Linear Logical Voting Protocols

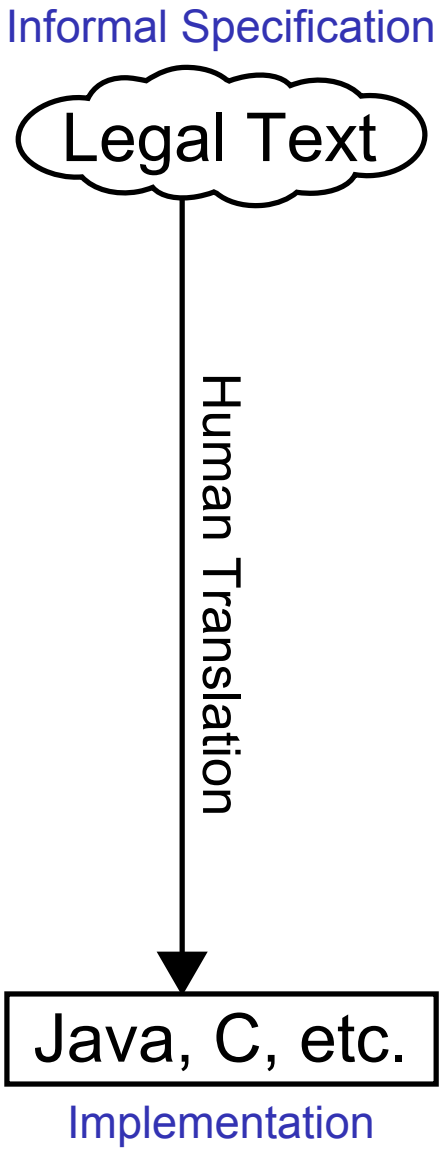
Henry DeYoung[1] Carsten Schürmann[2]

[1]Carnegie Mellon University

[2]IT University of Copenhagen

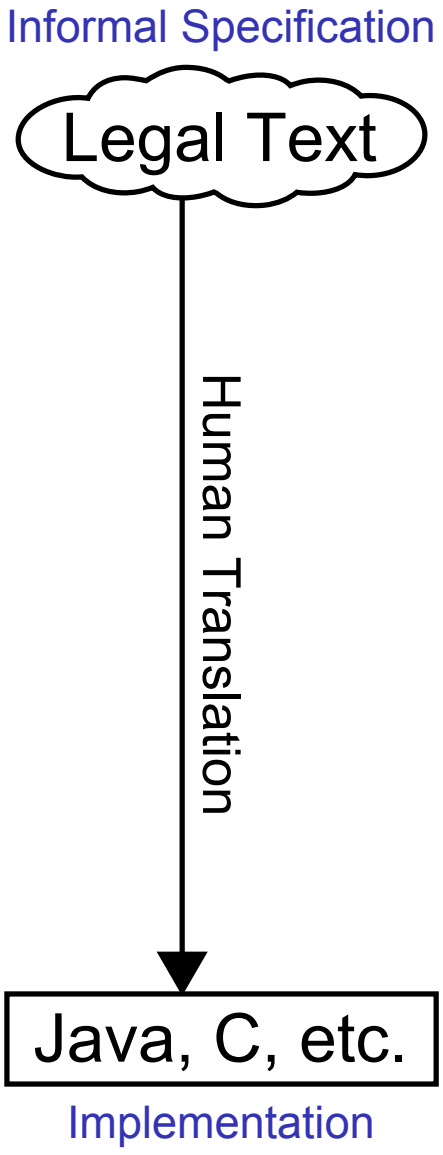
3rd International Conference on E-Voting and Identity
September 29, 2011

