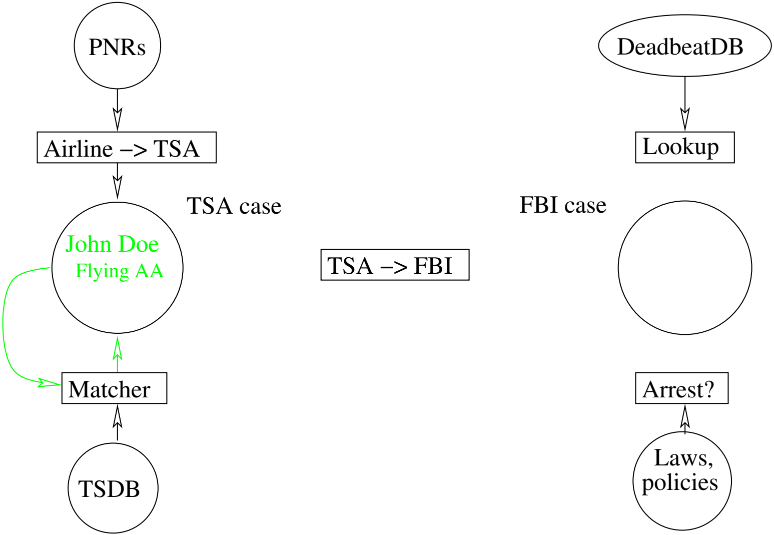
# And much can look like propagation

if you squint



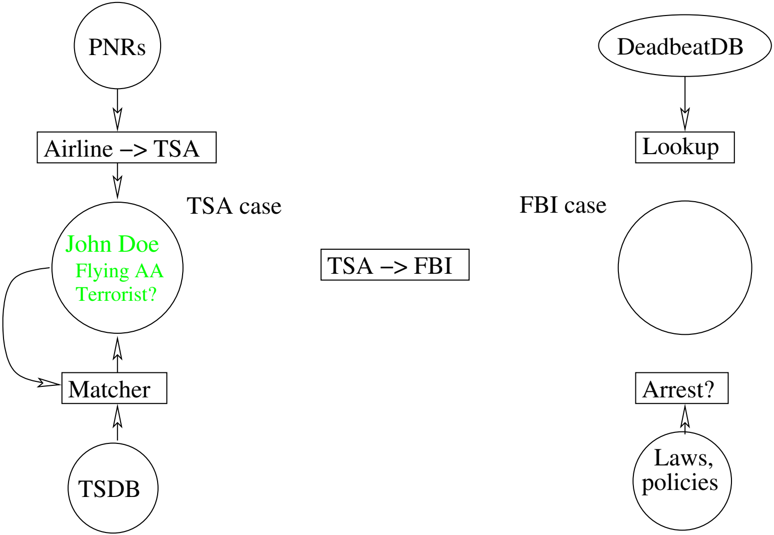
# And much can look like propagation

if you squint



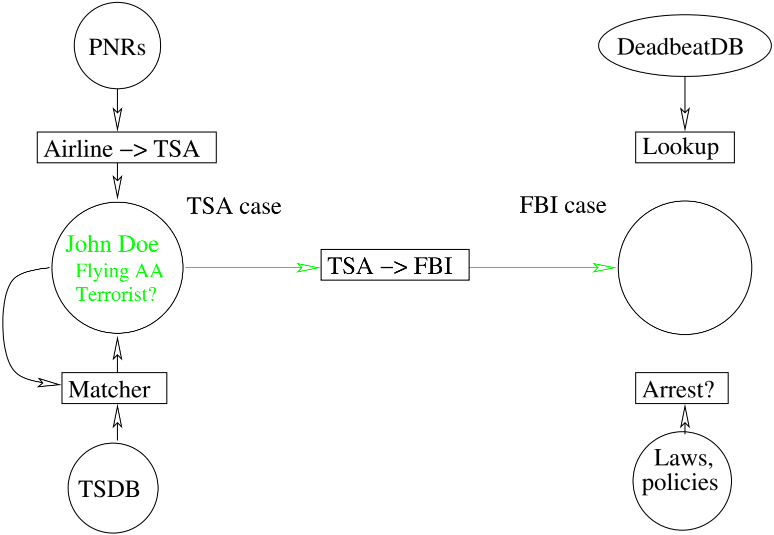
# And much can look like propagation

if you squint



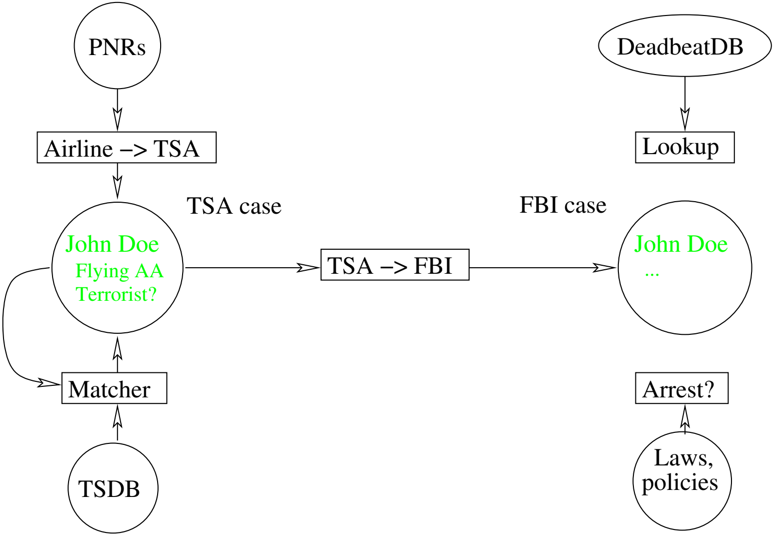
# And much can look like propagation

if you squint



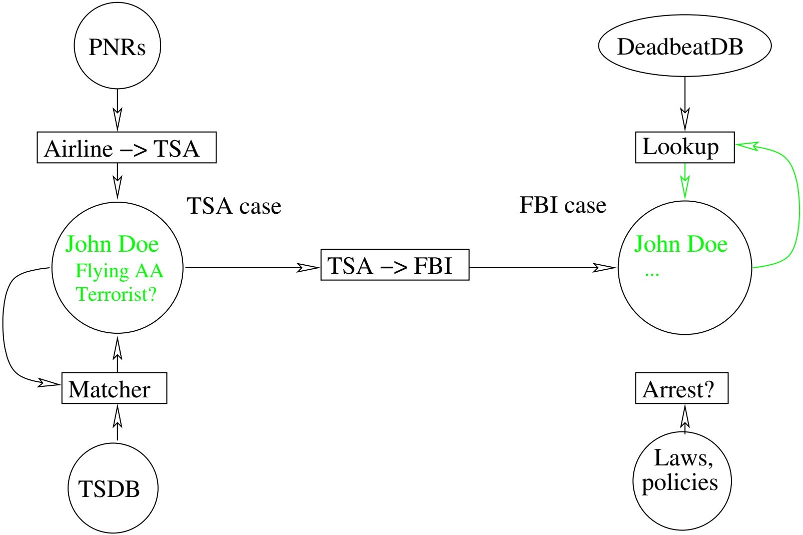
# And much can look like propagation

if you squint



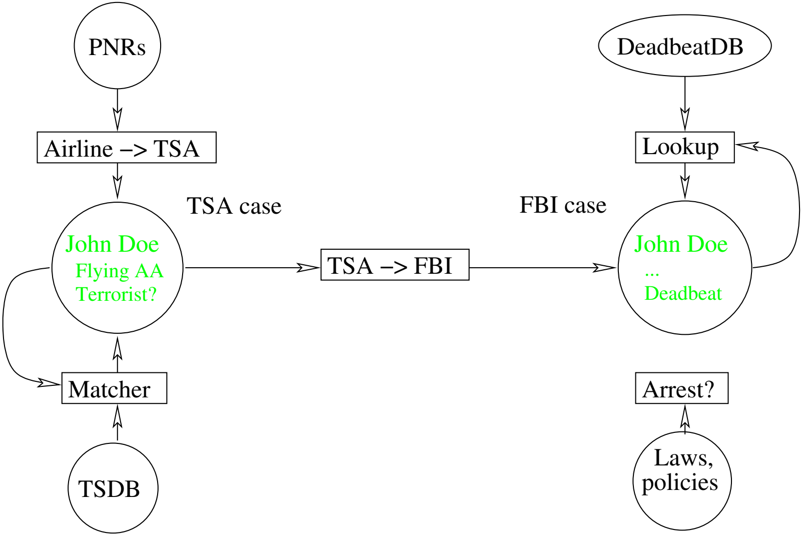
