

The (Abridged) Art of the Propagator, Abridged

Alexey Radul

March 23, 2009
International Lisp Conference

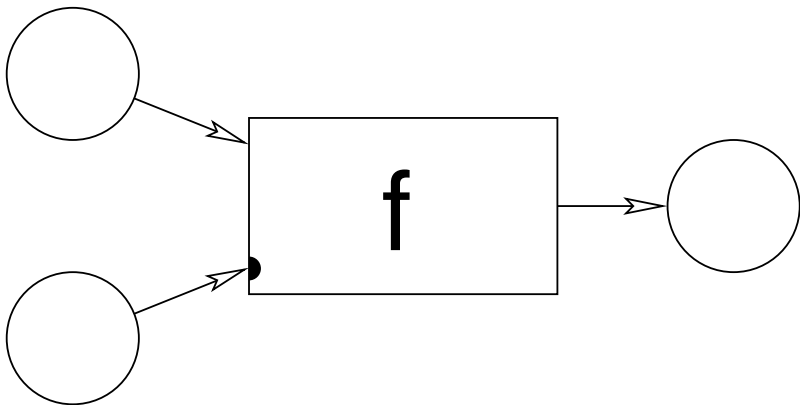
or

Computing with Boxes, Arrows, and Circles

or

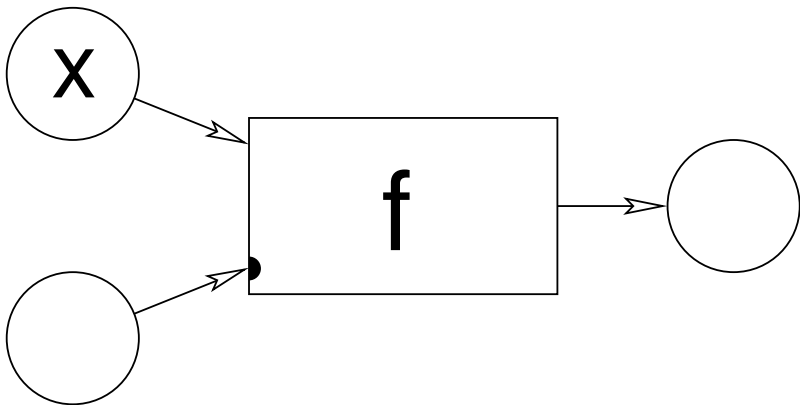
Computing with Boxes, Arrows, and Circles

- ▶ Why you might want to
- ▶ Something to get right if you try it



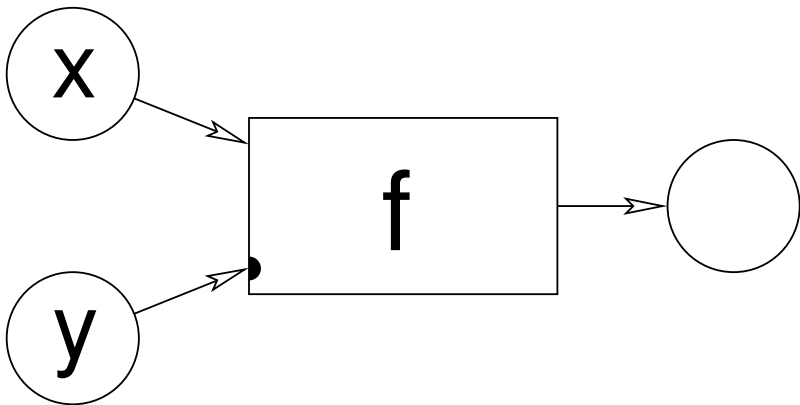
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



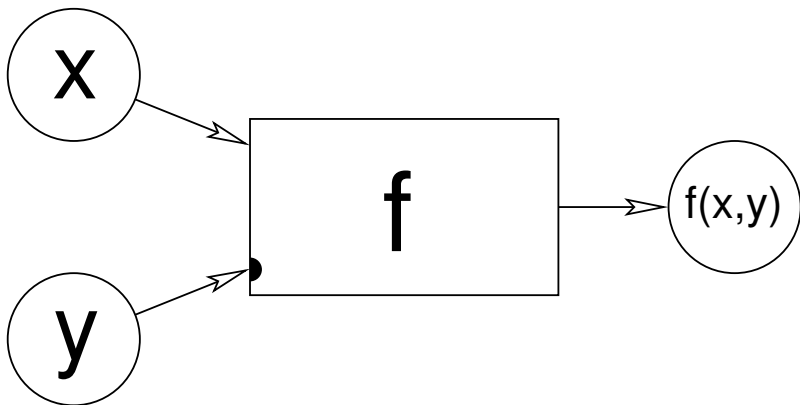
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



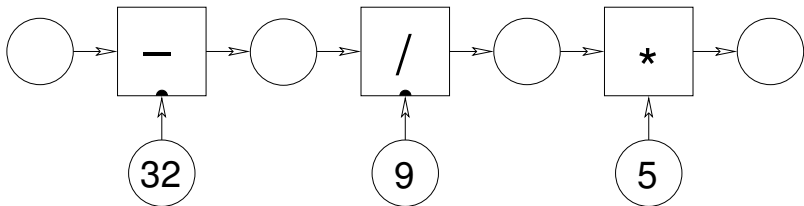
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



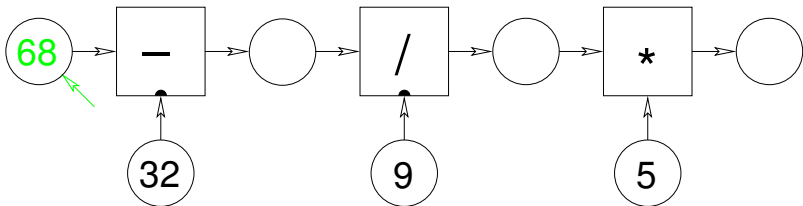
A propagator is a machine that reads some cells and can write to some cells

