Distributed protocols: a logical approach

SPLV 2026

Murdoch J. Gabbay

4 August 2026

Thanks

Thank you to the SPLV 2026 organisers for giving me the opportunity to deliver this course.

Thanks to Luca Zanolini and Isaac Sheff for teaching me about distributed protocols. Thanks to Jan Mas Rovira and Anthony Hart for formalising many of the proofs on which this talk is based.

Thank you for coming!

I welcome questions. I'm here for you: if you need clarification on some point, then just ask.

References



1. Declarative Bracha Broadcast (in press): https://arxiv.org/abs/2512.21137.
   Jan Mas Rovira's Lean formalisation of the DBB proofs: https://github.com/janmasrovira/mlda.



2. Declarative Paxos journal paper (submitted): https://arxiv.org/abs/2502.00892.



3. Semitopologies journal paper (published): https://doi.org/10.1093/logcom/exae050.



4. Semitopologies book (published): collegepublications.co.uk/logic/?00056.
5. DBB model-checker online: https://github.com/bellissimogiorno/declarative-protocols.
6. Three-valued logic tool at https://bellissimogiorno.github.io/threeval/.

Part 1

Message-passing protocols,
without message passing,
or protocols

Contents of this lecture: easy as 1-2-3

1. Recall what distributed protocols are and what makes their maths distinctive.
2. Consider a toy example using English pseudocode, show that it's fatally ambiguous, and use declarative methods to disambiguate.
3. Consider a slightly less toy example that requires a full modal logic (also: explain modal logic).

We will study Bracha Broadcast in Part 2. I might not finish Part 1 in the first lecture slot; and if so then that's OK.

What is a distributed protocol?

A parallel computation computes on multiple processes simultaneously, which may share memory or state.

A distributed protocol computes on multiple processes, which do not directly share state. They may communicate using messages.

A decentralised protocol is a distributed protocol in which there is no central coordination.

$$\text{Parallel} \nRightarrow \nLeftarrow \text{Distributed} \nRightarrow \nLeftarrow \text{Decentralised}.$$

These lectures will be about distributed / decentralised protocols.

Quick look at some examples (taxonomy debatable)

System/Protocol	Distrib.	Decent.	Reason
TCP/IP suite	Yes	Largely	Routing & endpoints cross independent networks. Addressing & governance partly hierarchical.
BGP	Yes	Largely	Autonomous systems set policy independently. No global router.
DNS	Yes	Partly	Resolution is distributed. Naming authority is hierarchical, rooted at DNS root zone.
Raft / Paxos	Yes	Usually not	Replicas agree on state; membership & authority usually remain in one administrative domain.
Bitcoin	Yes	Largely	Open participation & distributed consensus; mining & infrastructure may concentrate.
2-phase commit	Yes	No	One coordinator directs commit/abort; failure may block progress.
IPFS	Yes	Largely	Content addressing & peer routing need no central host. Gateways & pinning may concentrate.
Parallel reduction	Perhaps	Usually not	Processes combine partial results. Execution may span hosts but launch, membership & root centrally configured.
MapReduce	Yes	Usually not	Workers map & reduce. Manager schedules work, handles failure & permissions membership.

What makes distributed protocols mathematically distinctive?

In my view, what gives the theory of distributed protocols its distinctive character is an assumption of *unreliability*:

> *The network and/or some of the processes may be faulty.*

E.g. the centrality of faults is evident in a famous quote by Lamport, which he attributes to SRC (lamport.azurewebsites.net/pubs/distributed-system.txt):

> *A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*

Fault models vary: messages may be delayed or may fail to arrive; processes may crash (silently or otherwise) or behave arbitrarily.

In practice, and notwithstanding the official definitions, my perception of the mathematics is this:

- Multiple processes + fault model ⇒ distributed.
- Multiple processes + no fault model ⇒ parallel.

The challenge of distributed protocols is to design algorithms that pull useful and coherent behaviour out of unreliable independently operating messaging processes.

Distributed vs. p2p

For the record, I perceive a further split as follows:

1. If *fewer than half of the processes are faulty*, so *most of the processes are correct*, then we are in 'distributed-protocol land'.
2. If most processes may be faulty, we are in 'peer-to-peer land'.

These lectures will focus on 'distributed-protocol land'. I'm not personally committed to this, it's just the community I've mostly worked with, up to now.

A confession

I'm not a 'distributed protocols' guy. I'm a 'logic and declarative programming' guy who met some distributed-protocol researchers.

I was surprised to discover that they specify protocols using imperative pseudocode. I'm used to logic, formal spec, theorem-proving, type theory, and declarative programming.

So . . . I translated the field of distributed protocols into a language that I could understand: logic and declarative axioms. This turned out to be helpful.

This lecture course will give you the dubious benefit of my personal, unsanctioned, declarative take on how distributed protocols can (and IMHO should!) be specified.

(Should I have mentioned this to the SPLV organisers?)

Declarative distributed protocols

I have found it personally helpful to study distributed algorithms declaratively as *axiomatic theories*. It captures their logical structure in a new, high-level way.

These methods scale well. In collaboration, I have used declarative analysis to detect fatal errors in proposed industrial protocols, *and* to fix these errors. See
https://zenodo.org/records/17636313 HBB and
https://zenodo.org/records/21074359 Controllers.

I may base this lecture course on toy examples, for accessibility, but please make no mistake: these ideas are scalable and applicable. Indeed, the advantages of declarative reasoning *increase* with scale, because they tend to eliminate nonessential complexity and focus on the logical core.

Some pointers again:

- Declarative Bracha Broadcast journal paper: https://arxiv.org/abs/2512.21137.
- Declarative Paxos paper (submitted): https://arxiv.org/abs/2502.00892.
- Declarative model-checking tool (AI generated): https://github.com/bellissimogiorno/declarative-protocols.
- Lean formalisation of the DBB proofs: https://github.com/janmasrovira/mlda.
- Three-valued logic tool: https://bellissimogiorno.github.io/threeval/.





Imperative pseudocode / English considered harmful

Let's go right back to imperative pseudocode / English descriptions of protocols. Let me illustrate why it concerns me.

If you've studied formal specification then you may have some idea of what I'm about to do, but let's run through this anyway. Consider an apparently innocuous example.

I give you ...



. . . a simple protocol: one-player ping-pong

English specification:

1. A correct participant may *ping*.
2. If a correct participant *pings* then it must also *pong*.
3. (Failure assumption) The participant is either correct or faulty. A faulty participant may display arbitrary behaviour.

Correctness property: a correct participant will either do nothing, or ping and pong.

Proof: a paragraph of verbal reasoning on possible execution paths.

One-player ping-pong

English specification:

1. A correct participant may *ping*.
2. If a correct participant *pings* then it must also *pong*.
3. (Failure assumption) A faulty participant may display arbitrary behaviour.

Correctness property: a correct participant will either do nothing, or ping and pong.

Proof: a paragraph of verbal reasoning on possible execution paths.

Q: Spot the error.

One-player ping-pong

English specification:

1. A correct participant may *ping*.
2. If a correct participant *pings* then it must also *pong*.
3. (Failure assumption) A faulty participant may display arbitrary behaviour.

Correctness property: a correct participant will either do nothing, or ping and pong.

Proof: a paragraph of verbal reasoning on possible execution paths.

Q: Spot the error.

A: Can a participant just pong? Can a participant become faulty (e.g. crash) *after* a ping but *before* it can pong?

A simple corrected protocol

1. A correct participant may *ping*.
2. ~~If a correct participant *pings* then it must also *pong*.~~
   If a correct participant *pings* then it must also *pong*; otherwise it does nothing.
3. (Failure assumption) The participant is either always-correct or always-faulty. A faulty participant may display arbitrary behaviour.

So, even this one-line protocol concealed a multitude of ambiguities.
Be afraid: some real protocols contain more than one line!

Can we fix this?