# And much can look like propagation

if you squint



# And much can look like propagation

if you squint



PNRs

Airline –> TSA

TSA case

John Doe
Flying AA
Terrorist?

Matcher

TSDB

TSA –> FBI

FBI case

DeadbeatDB

Lookup

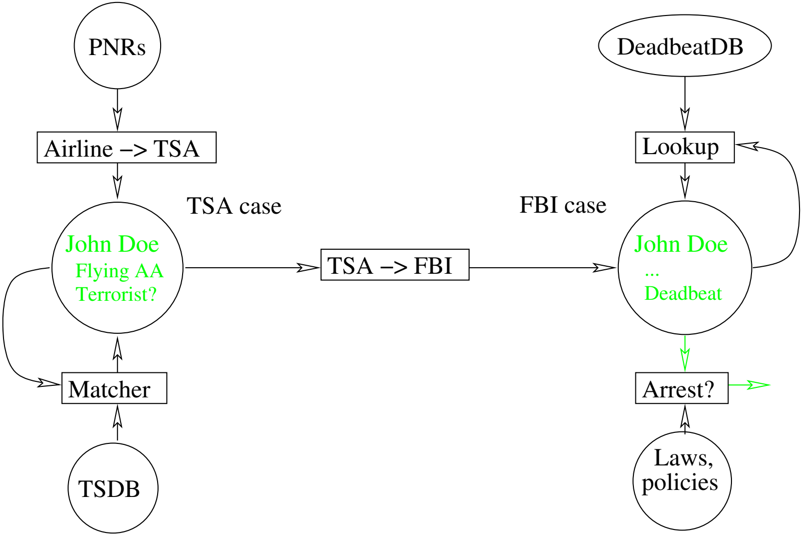
John Doe
...
Deadbeat

Arrest?

Laws,
policies

# And much can look like propagation

if you squint



# And much can look like propagation

if you squint

Pick the knowledge representation for your own problem, but

Partial information and propagator networks are essentially intertwined