always on, asynchronous, stateless



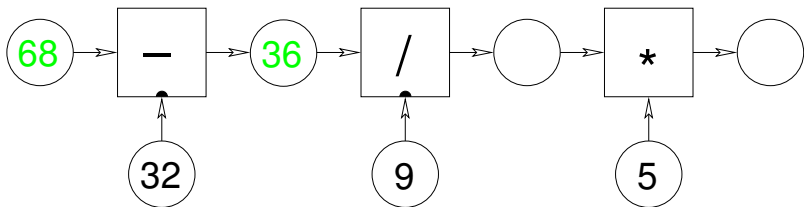
Network them, and values propagate

this distributes naturally



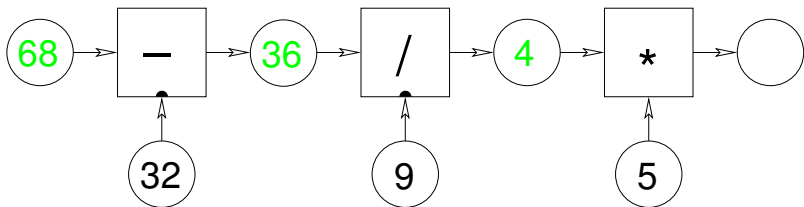
Network them, and values propagate

this distributes naturally



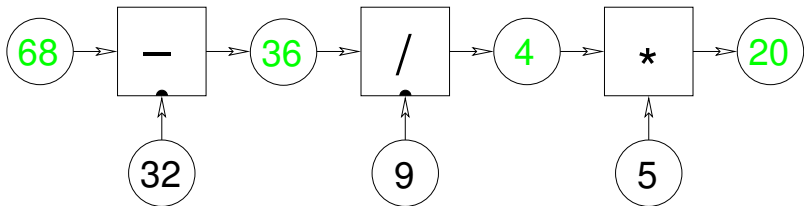
Network them, and values propagate

this distributes naturally



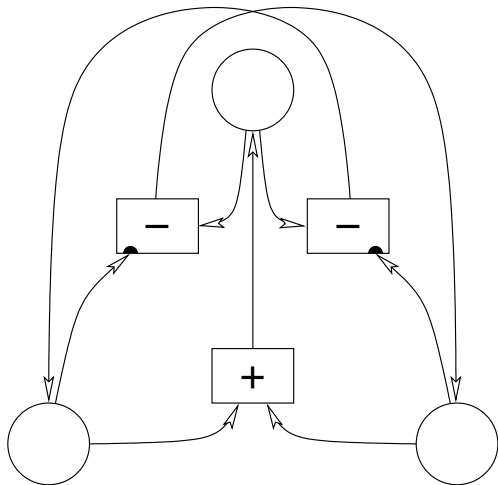
Network them, and values propagate

this distributes naturally

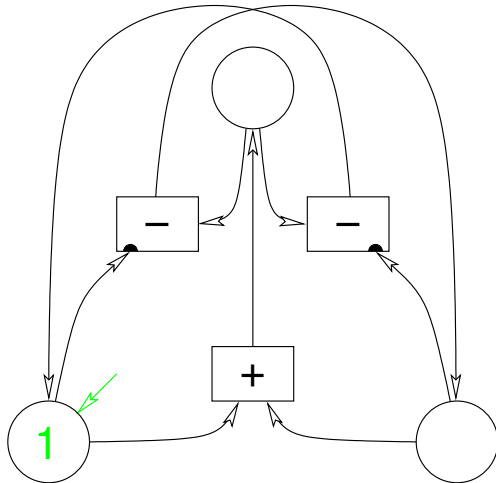


Network them, and values propagate

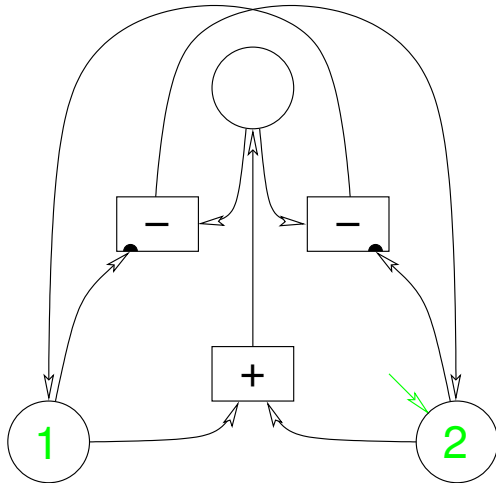
this distributes naturally



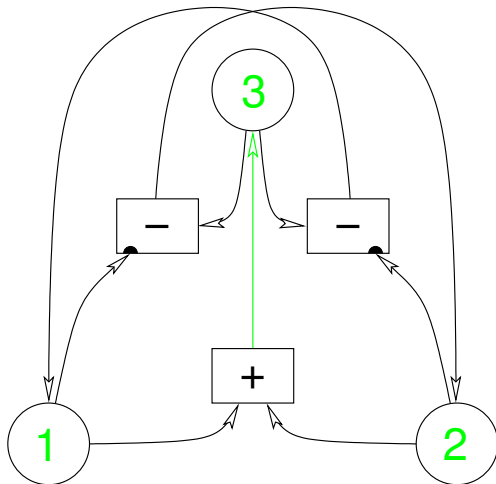
Win: Constraints are just piles of mutually inverse propagators



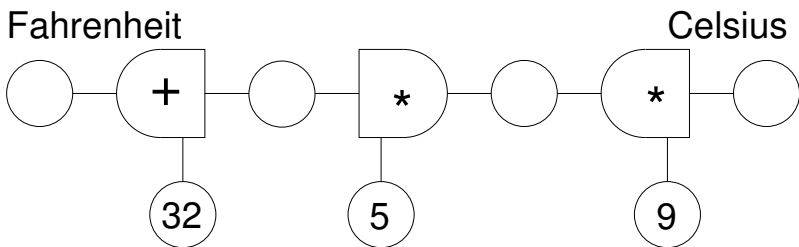
Win: Constraints are just piles of mutually inverse propagators



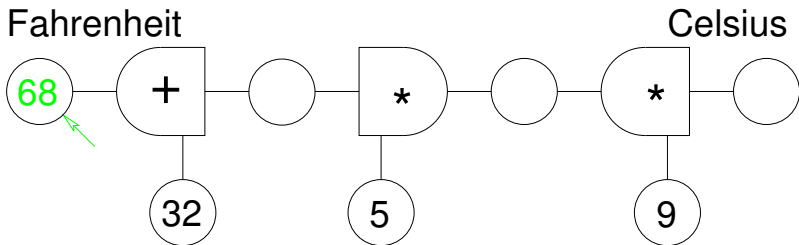
Win: Constraints are just piles of mutually inverse propagators



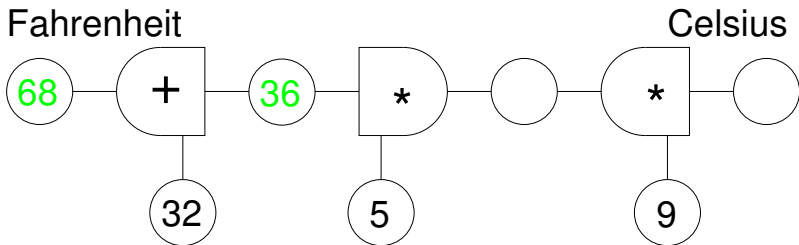
Win: Constraints are just piles of mutually inverse propagators



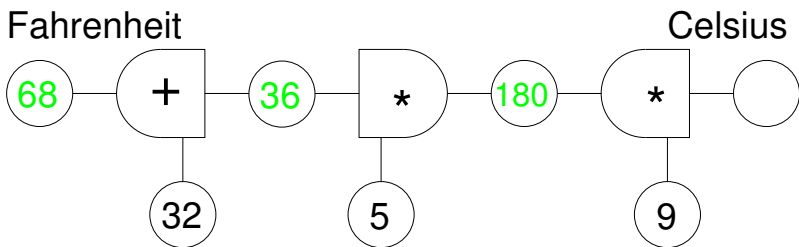
Win: Constraints compose into multidirectional computations



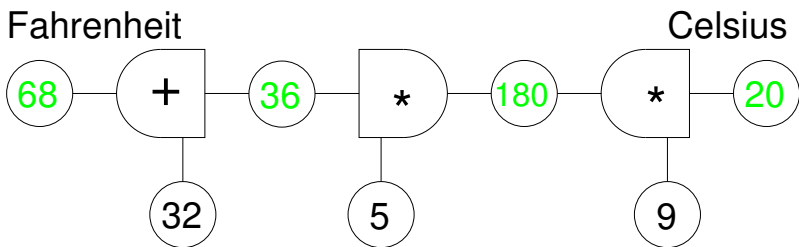
Win: Constraints compose into multidirectional computations



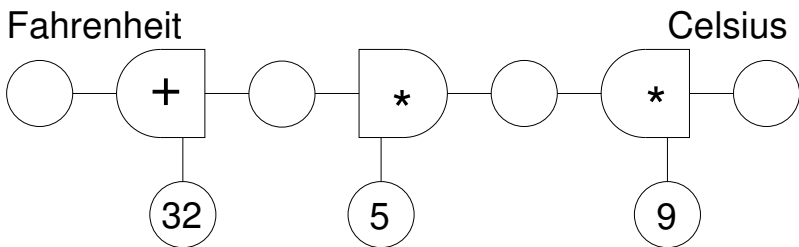
Win: Constraints compose into multidirectional computations



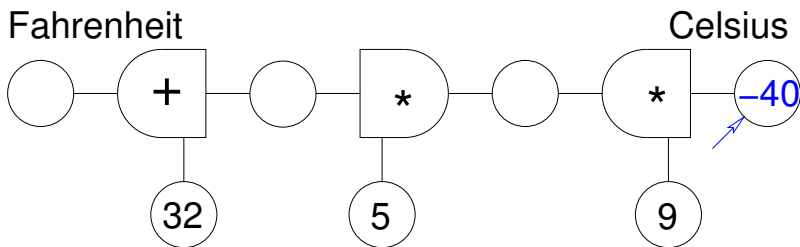
Win: Constraints compose into multidirectional computations