A conceptual shift ('⇝' means 'abstracts to')

Protocol-world		Logic-world
Instructions	⇝	Axioms
Protocol P	⇝	Logical theory Thy_P
Run of a protocol	⇝	Model of Thy_P
Correctness property	⇝	Valid entailment $\mathrm{Thy}_P \models \phi$
Distributed protocol design	⇝	Logic and model theory

So instead of thinking about concrete runs of imperative protocols, we'll think about a logical abstraction of such, where a protocol abstracts to a set of axioms (= a logical theory), and a run of the protocol abstracts to a model of those axioms.

- $[\![\phi]\!]_p$ will mean 'the truth-value of ϕ at p'.
- $\mathrm{Thy}_P \models \phi$ means 'every model of Thy_P satisfies ϕ'.

If you aren't familiar with these terms, don't worry; I'll explain.
Or please just ask.

Modal logic

Our logics will be naturally modal. Modal logic is logic whose notion of validity exists *at some time and/or place*. The jargon for this is possible worlds.

In these lectures, the notion of possible world will just be the participants in the protocol: a predicate ϕ (syntax of ϕ defined later) takes a truth-value *at each participant* p.

An important point, which I will *not* have time to properly bring out in these talks, is an empirical observation that each protocol-family is characterised by its own distinctive notion of possible world:

- participant (Bracha Broadcast, Crusader Agreement),
- participant and round number (Paxos),
- participant and history (Heterogeneous Bracha Broadcast).

To some extent, it is possible to do protocol analysis on these families, just by studying structure of possible worlds. A more advanced course would study this more closely. No time! No time! So much to write and so little

Handling faults using three-valued logic

I said distributed protocols are made mathematically distinctive by the presence of faults. How do we handle this?

Introduce $\mathbf{3} = \{\mathbf{f}, \mathbf{b}, \mathbf{t}\}$, a three-valued domain of truth-values.

$$\mathbf{f} \leq \mathbf{b} \leq \mathbf{t}$$

Consider some predicate ϕ (syntax of which to be determined later):

1. A participant p is non-faulty or correct with respect to ϕ when $[\![\phi]\!]_p \in \{\mathbf{f}, \mathbf{t}\}$.
2. A participant is faulty when $[\![\phi]\!]_p = \mathbf{b}$.

$\mathbf{b}$ is the byzantine or faulty truth-value. $\mathbf{f}$ and $\mathbf{t}$ are the correct truth-values.

By definition above, a process is faulty with respect to ϕ when $[\![\phi]\!]_p = \mathbf{b}$, and p is correct with respect to ϕ when $[\![\phi]\!]_p \in \{\mathbf{t}, \mathbf{f}\}$.

Some connectives, for $tv, tv' \in 3$

Weak implies

$tv \to tv'$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{b}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$

Strong implies

$tv \rightharpoonup tv'$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{b}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$

Equiv

$tv \equiv tv'$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{t}$

Neg

$\neg tv$
$\mathbf{f}$
$\mathbf{b}$
$\mathbf{t}$

Validity judgement: Define $\vDash tv$ to mean $tv \in \{\mathbf{t}, \mathbf{b}\}$. We may call $\{\mathbf{t}, \mathbf{b}\}$ the valid truth-values.

- $\vDash tv \to tv'$ when $tv = \mathbf{t}$ implies $tv' \in \{\mathbf{t}, \mathbf{b}\}$.
- $\vDash tv \rightharpoonup tv'$ when $tv = \mathbf{t}$ implies $tv' = \mathbf{t}$.
- $\vDash tv \equiv tv'$ when $tv = tv'$.

See tool at
https://bellissimogiorno.github.io/threeval/.

Let's specify a one-player ping-pong tournament

Fix a set of participants Pnt and two atomic predicates $\mathtt{ping}$ and $\mathtt{pong}$.

An interpretation is a pair of functions $[\![\mathtt{ping}]\!], [\![\mathtt{pong}]\!] : \mathrm{Pnt} \to \mathbf{3}$. Intuitively, these functions describe the moves of each participant in the tournament.

It's a simple tournament: just some participants independently playing $\mathbf{3}$-valued one-player ping-pong.

An axiom $\mathtt{Ax}$ is a predicate, but calling it 'an axiom' signals that we intend this predicate to hold at every participant.

A logical theory Thy is a (possibly empty, possibly singleton) set of axioms. So a theory is a collection of predicates which we intend to hold everywhere.

A model of Thy is an interpretation in which every axiom in Thy holds at every participant.

Let's specify a one-player ping-pong tournament

I give you several candidate logical theories:

1. $\{\texttt{ping} \equiv \texttt{pong}\}$.
   Participants are correct or faulty. If correct, then ping implies pong, and no ping implies no pong.
2. $\{\texttt{ping} \rightharpoonup \texttt{pong}\}$.
   Participants are correct or faulty. Spontaneous pong permitted.
3. $\{\texttt{ping} \to \texttt{pong}\}$.
   Participants may crash. Spontaneous pong permitted.
4. $\{\texttt{ping} \to \texttt{pong},\ \texttt{pong} \rightharpoonup \texttt{ping}\}$.
   Participants may crash after ping. No ping, no pong.

It seems to me that all the above plausibly correspond to the pseudocode specification. All satisfy this correctness property:

$$\texttt{ping} \to \texttt{pong}.$$

Cf. $\texttt{ping} \rightharpoonup \texttt{pong}$ and $\texttt{ping} \equiv \texttt{pong}$.

A slightly less toy protocol

As before, fix a set of participants Pnt.

Toy is as follows:

1. *Propose.* Any participant p may broadcast a `propose` message to all participants.
2. *Accept.* If a participant receives a `propose` message, then it responds with `accept`.
3. *Decide.* If the participant p, having broadcast its `propose` message, receives an `accept` message, then it declares `decide`.

(I made this up; it's loosely based on a 1-value version of Paxos.)

Modalities

Our logic for formalising **Toy** will be explicitly modal, by which I mean that the axioms may contain modalities quantifying over some appropriate notion of *place and/or time*, which in modal jargon is a *possible world*.

For example:

"It's raining" is not explicitly modal.

"It's raining *somewhere*" is modal: typically written $\Diamond\texttt{raining}$.

"It's raining *everywhere*" is modal: typically written $\Box\texttt{raining}$.

For **Toy**, possible worlds are participants $p \in \mathrm{Pnt}$ (no notion of time required).

More formally: the truth-value $[\![\phi]\!]_w$ is a function of ϕ and w where w ranges over a suitable set of possible worlds.

Write $w \vDash \phi$ for '$[\![\phi]\!]_w \in \{\mathbf{b}, \mathbf{t}\}$', read '$[\![\phi]\!]_w$ is valid'.

Background 2. Three-valued (modal) logic

Recall the three truth-values:

1. $\mathbf{t}$ ('true'). When $[\![\phi]\!]_w = \mathbf{t}$, call ϕ *true* at w.
2. $\mathbf{f}$ ('false'). When $[\![\phi]\!]_w = \mathbf{f}$, call ϕ *false* at w.
3. $\mathbf{b}$ ('byzantine'). When $[\![\phi]\!]_w = \mathbf{b}$, call ϕ *byzantine* at w.

Call $x \in \{\mathbf{t}\}$ true, $x \in \{\mathbf{t}, \mathbf{f}\}$ correct, and $x \in \{\mathbf{t}, \mathbf{b}\}$ valid.

For x a truth-value, define modalities $\mathtt{T}$, $\mathbb{F}$, and $\mathbb{B}$ by:

- $\mathtt{T}\, x = \mathbf{t}$ when $x \in \{\mathbf{t}\}$ and $\mathtt{T}\, x = \mathbf{f}$ otherwise;
- $\mathbb{F}\, x = \mathbf{t}$ when $x \in \{\mathbf{t}, \mathbf{f}\}$ and $\mathbb{F}\, x = \mathbf{f}$ otherwise; and
- $\mathbb{B}\, x = \mathbf{t}$ when $x \in \{\mathbf{t}, \mathbf{b}\}$ and $\mathbb{B}\, x = \mathbf{f}$ otherwise.

Write $w \vDash \phi$ to mean $[\![\phi]\!]_w \in \{\mathbf{t}, \mathbf{b}\}$. Write $\vDash \phi$ to mean $\forall w.(w \vDash \phi)$.