Current Approach to Electronic Elections



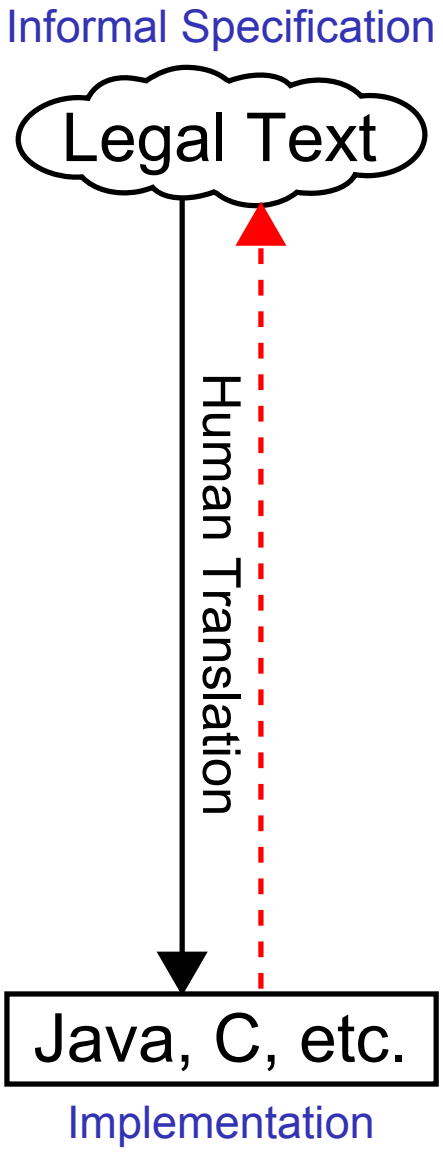
1. Translate legal text to imperative source code.

Current Approach to Electronic Elections



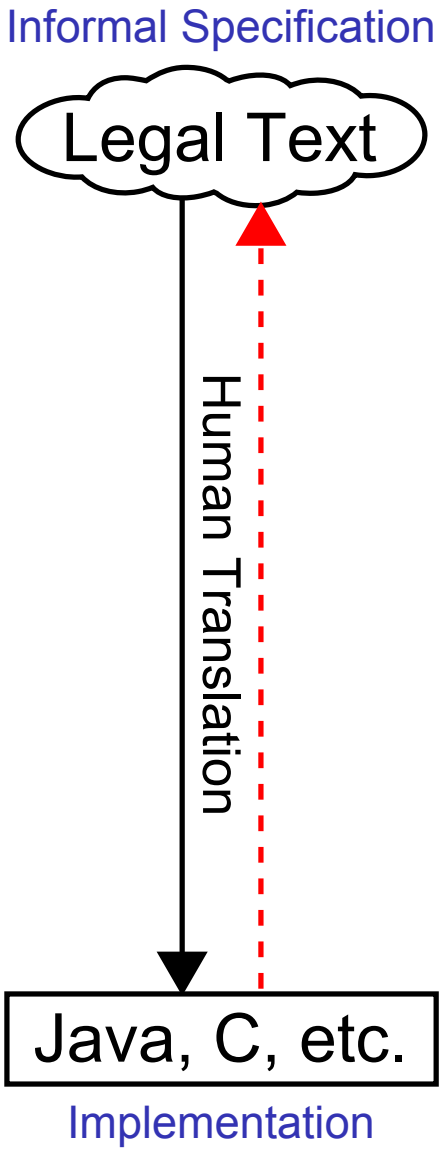
1. Translate legal text to imperative source code.
   - How to trust this?

Current Approach to Electronic Elections



1. Translate legal text to imperative source code.
   - How to trust this?
2. Certify that code meets legal specification.
   - *Very* hard!