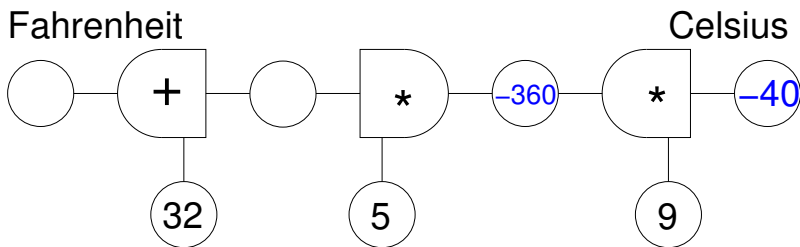
Win: Constraints compose into multidirectional computations



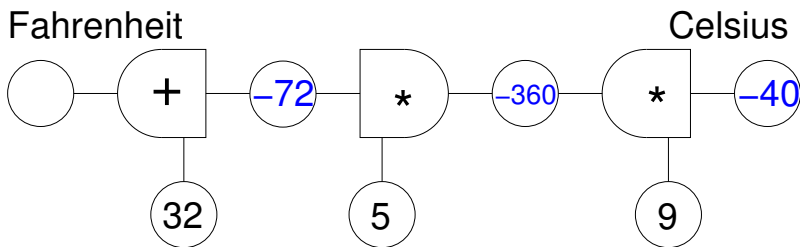
Win: Constraints compose into multidirectional computations



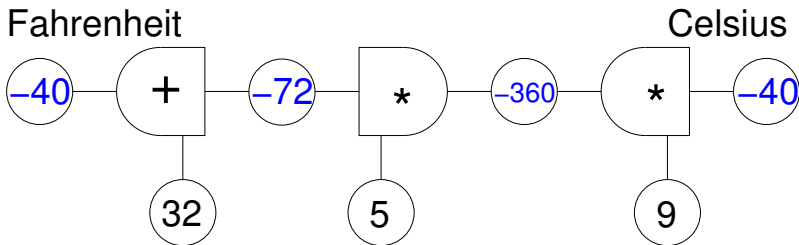
Win: Constraints compose into multidirectional computations



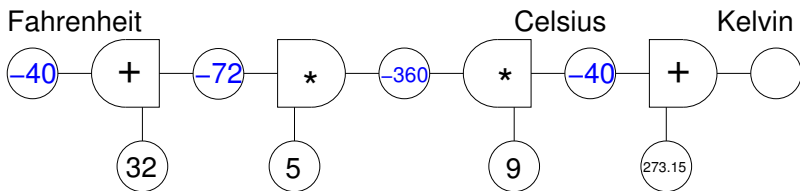
Win: Constraints compose into multidirectional computations



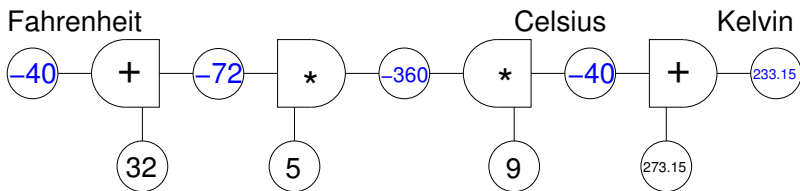
Win: Constraints compose into multidirectional computations



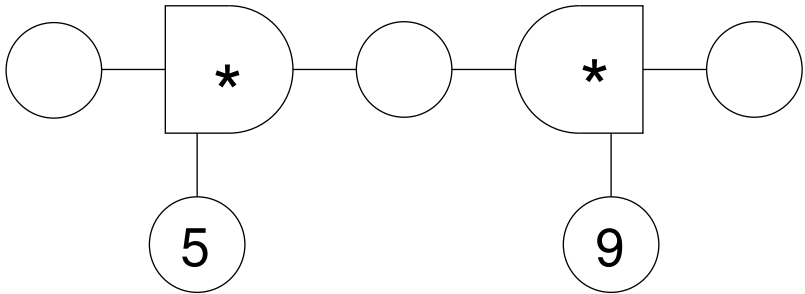
Win: Constraints compose into multidirectional computations



which can grow incrementally without adjusting explicit controls

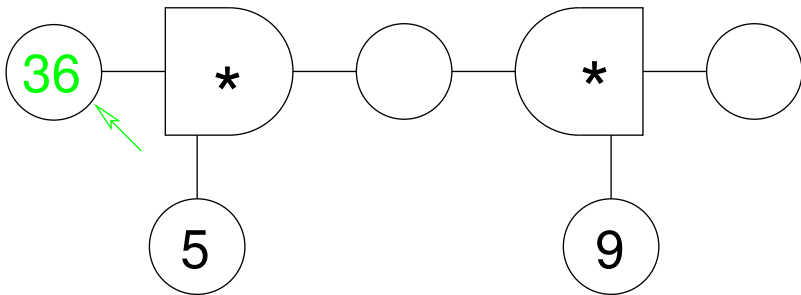


which can grow incrementally without adjusting explicit controls



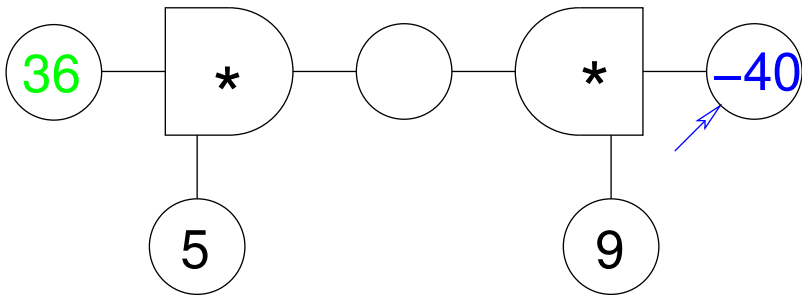
But: A cell can get stuff from
multiple sources

Is this bad?



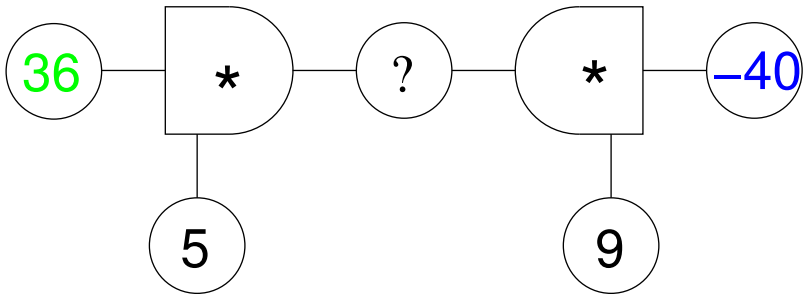
But: A cell can get stuff from multiple sources

Is this bad?



But: A cell can get stuff from multiple sources

Is this bad?



But: A cell can get stuff from multiple sources

Is this bad?