Some connectives, for reference

Weak implication

$tv \to tv'$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{b}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$

Strong implication

$tv \rightharpoonup tv'$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{t}$	$\mathbf{b}$	$\mathbf{b}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$

$\neg tv$	
$\mathbf{t}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{b}$
$\mathbf{f}$	$\mathbf{t}$

$\mathbb{T}\,tv$	
$\mathbf{t}$	$\mathbf{t}$
$\mathbf{b}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{f}$

$\mathbb{B}\,tv$	
$\mathbf{t}$	$\mathbf{f}$
$\mathbf{b}$	$\mathbf{t}$
$\mathbf{f}$	$\mathbf{f}$

$\mathbb{TB}\,tv$	
$\mathbf{t}$	$\mathbf{t}$
$\mathbf{b}$	$\mathbf{t}$
$\mathbf{f}$	$\mathbf{f}$

$\mathbb{TF}\,tv$	
$\mathbf{t}$	$\mathbf{t}$
$\mathbf{b}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{t}$

Intuitively:

- $\models tv \rightharpoonup tv'$ means $tv = \mathbf{t}$ implies $tv' = \mathbf{t}$.
- $\models tv \to tv'$ means $tv = \mathbf{t}$ implies $tv' \in \{\mathbf{t}, \mathbf{b}\}$.

Recall Toy

We assume participants $p \in \mathsf{Pnt}$.

1. *Propose.* Any participant p may broadcast a `propose` message to all participants.
2. *Accept.* If a participant receives a `propose` message, then it responds with `accept`.
3. *Decide.* If the participant p, having broadcast its `propose` message, receives an `accept` message, then it declares `decide`.

Let's make this declarative. We propose *four* theories (= set of axioms), corresponding to different failure assumptions.

Theory 1. Honest & live

Assume atomic propositions `propose`, `accept`, and `decide`.

Backward rules	
(**SimpAccept**?)	$\texttt{accept} \rightharpoonup \Diamond\texttt{propose}$
(**SimpDecide**?)	$\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond\texttt{accept})$
Forward rules	
(**SimpAccept**!)	$(\Diamond\texttt{propose}) \rightharpoonup \texttt{accept}$
(**SimpDecide**!)	$(\texttt{propose} \wedge \Diamond\texttt{accept}) \rightharpoonup \texttt{decide}$
Other rules	
(**Correct**)	$\mathbb{TF}[\texttt{propose}, \texttt{accept}, \texttt{decide}]$

$p \vDash \phi \rightharpoonup \phi'$ means $p \vDash \mathtt{T}\,\phi \Rightarrow p \vDash \mathtt{T}\,\phi'$
(if ϕ is true then ϕ' is true; material implication).

$p \vDash \Diamond\phi$ means $\exists p'.p' \vDash \phi$. ($\phi$ is valid somewhere).

A model (= assignment of truth-values to atomic propositions) satisfies a theory $\mathcal{T}$ when $\vDash (\mathbf{Ax})$ in that model, for every $(\mathbf{Ax}) \in \mathcal{T}$.

Theory 2. Participants may crash

Backward rules

(**SimpAccept**?) $\texttt{accept} \rightharpoonup \Diamond\texttt{propose}$

(**SimpDecide**?) $\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond\texttt{accept})$

Forward rules

(**SimpAccept**!) $(\Diamond\texttt{propose}) \rightarrow \texttt{accept}$

(**SimpDecide**!) $(\texttt{propose} \wedge \Diamond\texttt{accept}) \rightarrow \texttt{decide}$

$p \vDash \phi \rightharpoonup \phi'$ means $p \vDash \mathtt{T}\,\phi \Rightarrow p \vDash \mathtt{T}\,\phi'$

$p \vDash \phi \rightarrow \phi'$ means $p \vDash \mathtt{T}\,\phi \Rightarrow p \vDash \mathbb{TB}\,\phi'$

$p \vDash \Diamond\phi$ means $\exists p'.p' \vDash \phi$.

Theory 3. Honest & live; messages may fail to arrive

Backward rules

(**SimpAccept**?) $\texttt{accept} \rightharpoonup \Diamond\texttt{propose}$

(**SimpDecide**?) $\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond\texttt{accept})$

Forward rules

Other rules

(**Correct**) $\mathbb{TF}[\texttt{propose}, \texttt{accept}, \texttt{decide}]$

$p \vDash \phi \rightharpoonup \phi'$ means $p \vDash \mathsf{T}\phi \Rightarrow p \vDash \mathsf{T}\phi'$

$p \vDash \phi \rightarrow \phi'$ means $p \vDash \mathsf{T}\phi \Rightarrow p \vDash \mathbb{TB}\phi'$

$p \vDash \Diamond\phi$ means $\exists p'.p' \vDash \phi$.

Theory 4. Some participants may be byzantine

Backward rules

(**SimpAccept**?) $\texttt{accept} \rightarrow \Diamond\texttt{propose}$

(**SimpDecide**?) $\texttt{decide} \rightarrow (\texttt{propose} \wedge \Diamond\texttt{accept})$

Forward rules

(**SimpAccept**!) $(\Diamond\texttt{propose}) \rightarrow \texttt{accept}$

(**SimpDecide**!) $(\texttt{propose} \wedge \Diamond\texttt{accept}) \rightarrow \texttt{decide}$

$p \vDash \phi \rightharpoonup \phi'$ means $p \vDash \mathtt{T}\,\phi \Rightarrow p \vDash \mathtt{T}\,\phi'$

$p \vDash \phi \rightarrow \phi'$ means $p \vDash \mathtt{T}\,\phi \Rightarrow p \vDash \mathbb{TB}\,\phi'$

$p \vDash \Diamond\phi$ means $\exists p'.p' \vDash \phi$.

1.

$\texttt{accept} \rightharpoonup \Diamond \texttt{propose}$

$\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond \texttt{accept})$

$(\Diamond \texttt{propose}) \rightharpoonup \texttt{accept}$

$(\texttt{propose} \wedge \Diamond \texttt{accept}) \rightharpoonup \texttt{decide}$

$\mathbb{TF}\,[\texttt{propose}, \texttt{accept}, \texttt{decide}]$

2.

$\texttt{accept} \rightharpoonup \Diamond \texttt{propose}$

$\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond \texttt{accept})$

$(\Diamond \texttt{propose}) \rightarrow \texttt{accept}$

$(\texttt{propose} \wedge \Diamond \texttt{accept}) \rightarrow \texttt{decide}$

3.

$\texttt{accept} \rightharpoonup \Diamond \texttt{propose}$

$\texttt{decide} \rightharpoonup (\texttt{propose} \wedge \Diamond \texttt{accept})$

$\mathbb{TF}\,[\texttt{propose}, \texttt{accept}, \texttt{decide}]$

4.

$\texttt{accept} \rightarrow \Diamond \texttt{propose}$

$\texttt{decide} \rightarrow (\texttt{propose} \wedge \Diamond \texttt{accept})$

$(\Diamond \texttt{propose}) \rightarrow \texttt{accept}$

$(\texttt{propose} \wedge \Diamond \texttt{accept}) \rightarrow \texttt{decide}$

What's really happening here?

Declarative programming is happening. Consider:

```
# Python
def sum_sq_even(xs):
    total = 0
    for x in xs:
        if x % 2 == 0:
            total += x * x
    return total

-- Haskell
sumSqEven xs =
  sum [x*x | x <- xs, even x]
```

```
# Python
def reachable(g, start):
    seen = {start}
    todo = [start]
    while todo:
        x = todo.pop()
        for y in g.get(x, ()):
            if y not in seen:
                seen.add(y)
                todo.append(y)
    return seen

% Datalog
reach(X,X) :- node(X).
reach(X,Z) :- reach(X,Y), edge(Y,Z).
```

```
# Python
def balanced(s):
    depth = 0
    for c in s:
        if c == '(':
            depth += 1
        elif c == ')':
            depth -= 1
            if depth < 0:
                return False
        else:
            return False
    return depth == 0

% Prolog
balanced --> [].
balanced --> ['('], balanced, [')'], balanced.
```

(Q. Is the Python balanced bracketing above, correct?)

```python
# Python
def balanced(s):
    depth = 0
    for c in s:
        if c == '(':
            depth += 1
        elif c == ')':
            depth -= 1
            if depth < 0:
                return False
        else:
            return False
    return depth == 0
```

```
% Prolog
balanced --> [].
balanced --> ['('], balanced, [')'], balanced.
```

(Q. Is the Python balanced bracketing above, correct?)