Current Approach to Electronic Elections



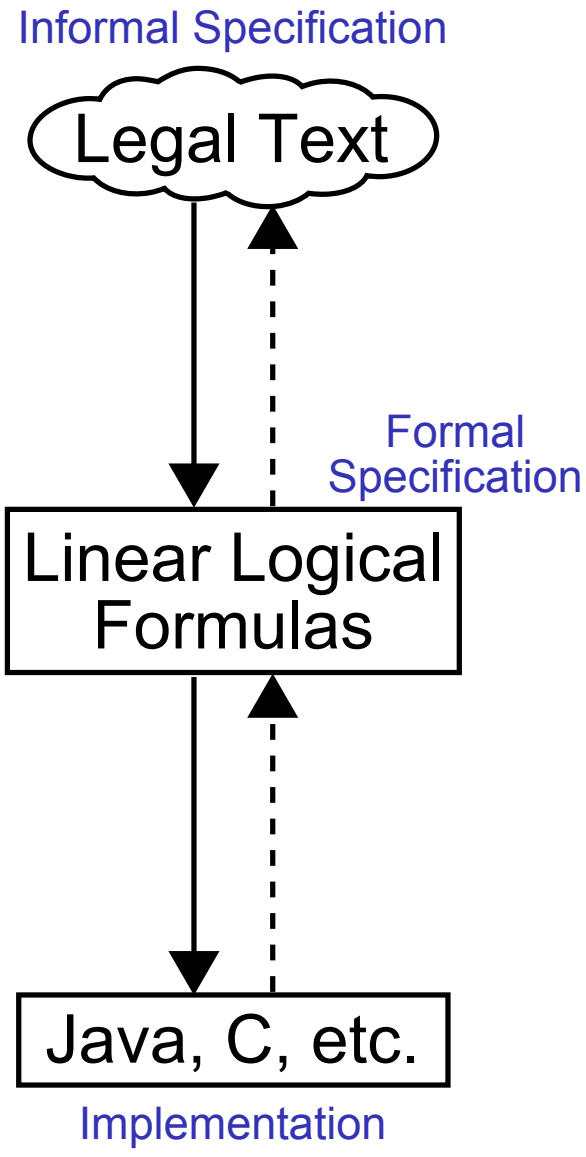
1. Translate legal text to imperative source code.
   - How to trust this?
2. Certify that code meets legal specification.
   - *Very* hard!

Is this approach really trustworthy?

Key Idea

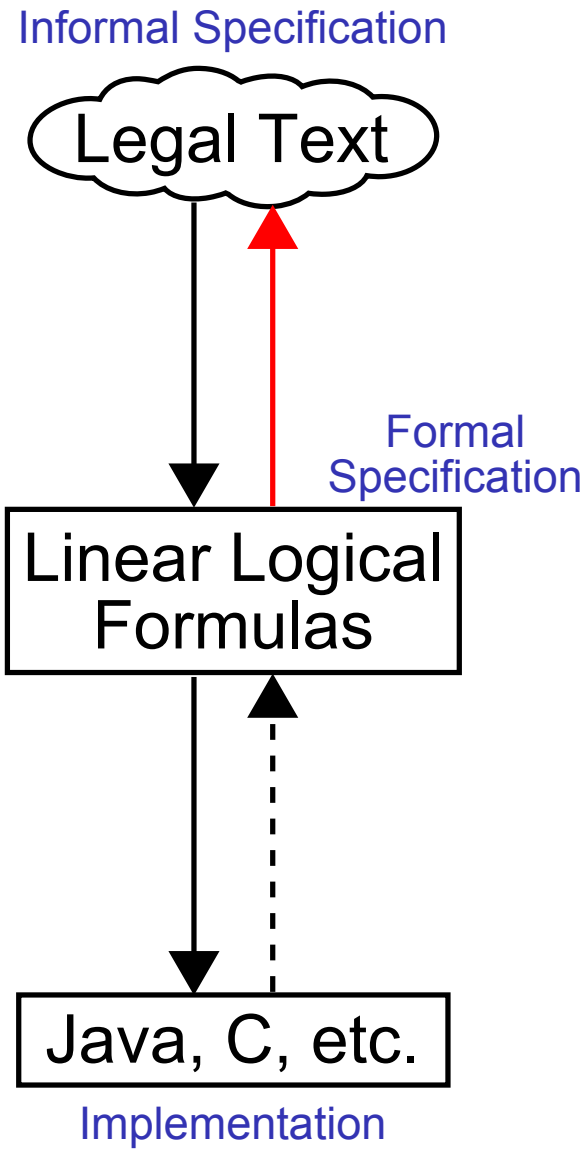
Formal logic, particularly linear logic, is well-suited to the trustworthy specification and implementation of voting protocols.