“Old View”: Cells hold values

“Old View”: Cells hold values

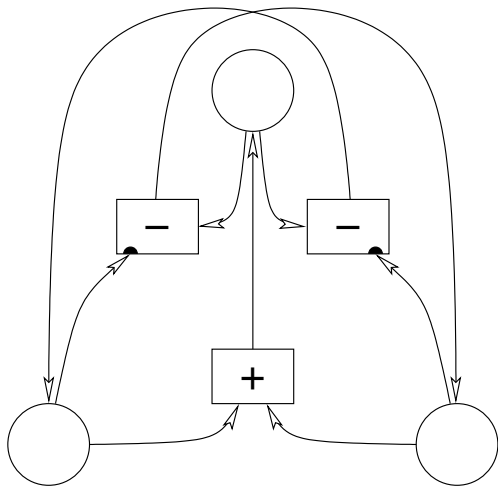
Leads to all kinds of trouble

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

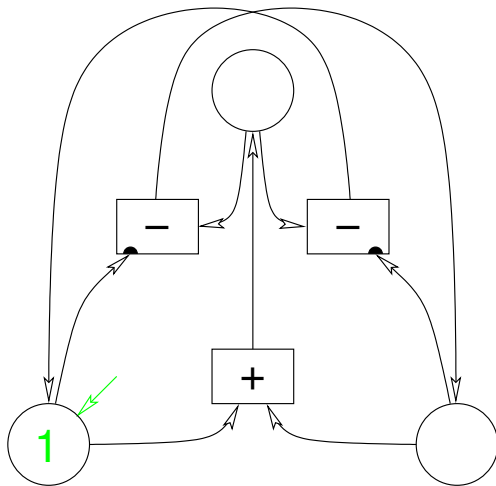
“Old View”: Cells hold values

Leads to all kinds of trouble

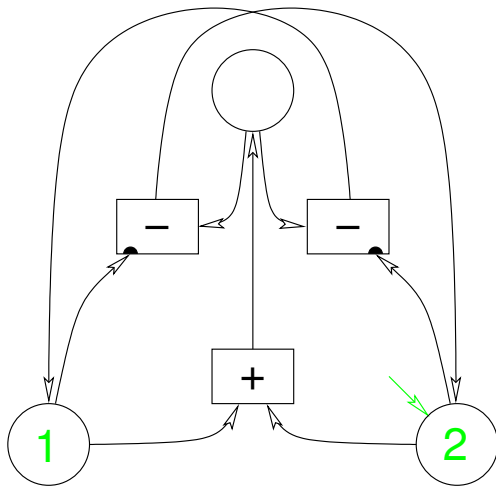
- ▶ precedence
- ▶ overwriting
- ▶ infinite reactions and fights
- ▶ ...
- ▶ loss of essence