(A. Wrong question!)

Declarative method: from algorithmic to logical time

There is no explicit message-passing, thus no model of message-passing order. Message-passing protocols, without message-passing! Whoa!

The axiomatic theories exist in what I like to call logical time; the inherent logical structure of the protocol is brought to the fore.

We elide what I like to call algorithmic time; that is, precisely how logical time is embedded into physical reality.

I propose that *logical* time is the true home of distributed protocols (cf. imperative vs declarative programs). Algorithmic time is a(n important) detail of implementation, like bytecode.

It's not all about abstraction though; it's also in my view about precision. E.g. the failure model is made symbolically explicit.

We end up with precise logical artefacts that are amenable to high-level logical analysis. Next up: Bracha Broadcast.

Part 2

Bracha Broadcast
as a logical theory

Semitopologies, quorums, coquorums, & contraquorums

A semitopology consists of a pair (Pnt, Opn) of

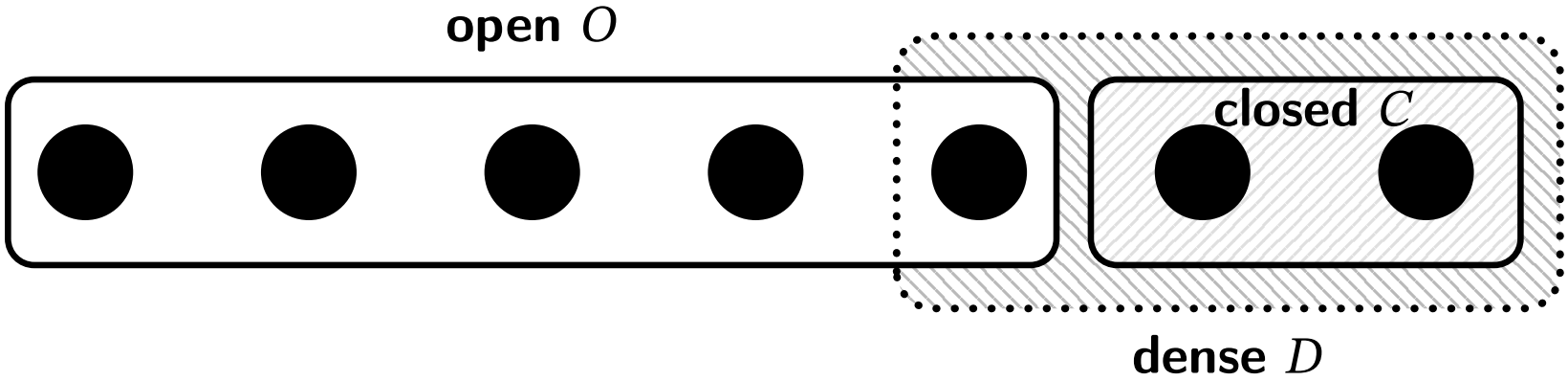
- a set of points Pnt, and
- a set of open sets Opn $\subseteq pow(\text{Pnt})$ that is closed under arbitrary union and contains $\emptyset$ and Pnt.

This is like a topology, but without insisting that pairwise intersection of opens must be open (though it might be). In topology, the intuition of an open set is a *positive affirmation*. In semitopology, the intuition is of an *actionable coalition*; a set of participants with the capacity, though not the obligation, to act together to perform some action.

1. Every topology is trivially a semitopology.
2. Fix some $f \geq 0$ and let Pnt be a finite set with $\#\text{Pnt} = 3f + 1$. For $O \subseteq \text{Pnt}$, set $O \in \text{Opn}$ when O is empty or $\#O \geq 2f + 1$. Call this a threshold semitopology.

A threshold semitopology

We set $f = 2$ and $\#\mathrm{Pnt} = 3f + 1 = 7$.



In this case:

- O is a quorum (nonempty open) when $\#O \geq 2f + 1 = 5$.
- C is a coquorum (nontotal closed) when $\#C \leq f = 2$.
- D is dense / a contraquorum / a blocking-set when $\#D \geq f + 1 = 3$.

Lemma: any three nonempty open sets intersect in at least one point; equivalently, the intersection of any *two* nonempty open sets is *dense*.

Semitopologies, quorums, coquorums, & contraquorums

Given a semitopology $(\mathrm{Pnt}, \mathrm{Opn})$:

- Call $O \in \mathrm{Opn}$ an open and $O \in \mathrm{Opn}_{\neq\emptyset}$ a nonempty open.
- Call $C \subseteq \mathrm{Pnt}$ closed when $C = \mathrm{Pnt} \setminus O$ for some $O \in \mathrm{Opn}$.
- We may call a nonempty open a quorum, and a nontotal closed set a coquorum.
- Call $D \subseteq \mathrm{Pnt}$ dense or a contraquorum when $D \cap O \neq \emptyset$ for every nonempty $O \in \mathrm{Opn}$ (also called a *blocking set*).
  This is exactly the notion of *dense set* from topology.

In the case of a threshold semitopology where $\#\mathrm{Pnt} = 3f + 1$, and $O \in \mathrm{Opn}$ when O is empty or $\#O \geq 2f + 1$,

- $C \subseteq \mathrm{Pnt}$ is closed precisely when $C = \mathrm{Pnt}$ or $\#C \leq f$, and
- $D \subseteq \mathrm{Pnt}$ is dense precisely when $\#D \geq f + 1$.

Bracha Broadcast (high-level view)

For correct processes:

1. A designated sender process *broadcasts* v to all processes. We assume all messages arrive.
2. If a process receives a broadcast v message from the sender — we assume authenticated point-to-point channels — it sends an *echo* v to all processes.
   Each process will echo at most *once*, so if it receives two broadcast messages with different values from a (byzantine) sender, it will only echo one of them.
3. If a process receives a quorum of echo messages for a value v, or a contraquorum of ready messages for a value v, then it sends messages to all processes declaring itself *ready* with v.
4. If a process receives a quorum of ready messages for a value v, then the process *delivers* v.

For faulty processes: Arbitrary behaviour.

Time to build a 3-valued modal logic over semitopologies

- Fix unary predicate-symbols `dlvr`, `ready`, `echo`, and `brdcst`. Let $\mathtt{P}$ range over these.
- Fix a semitopology (Pnt, Opn). Recall that $\mathbf{3} = \{\mathbf{f}, \mathbf{b}, \mathbf{t}\}$.
- Fix a nonempty set of values Val.
- Let predicates ϕ be generated by the following grammar:

$$\begin{aligned}\phi ::= {} & \mathbf{f} \mid \mathbf{b} \mid \mathbf{t} \mid \mathtt{P}(v) \mid \phi \vee \phi \mid \neg\phi \mid \phi \rightharpoonup \phi \mid \phi \to \phi \mid \\ & \Box\phi \mid \Diamond\phi \mid \boxdot\phi \mid \diamondsuit\phi \mid \\ & v = v' \mid \mathbb{F}\,\phi \mid \mathbb{B}\,\phi \mid \mathtt{B}\,\phi \mid \exists_{\mathbf{01}} v.\phi \mid \exists_{\mathbf{1}} v.\phi\end{aligned}$$

- An interpretation consists of an assignment to each predicate-symbol $\mathtt{P}$ of a function $[\![\mathtt{P}]\!] : (\mathrm{Pnt} \times \mathrm{Val}) \to \mathbf{3}$.