Our Approach to Electronic Elections



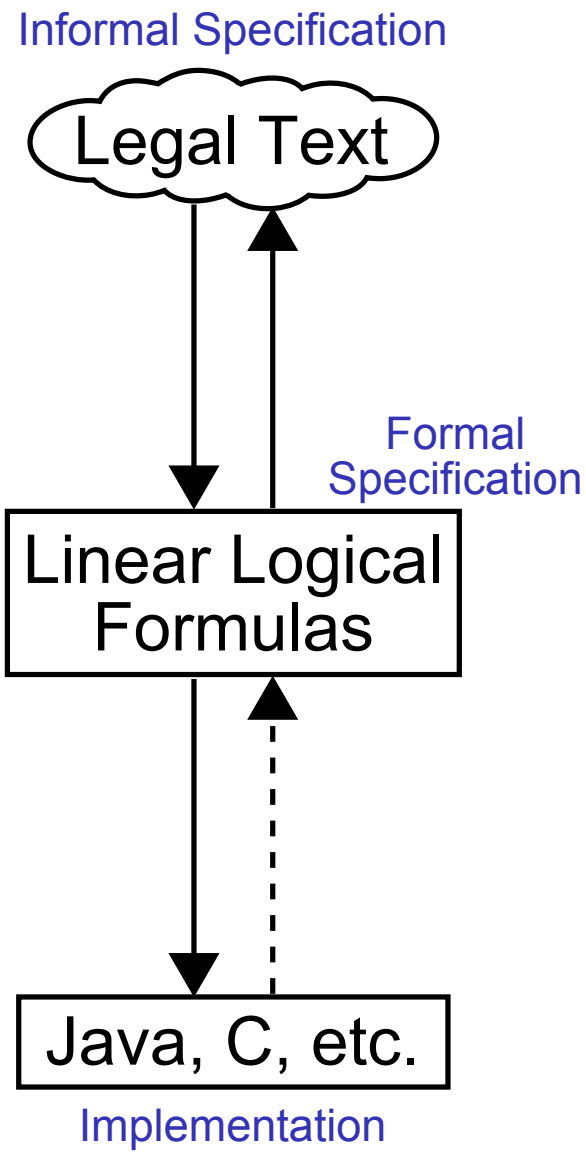
1. Translate legal text to logical formulas.
   - Algorithms at high level of abstraction

Our Approach to Electronic Elections



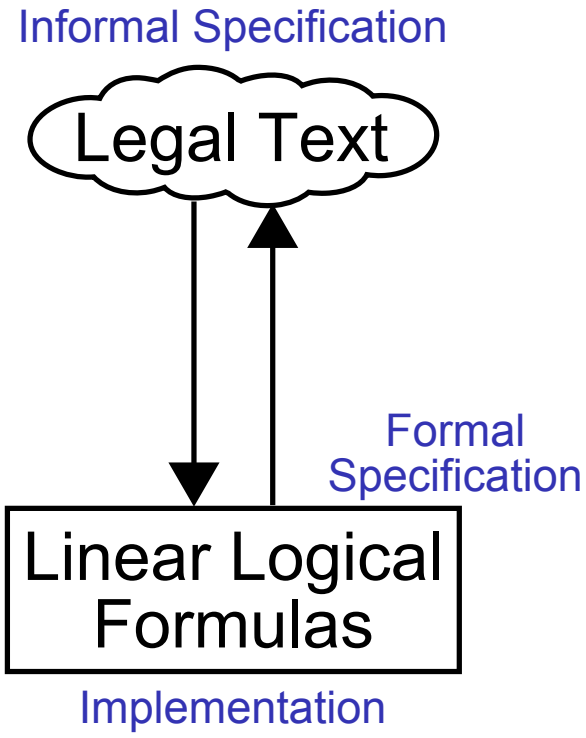
1. Translate legal text to logical formulas.
   - Algorithms at high level of abstraction
   - Much smaller gap from legal language!