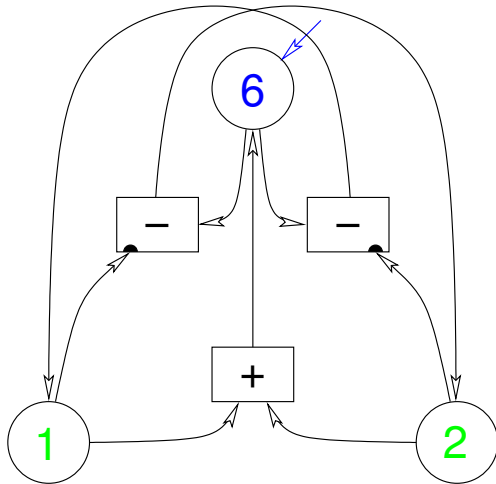
“Constraints” may stop constraining



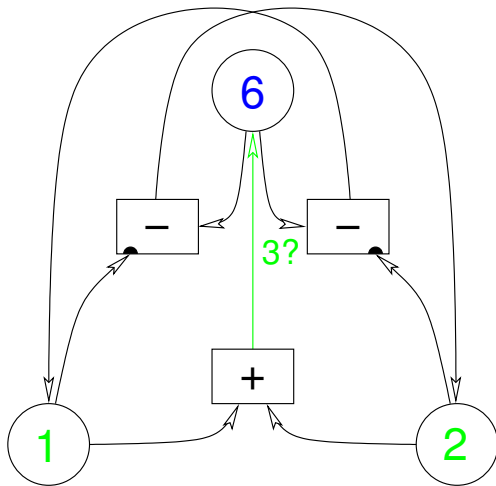
“Constraints” may stop constraining



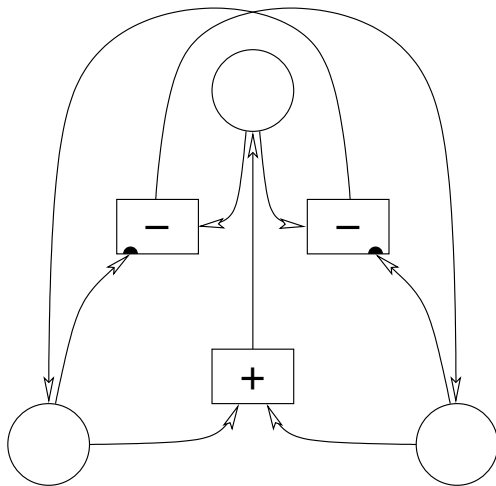
“Constraints” may stop constraining



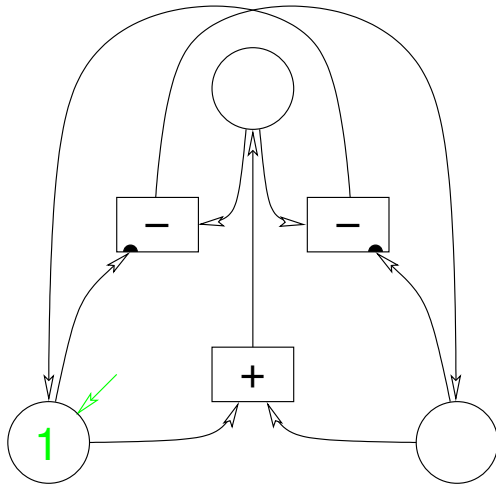
“Constraints” may stop constraining



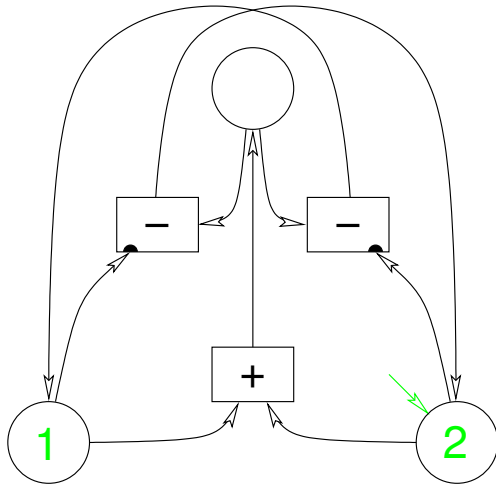
“Constraints” may stop constraining



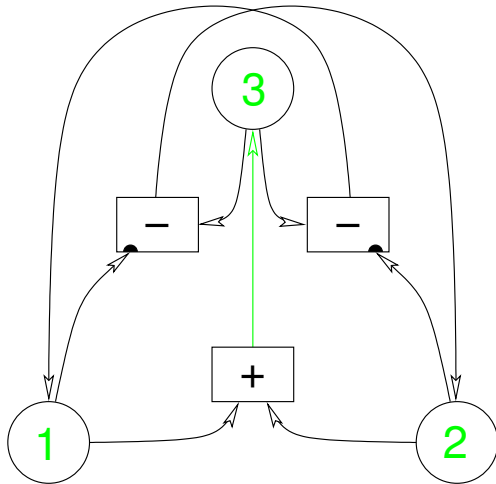
Loops may buzz forever



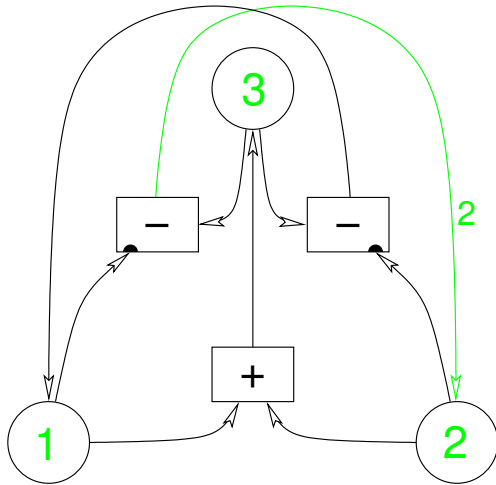
Loops may buzz forever



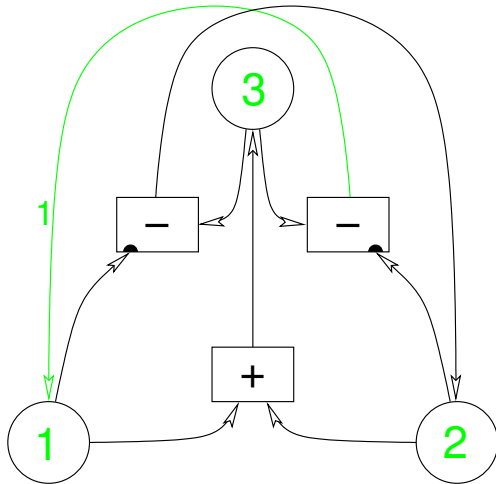
Loops may buzz forever



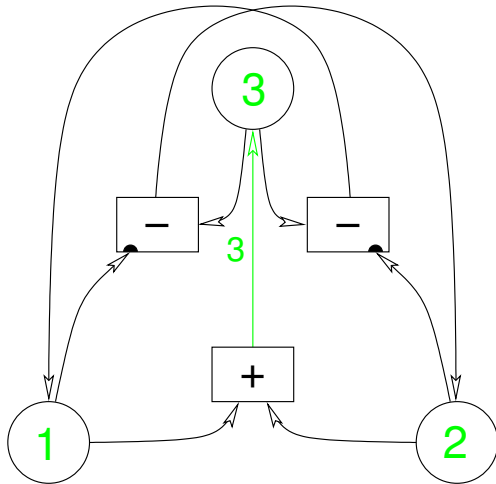
Loops may buzz forever



Loops may buzz forever

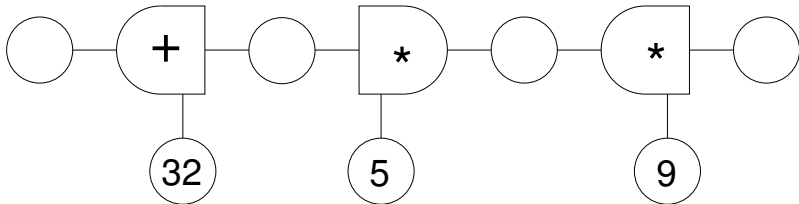


Loops may buzz forever



Loops may buzz forever

Fahrenheit

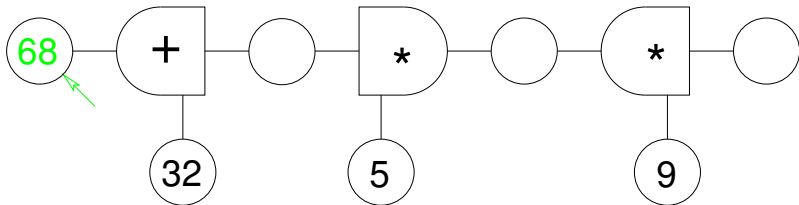


Celsius

Fights may break out

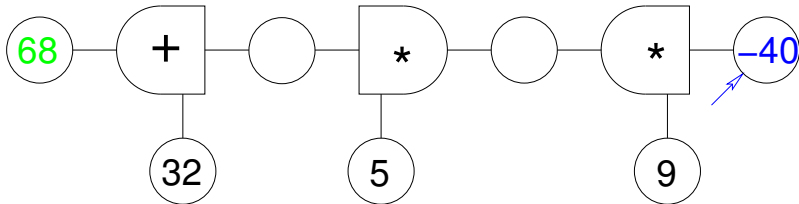
Fahrenheit

Celsius



Fights may break out

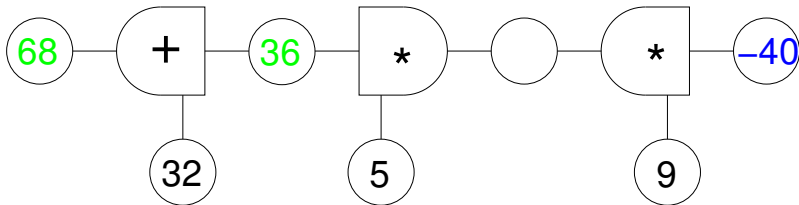
Fahrenheit



Celsius

Fights may break out

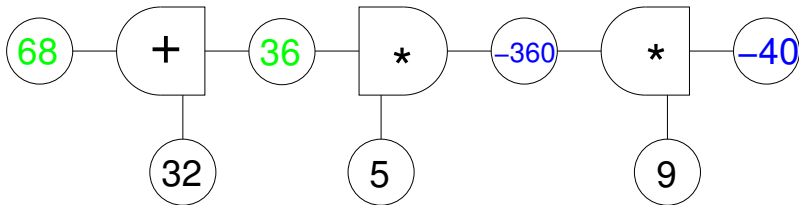
Fahrenheit



Celsius

Fights may break out

Fahrenheit

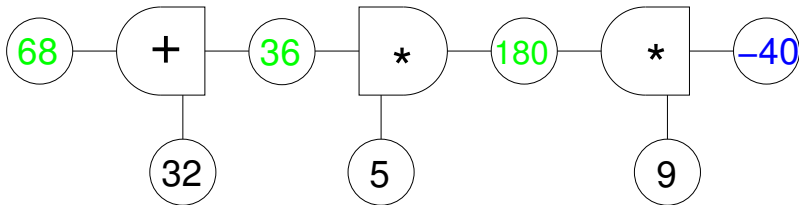


Celsius

Fights may break out

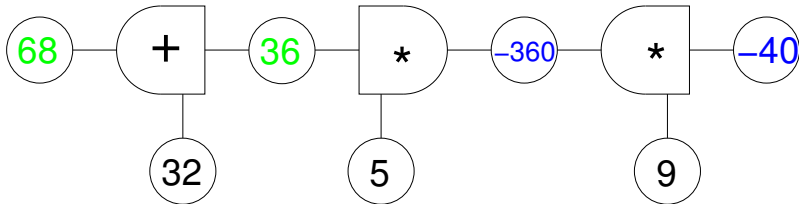
Fahrenheit

Celsius



Fights may break out

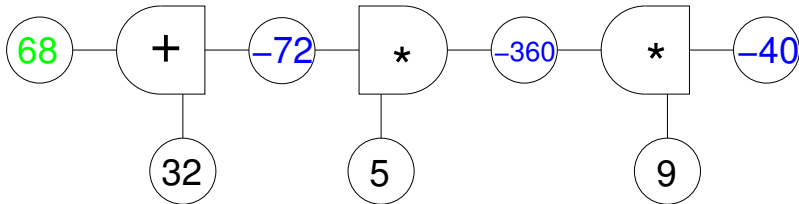
Fahrenheit



Celsius

Fights may break out

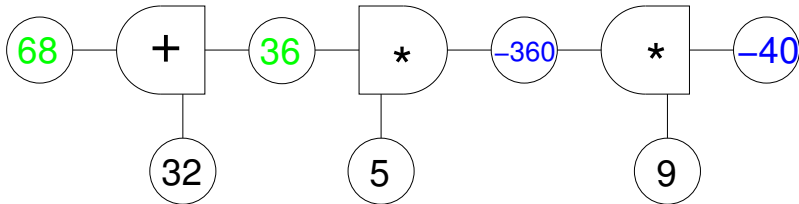
Fahrenheit



Celsius

Fights may break out

Fahrenheit



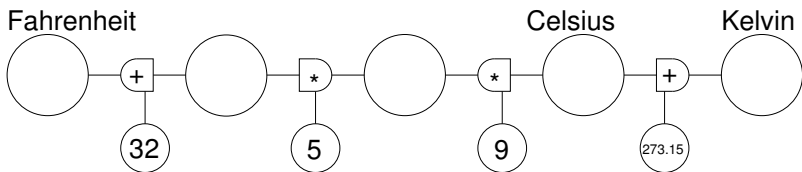
Celsius

Fights may break out

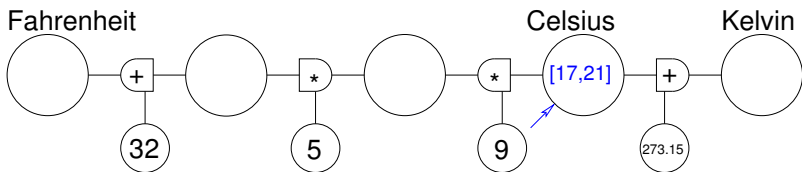
New View: Cells hold
information about values