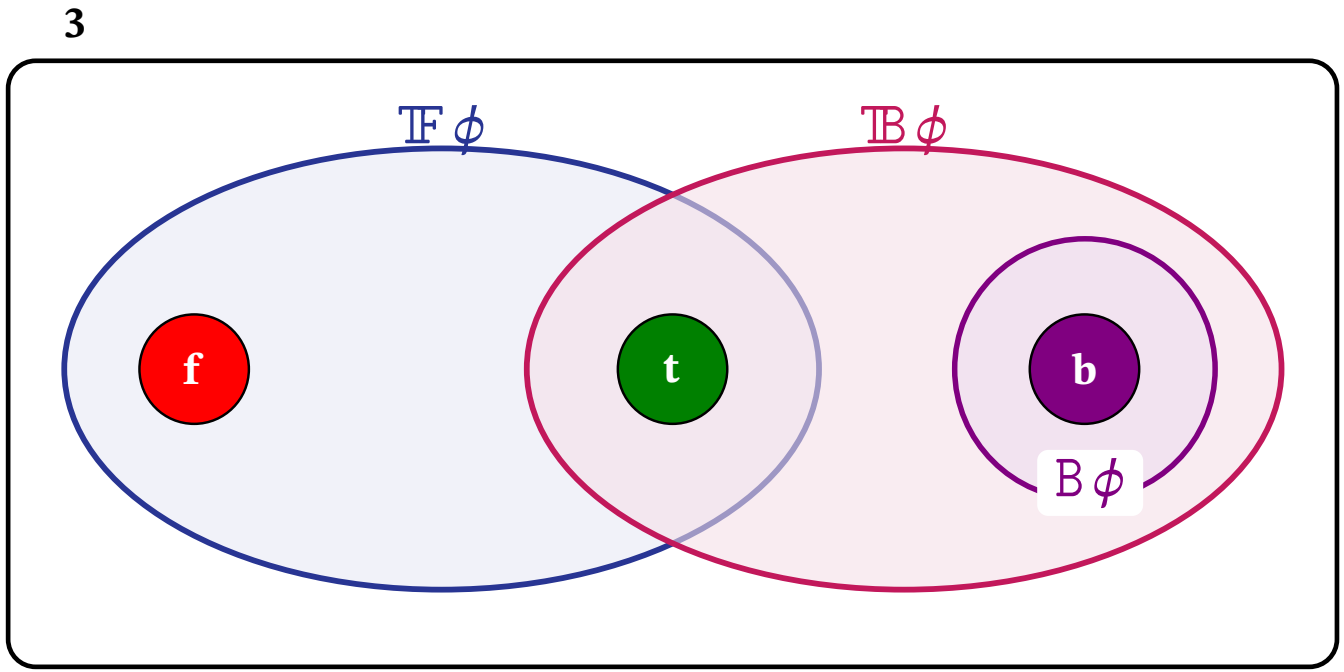
A formal semantics will follow. Here are some intuitions:

3-valued modal logic over semitopologies, cont.

- $\mathrm{P}(v)$ means informally that $\mathrm{P}(v)$ is valid *here* (at a participant). More formally, it means $[\![\mathrm{P}]\!](p, v) \in \{\mathbf{t}, \mathbf{b}\}$.
- $\Diamond\phi$ means 'ϕ valid at some participant' and $\Box\phi$ means 'ϕ valid at every participant'.
- $\boxdot\phi$ means 'ϕ valid on a quorum of participants' and $\diamondsuit\phi$ means 'ϕ valid on a contraquorum / dense set of participants'.
- $v = v'$ means 'v equals v'' (returns $\mathbf{t}$ or $\mathbf{f}$; never $\mathbf{b}$).
- $\mathbb{F}\,\phi$ means 'ϕ is correct, i.e. not the byzantine truth-value $\mathbf{b}$'. $\mathbb{B}\,\phi$ means 'ϕ is valid, i.e. not the false truth-value $\mathbf{f}$'. $\mathrm{B}\,\phi$ means 'ϕ is byzantine'.
- $\exists_{\mathbf{01}} v.\phi$ and $\exists_{\mathbf{1}} v.\phi$ mean 'ϕ holds of at most one / one value'. The truth-tables for this are non-evident (see later slide).

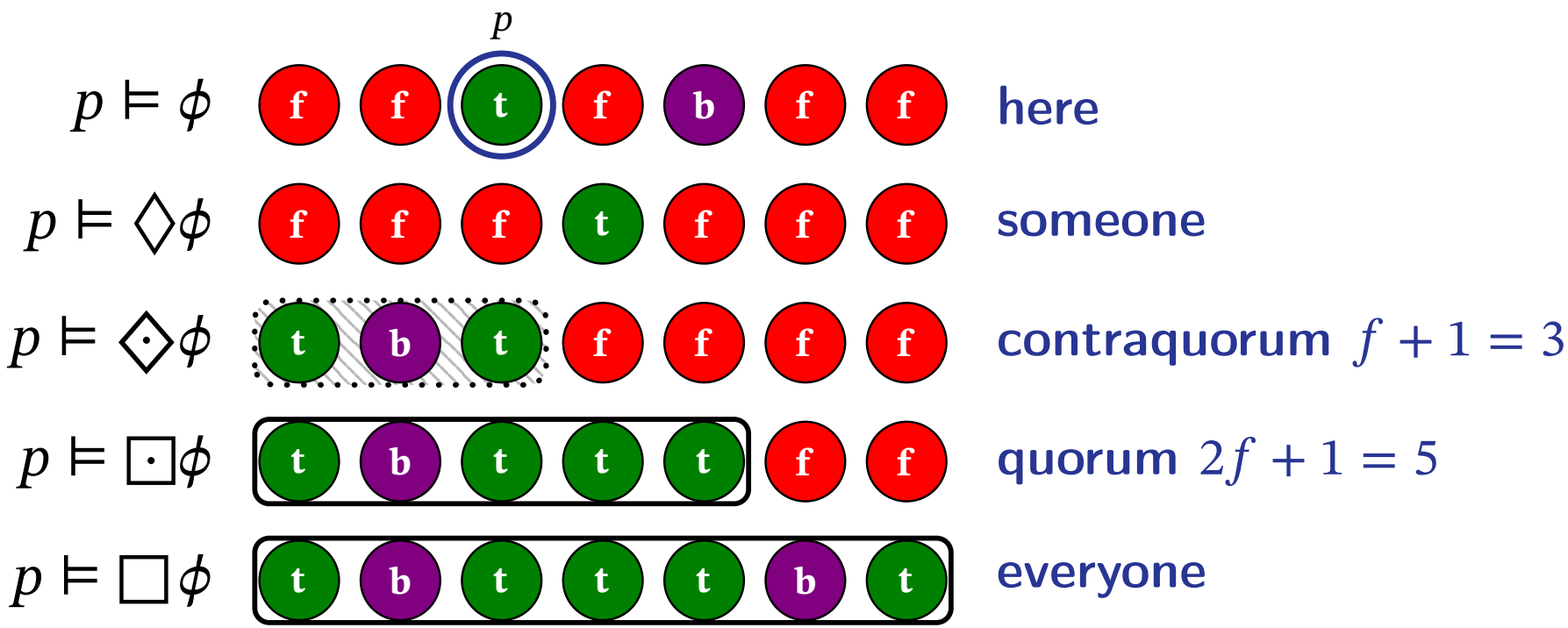
Truth-value modalities: $\mathbb{TF}$, $\mathbb{TB}$, and $\mathbb{B}$

$$\mathbf{3} = \{\mathbf{t}, \mathbf{f}, \mathbf{b}\}$$



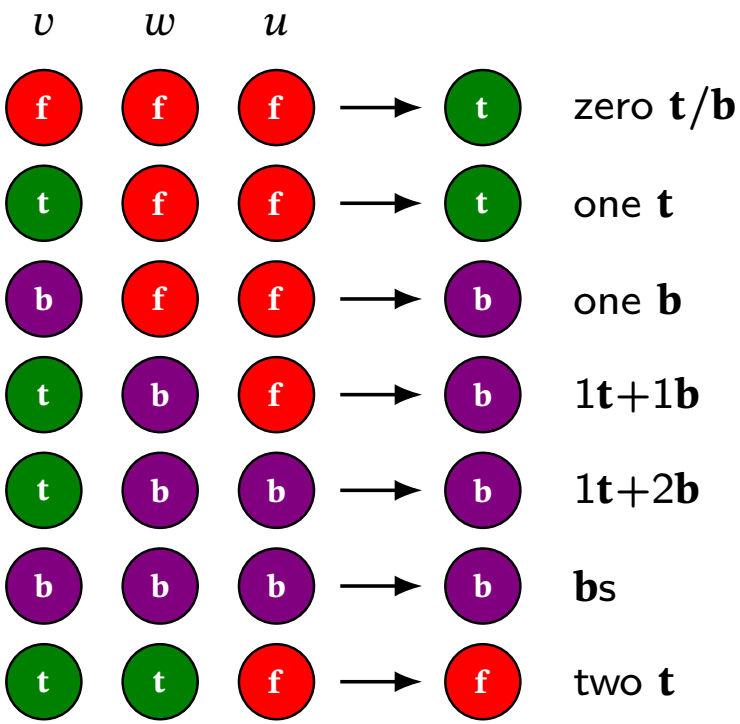
For each modality's oval, inside maps to $\mathbf{t}$; outside maps to $\mathbf{f}$