Our Approach to Electronic Elections



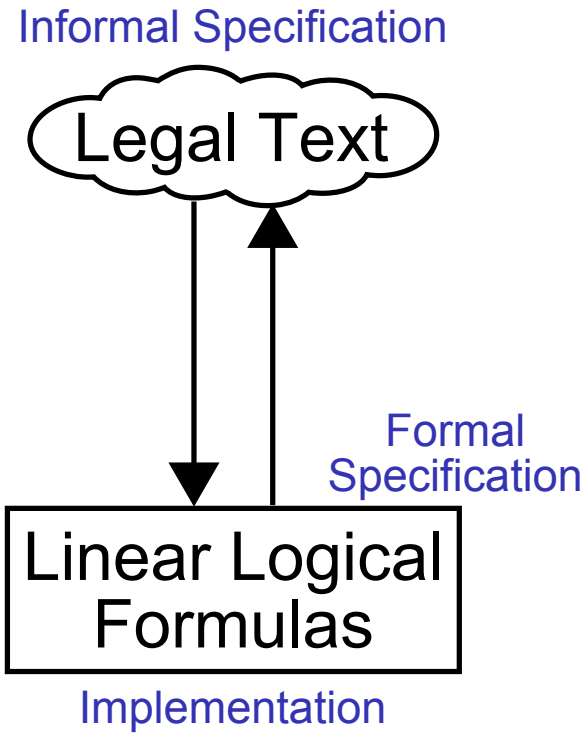
1. Translate legal text to logical formulas.
   - Algorithms at high level of abstraction
   - Much smaller gap from legal language!
2. Transliterate formulas to a logic program.
   - Formulas = source code

Our Approach to Electronic Elections



1. Translate legal text to logical formulas.
   - Algorithms at high level of abstraction
   - Much smaller gap from legal language!
2. Transliterate formulas to a logic program.
   - Formulas = source code
   - No further translation necessary!

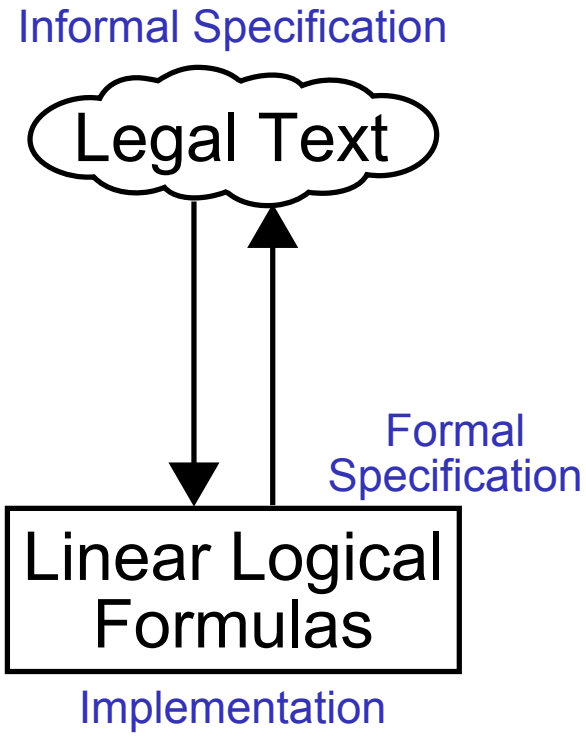
Our Approach to Electronic Elections



What must still be trusted?

1. Translation to logical formulas
   - Much smaller gap from legal language — more trustworthy!

Our Approach to Electronic Elections



What must still be trusted?

1. Translation to logical formulas
   - Much smaller gap from legal language — more trustworthy!
2. Correctness of logic programming engine
   - Equal to or easier than trusting compiler
   - Proof witnesses are audit trails for free.
   - Use a simpler proof checker to validate proof objects.

Contributions and Non-Contributions of Our Paper

Contributions:

- Full linear logical specifications of:
  - Single-winner first-past-the-post (SW-FPTP)
  - Single transferable vote (STV)
- Operational interpretation as Celf logic programs
- Proof sketch of correctness of SW-FPTP specification

Contributions and Non-Contributions of Our Paper

Contributions:

- Full linear logical specifications of:
  - Single-winner first-past-the-post (SW-FPTP)
  - Single transferable vote (STV)
- Operational interpretation as Celf logic programs
- Proof sketch of correctness of SW-FPTP specification

Non-contributions:

- Focus is on *verified* elections, not *voter-verifiable* elections.
  - Complementary: E2E techniques detect errors at run time; verified software minimizes run-time errors.
- Only operational correctness, not security properties
  - Linear logical voting specs should be readily amenable to meta-reasoning [Reed '09] about security; left to future work.

In This Talk

1. A Brief Introduction to Linear Logic
2. A Linear Logical Specification of Single Transferable Vote
3. Conclusion

A Brief Introduction to Linear Logic

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.

Linear Logic

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of food.