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

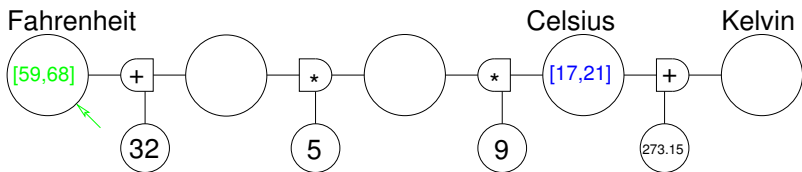
and merge it as it comes in from many sources



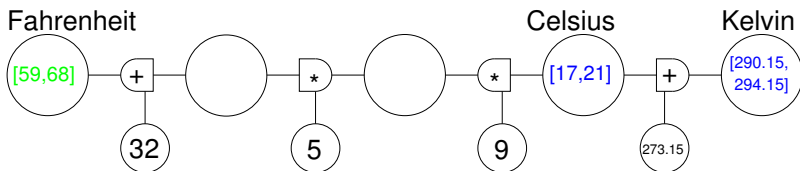
E.g. interval arithmetic is
partial information



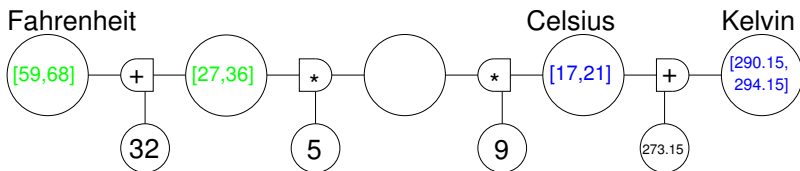
E.g. interval arithmetic is
partial information



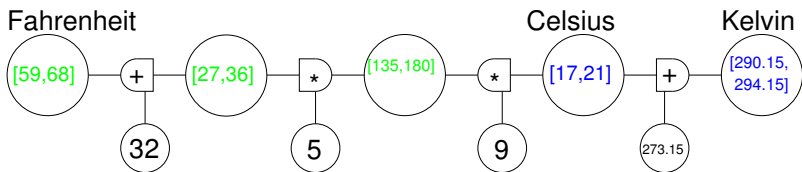
E.g. interval arithmetic is
partial information



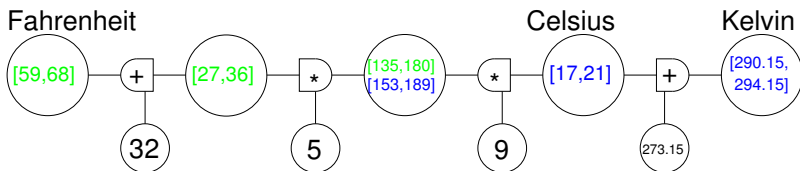
E.g. interval arithmetic is
partial information



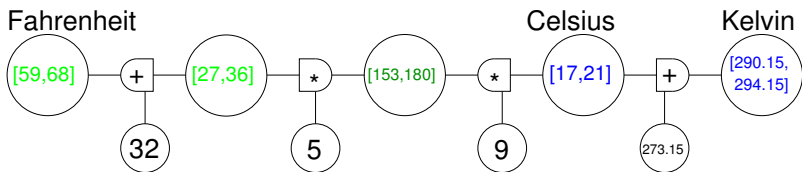
E.g. interval arithmetic is
partial information



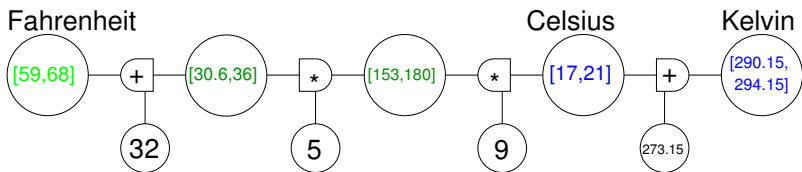
E.g. interval arithmetic is
partial information



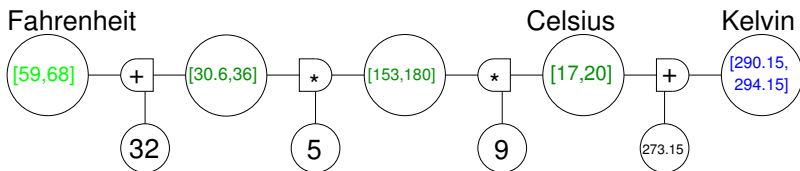
E.g. interval arithmetic is
partial information



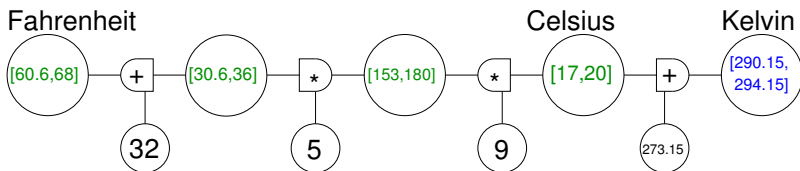
E.g. interval arithmetic is
partial information



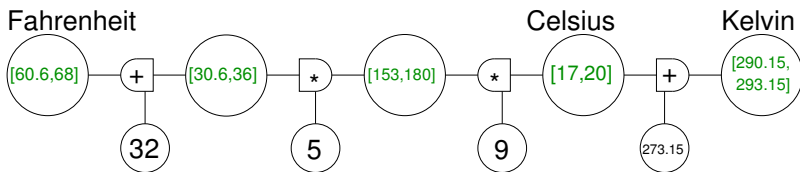
E.g. interval arithmetic is
partial information