Spatial modalities $\Diamond$, $\diamondsuit$, $\boxdot$, and $\Box$ for $\#P = 3f + 1$ and $\#O \geq 2f + 1$ and $f = 2$

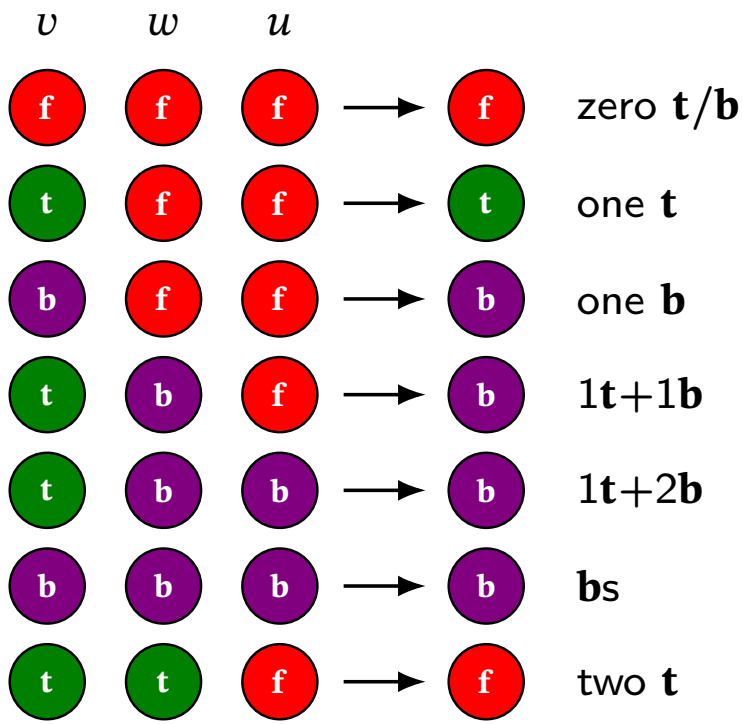


$\exists_{01}$ and $\exists_1$: count ts, treat b as 'potentially either'

$\exists_{01} v.\phi(v)$: at most one



$\exists_1 v.\phi(v)$: precisely one



Declarative Bracha Broadcast ThyDBB is the following set

Backward rules

(**BrDeliver**?) $\mathtt{dlvr}(v) \to \boxdot\mathtt{ready}(v)$

(**BrReady**?) $\mathtt{ready}(v) \to \boxdot\mathtt{echo}(v)$

(**BrEcho**?) $\mathtt{echo}(v) \to \Diamond\mathtt{brdcst}(v)$

Forward rules

(**BrDeliver**!) $\boxdot\mathtt{ready}(v) \to \mathtt{dlvr}(v)$

(**BrReady**!) $\boxdot\mathtt{echo}(v) \to \mathtt{ready}(v)$

(**BrEcho**!) $\Diamond\mathtt{brdcst}(v) \to \mathtt{echo}(v)$

(**BrReady**!!) $\Diamond\!\!\cdot\mathtt{ready}(v) \to \mathtt{ready}(v)$

Other rules

(**BrEcho01**) $\exists_{\mathbf{01}} v.\mathtt{echo}(v)$

(**BrBroadcast1**) $\exists_{\mathbf{1}} v.\Diamond\mathtt{brdcst}(v)$

(**BrCorrect**) $\boxdot\mathbb{F}[\mathtt{ready}, \mathtt{echo}, \mathtt{brdcst}]$

(**BrCorrect**′) $\mathbb{F}[\mathtt{P}] \vee \mathtt{B}[\mathtt{P}] \quad (\mathtt{P} \in \{\mathtt{ready}, \mathtt{echo}\})$

(**BrCorrect**″) $\Box\mathbb{F}[\mathtt{brdcst}] \vee \Box\mathtt{B}[\mathtt{brdcst}]$

Declarative Bracha Broadcast ThyDBB is not this

Backward rules

(**BrDeliver**?) $\texttt{dlvr}(v) \rightarrow \boxdot\texttt{ready}(v)$

(**BrReady**?) $\texttt{ready}(v) \rightarrow (\boxdot\texttt{echo}(v) \vee \diamondsuit\texttt{ready}(v))$

(**BrEcho**?) $\texttt{echo}(v) \rightarrow \diamondsuit\texttt{brdcst}(v)$

Forward rules

(**BrDeliver**!) $\boxdot\texttt{ready}(v) \rightarrow \texttt{dlvr}(v)$

(**BrReady**!) $\boxdot\texttt{echo}(v) \rightarrow \texttt{ready}(v)$

(**BrEcho**!) $\diamondsuit\texttt{brdcst}(v) \rightarrow \texttt{echo}(v)$

(**BrReady**!!) $\diamondsuit\boxdot\texttt{ready}(v) \rightarrow \texttt{ready}(v)$

Other rules

(**BrEcho01**) $\exists_{01} v.\texttt{echo}(v)$

(**BrBroadcast1**) $\exists_{1} v.\diamondsuit\texttt{brdcst}(v)$

(**BrCorrect**) $\boxdot\mathbb{F}[\texttt{ready}, \texttt{echo}, \texttt{brdcst}]$

(**BrCorrect**′) $\mathbb{F}[\mathtt{P}] \vee \mathtt{B}[\mathtt{P}] \quad (\mathtt{P} \in \{\texttt{ready}, \texttt{echo}\})$

(**BrCorrect**″) $\Box\mathbb{F}[\texttt{brdcst}] \vee \Box\mathtt{B}[\texttt{brdcst}]$

Semantics, for reference

$$[\![\mathrm{P}(v)]\!]_p = [\![\mathrm{P}]\!](p, v)$$

$$[\![\neg\phi]\!]_p = \neg[\![\phi]\!]_p \qquad [\![\phi \vee \phi']\!]_p = [\![\phi]\!]_p \vee [\![\phi']\!]_p$$

$$[\![\phi \rightharpoonup \phi']\!]_p = [\![\phi]\!]_p \rightharpoonup [\![\phi']\!]_p \qquad [\![\phi \to \phi']\!]_p = [\![\phi]\!]_p \to [\![\phi']\!]_p$$

$$[\![\boxdot\phi]\!]_p = \bigvee_{O \in \mathsf{Opn}_{\neq\emptyset}} \bigwedge_{p' \in O} [\![\phi]\!]_{p'} \qquad [\![\diamondsuit\phi]\!]_p = \bigwedge_{O \in \mathsf{Opn}_{\neq\emptyset}} \bigvee_{p' \in O} [\![\phi]\!]_{p'}$$

$$[\![\Box\phi]\!]_p = \bigwedge_{p' \in O} [\![\phi]\!]_{p'} \qquad [\![\Diamond\phi]\!]_p = \bigvee_{p' \in O} [\![\phi]\!]_{p'}$$

$$[\![\exists_{\mathbf{01}} v.\phi(v)]\!]_p = \bigwedge_{v,v' \in \mathsf{Val}} [\![\phi(v)]\!]_p \to ([\![\phi(v')]\!]_p \to v{=}v')$$

$$[\![\exists_{\mathbf{1}} v.\phi(v)]\!]_p = \bigvee_{v \in \mathsf{Val}} [\![\phi(v)]\!]_p \wedge [\![\exists_{\mathbf{01}} v.\phi(v)]\!]_p$$

$$[\![\mathbb{F}\,\phi]\!]_p = \mathbb{F}\,[\![\phi]\!]_p \qquad [\![\mathbb{B}\,\phi]\!]_p = \mathbb{B}\,[\![\phi]\!]_p$$

$p \vDash \phi$ means $[\![\phi]\!]_p \in \{\mathbf{t}, \mathbf{b}\}$ and $\vDash \phi$ means $\forall p \in \mathsf{Pnt}.(p \vDash \phi)$.