E.g. interval arithmetic is
partial information

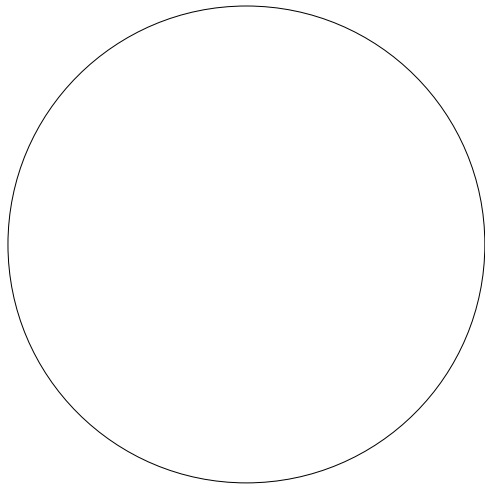


E.g. interval arithmetic is
partial information

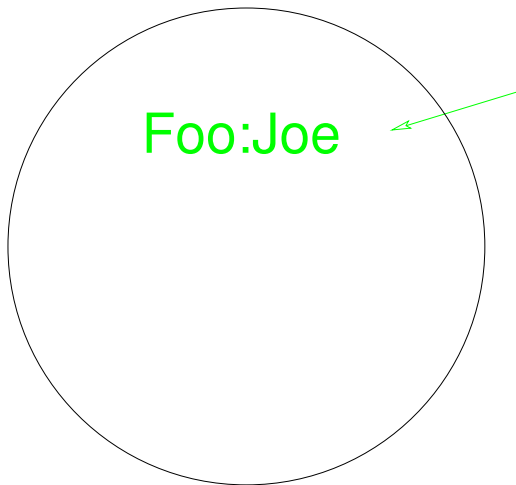


E.g. interval arithmetic is
partial information

Partial information makes no sense without multiple sources



Win: Truth maintenance is
partial information

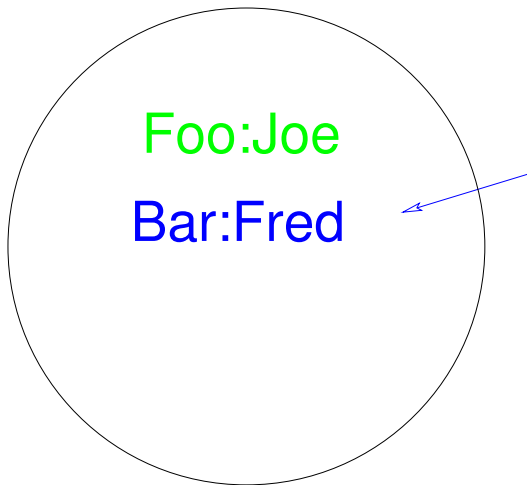


Win: Truth maintenance is
partial information



Foo:Joe

Win: Truth maintenance is
partial information

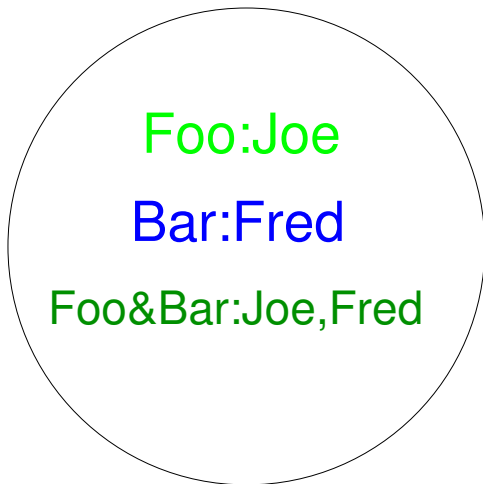


Win: Truth maintenance is
partial information



Foo:Joe
Bar:Fred

Win: Truth maintenance is
partial information



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**

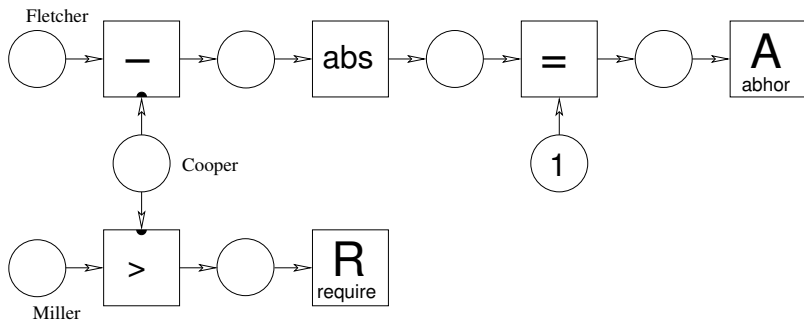
and now we can use many different kinds of accumulators

Example: Implicit Search

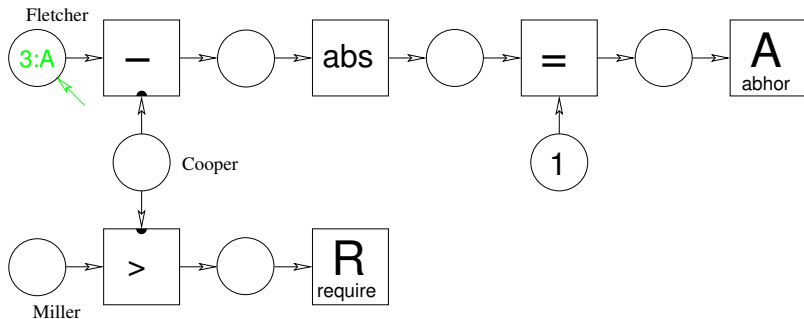
Baker, Cooper, Fletcher, Miller, and Smith live on the first five floors of this apartment house. Baker does not live on the fifth floor. Cooper does not live on the first floor. Fletcher does not live on either the fifth or the first floor. Miller lives on a higher floor than does Cooper. Smith does not live on a floor adjacent to Fletcher's. Fletcher does not live on a floor adjacent to Cooper's.

We can write a program like this

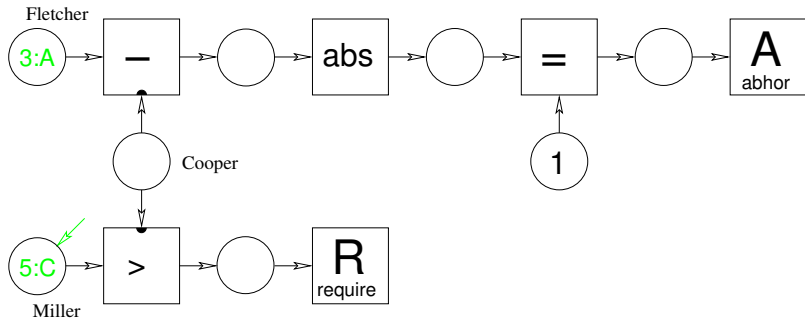
```
(define (multiple-dwelling)
  (let ((baker (one-of 1 2 3 4 5))
        (cooper (one-of 1 2 3 4 5))
        (fletcher (one-of 1 2 3 4 5))
        (miller (one-of 1 2 3 4 5))
        (smith (one-of 1 2 3 4 5)))
    (require-distinct
     (list baker cooper fletcher miller smith))
    (abhor (= baker 5))      (abhor (= cooper 1))
    (abhor (= fletcher 5))  (abhor (= fletcher 1))
    (require (> miller cooper))
    (abhor (= 1 (abs (- smith fletcher))))
    (abhor (= 1 (abs (- fletcher cooper))))
    (list baker cooper fletcher miller smith)))
```