An axiom or correctness property ϕ is considered valid in an interpretation $[\![\text{-}]\!]$ when $\vDash \phi$.

Some notation

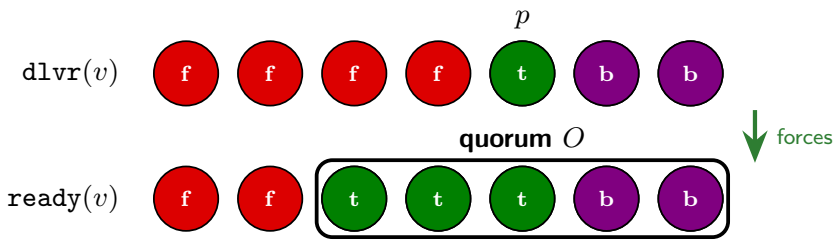
$\exists_{\mathbf{01}} v.\phi(v)$ is shorthand for $\forall v, v'.\phi(v) \to \phi(v') \to v = v'$.

- This means $\phi(v)$ is $\mathbf{t}$ for *at most* one v.
- $\phi(v)$ can be $\mathbf{f}$ or $\mathbf{b}$ for as many v as it wants!
- In particular, $\exists_{\mathbf{01}} v.\mathbf{b}$ is valid.

$\exists_{\mathbf{1}} v.\phi(v)$ is shorthand for $\exists_{\mathbf{01}} v.\phi(v) \wedge \exists v.\phi$.

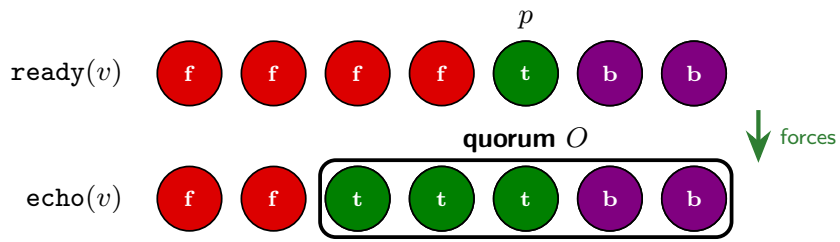
- This means $\phi(v)$ is $\mathbf{t}$ for *at most* one v, and is in $\{\mathbf{t}, \mathbf{b}\}$ for *at least* one v.
- In particular, $\exists_{\mathbf{1}} v.\mathbf{b}$ is valid (specifically, $[\![\exists_{\mathbf{1}} v.\mathbf{b}]\!]_w = \mathbf{b}$ always).

(BrDeliver?) $\texttt{dlvr}(v) \rightarrow \boxdot\texttt{ready}(v)$



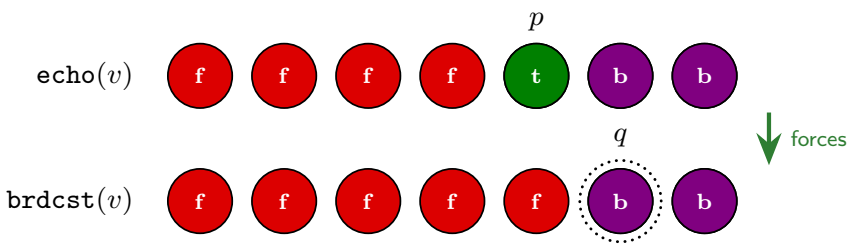
t-delivery at p requires **tb**-ready quorum, with up to $f{=}2$ **b**-nodes.

(BrReady?) $\texttt{ready}(v) \rightarrow \boxdot\texttt{echo}(v)$



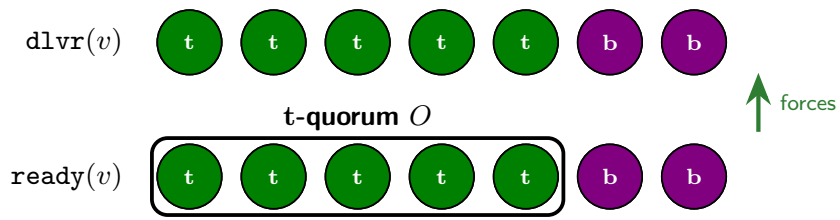
t-ready at p requires **tb**-echo quorum, with up to $f{=}2$ **b**-nodes.

(BrEcho?) $\texttt{echo}(v) \rightarrow \Diamond\texttt{brdcst}(v)$



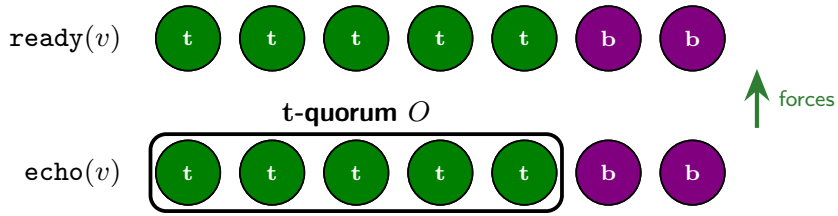
t-echo requires **tb**-broadcast witness.

(BrDeliver!) $\boxdot\texttt{ready}(v) \rightarrow \texttt{dlvr}(v)$



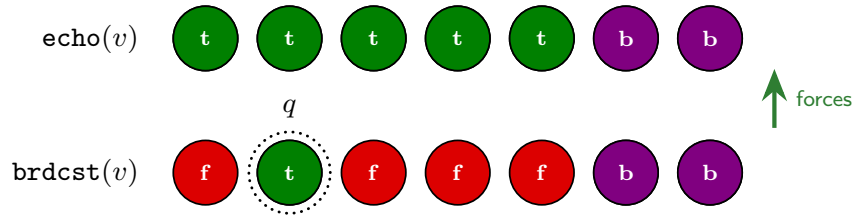
t-ready quorum forces **tb**-$\texttt{dlvr}(v)$ at every node, with up to $f{=}2$ **b**-nodes.

(BrReady!) $\boxdot \texttt{echo}(v) \rightarrow \texttt{ready}(v)$



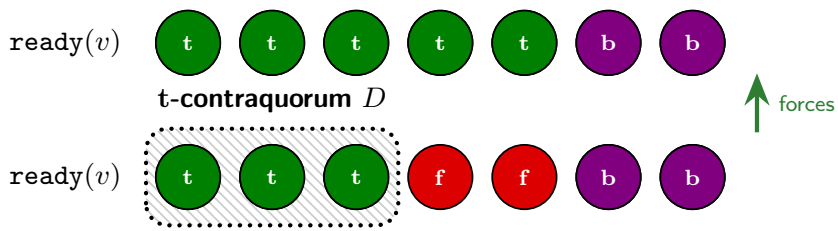
t-echo quorum forces **tb**-ready(v) at every node, with up to $f{=}2$ **b**-nodes.

(BrEcho!) $\Diamond \texttt{brdcst}(v) \rightarrow \texttt{echo}(v)$



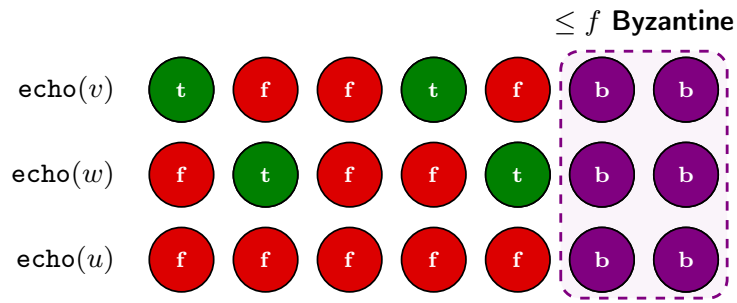
One **t**-broadcast witness forces **tb**-echo(v) at every node, with up to $f{=}2$ **b**-nodes.