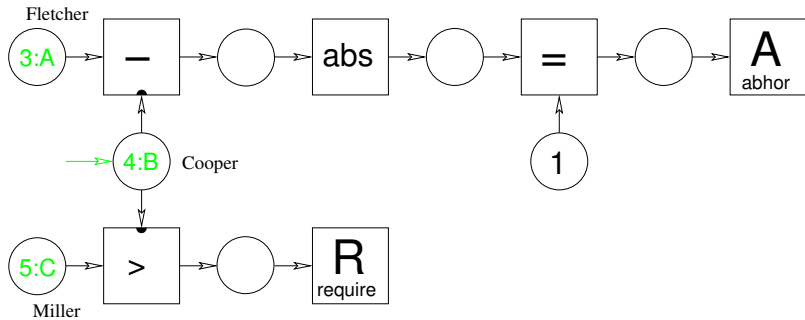
Compile it to a network like this



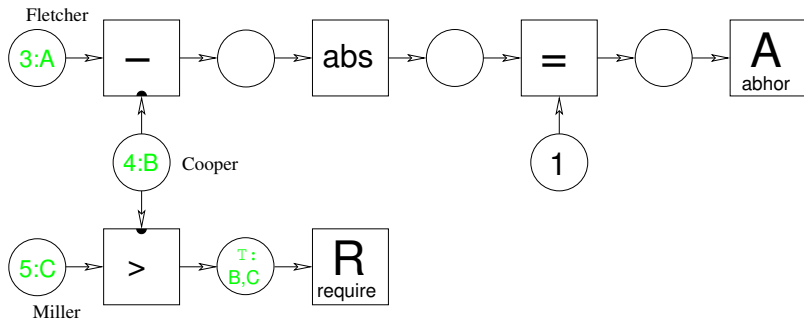
Track which guesses
a value depends on



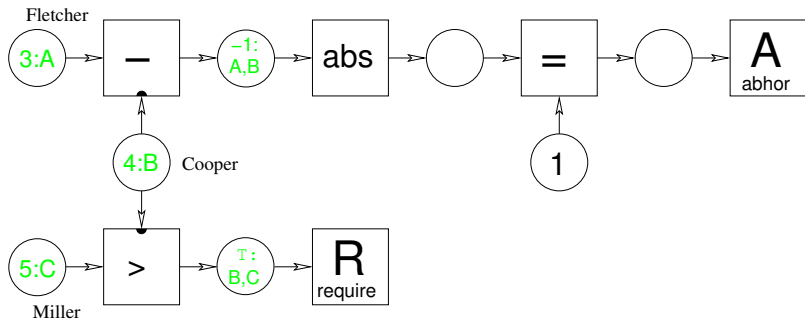
Track which guesses
a value depends on



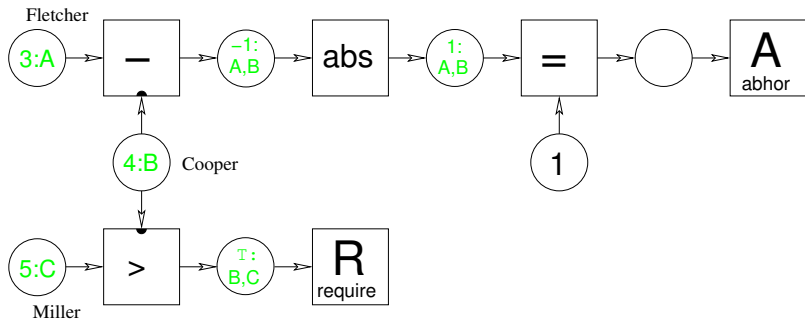
Track which guesses
a value depends on



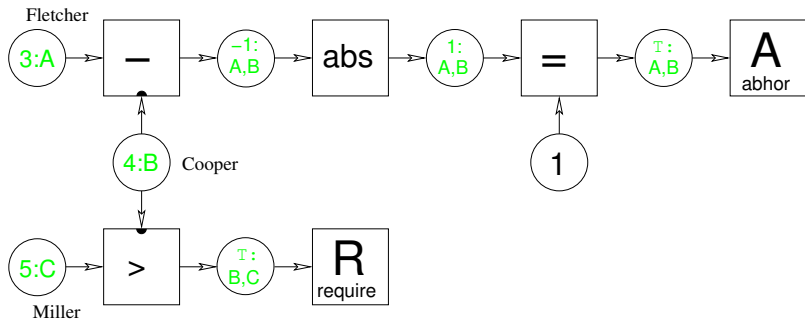
Track which guesses
a value depends on



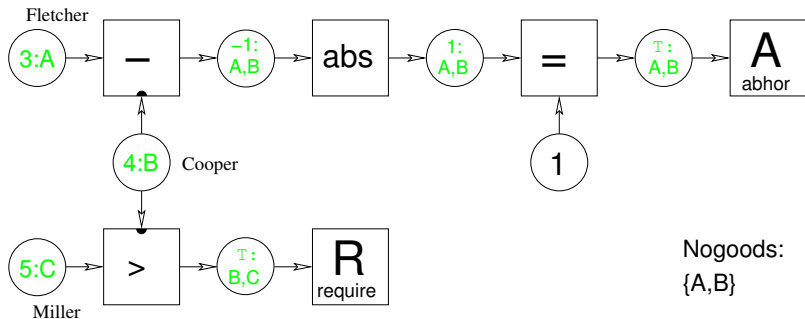
Track which guesses
a value depends on



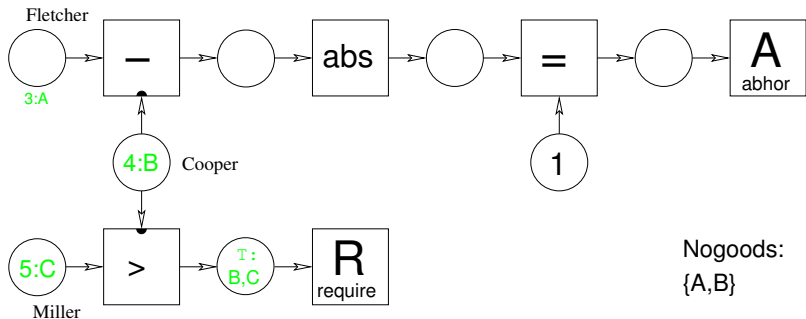
Track which guesses
a value depends on



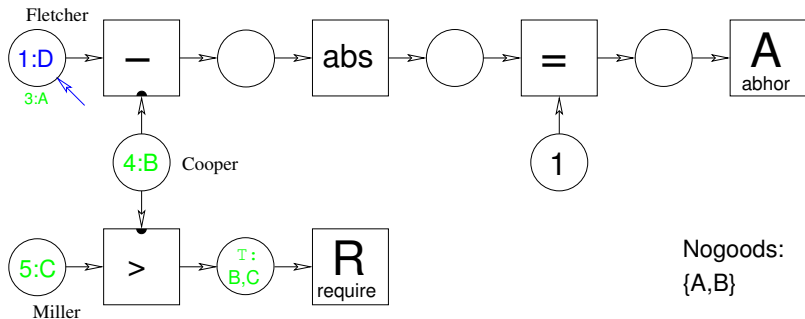
Track which guesses
a value depends on



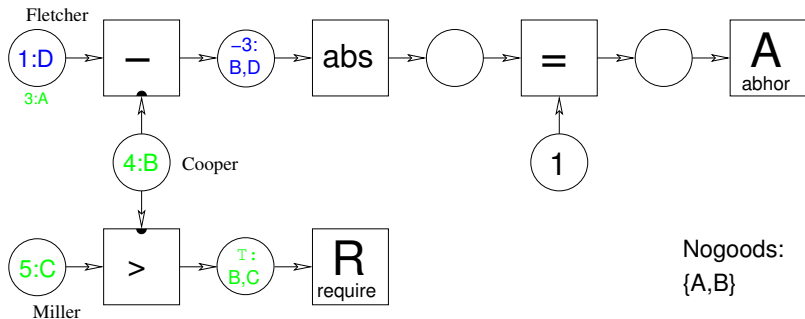
And use that for
accurate backtracking



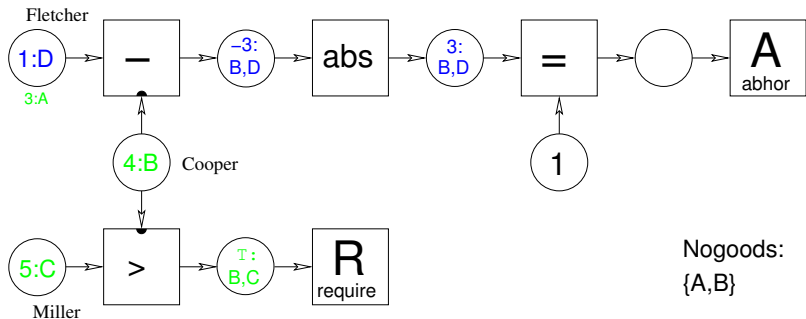
And use that for
accurate backtracking



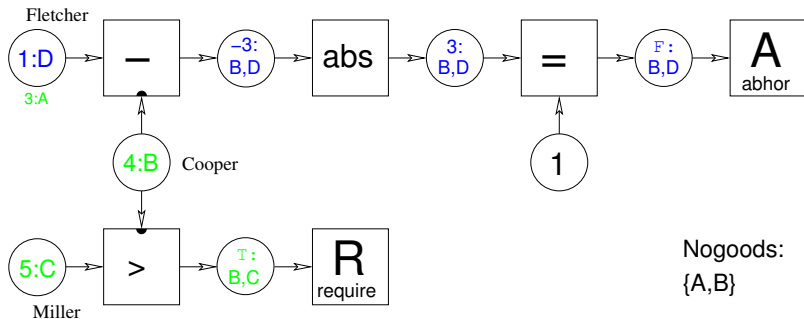
Without recomputing
anything unrelated



Without recomputing
anything unrelated



Without recomputing
anything unrelated



To solve that problem in
63 backtracks instead of 500

Pick the knowledge
representation for your own
problem, but

Partial information and
propagator networks are
essentially intertwined