(BrReady!!) $\Diamond \texttt{ready}(v) \rightarrow \texttt{ready}(v)$



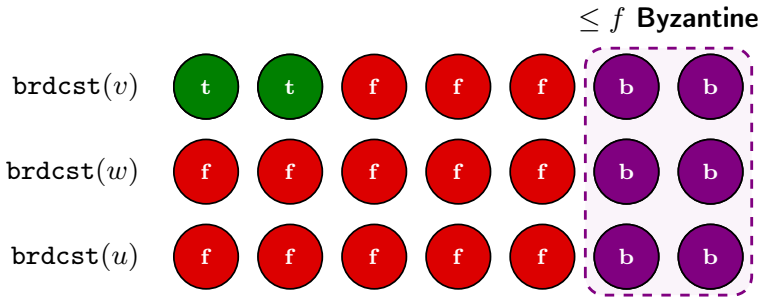
t-ready contraquorum ($f + 1 = 3$) forces **tb**-ready(v) everywhere, with up to $f{=}2$ **b**-nodes.

(BrEcho01) $\exists_{01} v.\, \texttt{echo}(v)$



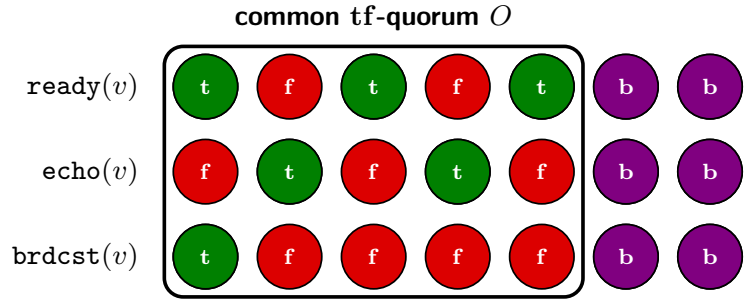
Each correct column has at most one green echo value; up to $f{=}2$ **b**-columns.

(BrBroadcast1) $\exists_1 v.\, \Diamond \mathtt{brdcst}(v)$



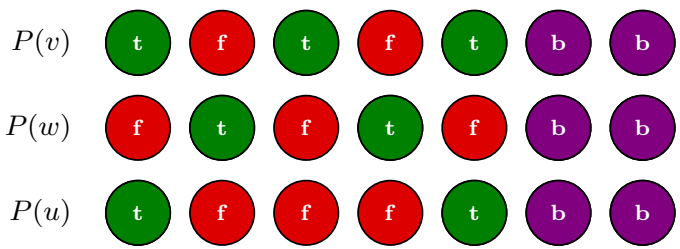
Only one value may have a green broadcast witness.
Up to $f{=}2$ purple columns may witness any value.

(BrCorrect) $\boxdot \mathtt{TF}[\mathtt{ready}, \mathtt{echo}, \mathtt{brdcst}]$



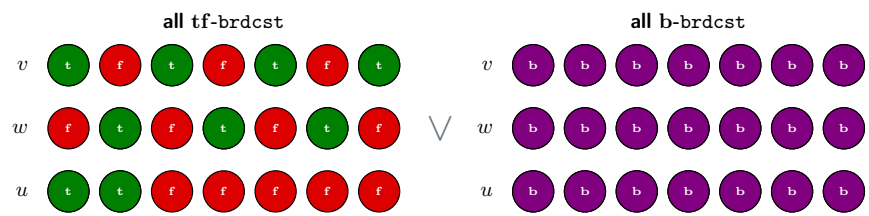
One quorum is red/green for all three
predicates; at most $f{=}2$ columns lie outside it.

(BrCorrect′) $\mathtt{TF}[P] \vee \mathtt{B}[P]$ $(P \in \{\mathtt{ready}, \mathtt{echo}\})$



Each column red/green throughout or purple
throughout; at most $f{=}2$ columns purple.

(BrCorrect″) $\Box \mathtt{TF}[\mathtt{brdcst}] \vee \Box \mathtt{B}[\mathtt{brdcst}]$



Either brdcst is red/green everywhere, or it is purple everywhere (the Byzantine-sender branch

Correctness properties (informal; from literature)

Correctness properties are sourced from pages 112 and 117 of *Introduction to reliable and secure distributed programming* (2nd ed):

1. Validity:
   If a correct process broadcasts some value v, then every correct process delivers v. *'Correct' not immediately defined in source.*
2. No duplication:
   Every correct process delivers at most one value.
3. Integrity:
   If some correct process delivers a message v with correct sender p, then v was broadcast by p.
4. Consistency:
   If one correct process delivers v and another correct process delivers v', then $v = v'$.
5. Totality:
   If some correct process delivers v, then every correct process delivers v.

3-twined, axiomatically

Call a semitopology (Pnt, Opn) 3-twined when any three nonempty open sets intersect (also called Q^3; see e.g. Definition 2 here).

Lemma: The following are equivalent [Lemma 5.3.1(2)]:

1. (Pnt, Opn) is 3-twined.
2. $\vDash (\boxdot\phi \wedge \boxdot\psi \wedge \boxdot\chi) \Rightarrow \diamondsuit(\phi \wedge \psi \wedge \chi)$ (for any ϕ, ψ, and χ).
3. $\vDash (\boxdot\phi \wedge \boxdot\psi) \Rightarrow \diamondsuit\!\!\!\cdot(\phi \wedge \psi)$.
4. $\vDash (\boxdot \mathbb{F}\phi \wedge \boxdot\phi) \Rightarrow \mathbb{T}\diamondsuit\!\!\!\cdot\phi$. *Correct on a quorum and valid on a quorum implies true on a contraquorum!*

$p \vDash \boxdot\phi$ means $\exists O \in \mathrm{Opn}_{\neq\emptyset}. \forall p' \in O. p' \vDash \phi$.

$p \vDash \phi \Rightarrow \phi'$ means $p \vDash \mathbb{B}\phi \Rightarrow p \vDash \mathbb{B}\phi'$ (ϕ valid implies ϕ' valid).

Correctness properties (in the logic)

Theorem: Suppose $[\![\text{-}]\!]$ is a model of **ThyDBB** over a 3-twined semitopology (i.e. an interpretation $[\![\text{-}]\!]$ such that $\vDash \mathtt{Ax}$ for every $\mathtt{Ax} \in$ **ThyDBB**). Then:

Validity:	$\vDash \Diamond\mathtt{brdcst}(v) \to \Box\mathtt{dlvr}(v)$
No duplication:	$\vDash \exists_{\mathbf{01}} v.\mathtt{dlvr}(v)$
Integrity:	$\vDash \mathtt{dlvr}(v) \to \Diamond\mathtt{brdcst}(v)$
Consistency:	$\vDash \exists_{\mathbf{01}} v.\Diamond\mathtt{dlvr}(v)$
Totality:	$\vDash \Diamond\mathtt{dlvr}(v) \to \Box\mathtt{dlvr}(v)$

- I don't explicitly represent messages! There are modal predicates evaluating to truth-values.
- p in informal integrity property corresponds to $\Diamond$ in (**Integrity**).
- Asymmetry between (**Validity**) and (**Integrity**).
- (**BrBroadcast1**).
- Paper proofs. AI model check.

Taking stock

We can present distributed algorithms declaratively as axiomatic theories.

Paragraphs of English text get distilled down to *precise axiomatic assertions*.

Difficult reasoning gets recast as formal logical reasoning. This is often routine lemma-following and symbol-pushing.

It's a powerful technique. Axiomatic reasoning is not new, but it's not been applied in this way.

. . . and it works! I find it indispensable for my own understanding, and it's already helped to catch real bugs. Cf. story of HBB and Controllers.