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.
- Use linear implication, $A \multimap \{B\}$.
  - $A \multimap \{B\} \approx$ "*consume* resource A to produce B."

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.
- Use linear implication, $A \multimap \{B\}$.
  - $A \multimap \{B\} \approx$ "*consume* resource A to produce B."

$$\mathit{voting\text{-}auth\text{-}card} \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I *give* an authorization card, then I *get* a blank ballot."

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.
- Use linear implication, $A \multimap \{B\}$.
  - $A \multimap \{B\} \approx$ "*consume* resource A to produce B."

$$\textit{voting-auth-card} \multimap \{\textit{blank-ballot}\}$$

"If I *give* an authorization card, then I *get* a blank ballot."

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.
- Use linear implication, $A \multimap \{B\}$.
  - $A \multimap \{B\} \approx$ "*consume* resource A to produce B."

$$\mathit{voting\text{-}auth\text{-}card} \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I *give* an authorization card, then I *get* a blank ballot."

What is linear logic?

Traditional Logic

"Truth is free."

- Assumptions may be used any number of times.
- The logic of facts.

Linear Logic

"Truth is a consumable resource."

- Assumptions must be used exactly once.
- The logic of ~~food~~ voting.

Let's specify a voter check-in process:

- *Consume* an authorization card to prevent multiple check-ins.
- Use linear implication, $A \multimap \{B\}$.
  - $A \multimap \{B\} \approx$ "*consume* resource A to produce B."

$$\textit{voting-auth-card} \multimap \{\textit{blank-ballot}\}$$

"If I *give* an authorization card, then I *get* a blank ballot."

“May I please see your identification?”

$$\textit{voting-auth-card} \text{ “and photo ID”} \multimap \{\textit{blank-ballot}\}$$

“If I give an auth. card and a photo ID, then I get a ballot.”

Problem: How to express a pair of resources?

"May I please see your identification?"

$$\textit{voting-auth-card} \text{ "and photo ID"} \multimap \{\textit{blank-ballot}\}$$

"If I give an auth. card and a photo ID, then I get a ballot."

Problem: How to express a pair of resources?

Solution: Use simultaneous conjunction, $A \otimes B$.

- $A \otimes B \approx$ "both resources A and B"

"May I please see your identification?"

$$\textit{voting-auth-card} \otimes \textit{photo-ID} \multimap \{\textit{blank-ballot}\}$$

"If I give an auth. card and a photo ID, then I get a ballot."

Problem: How to express a pair of resources?

Solution: Use simultaneous conjunction, $A \otimes B$.

- $A \otimes B \approx$ "both resources A and B"

"May I please see your identification?"

$$\textit{voting-auth-card} \otimes \textit{photo-ID} \multimap \{\textit{blank-ballot}\}$$

"If I give an auth. card and a photo ID, then I get a ballot."

Problem: How to express a pair of resources?

Solution: Use simultaneous conjunction, $A \otimes B$.

- $A \otimes B \approx$ "both resources A and B"

Problem: Linear implication incorrectly consumes the photo ID.
In linear logic, how do we only *show* the ID?

"May I please see your identification?"

$$\textit{voting-auth-card} \otimes \textit{photo-ID} \multimap \{\textit{blank-ballot}\}$$

"If I give an auth. card and a photo ID, then I get a ballot."

Problem: How to express a pair of resources?

Solution: Use simultaneous conjunction, $A \otimes B$.

- $A \otimes B \approx$ "both resources A and B"

Problem: Linear implication incorrectly consumes the photo ID.
In linear logic, how do we only *show* the ID?

Solution: Use the unrestricted modality, $!A$.

- $!A \approx$ "a version of A that is never consumed."

"May I please see your identification?"

$$\mathit{voting\text{-}auth\text{-}card} \otimes !\mathit{photo\text{-}ID} \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a photo ID, then I get a ballot."

Problem: How to express a pair of resources?

Solution: Use simultaneous conjunction, $A \otimes B$.

- $A \otimes B \approx$ "both resources A and B"

Problem: Linear implication incorrectly consumes the photo ID.
In linear logic, how do we only *show* the ID?

Solution: Use the unrestricted modality, $!A$.

- $!A \approx$ "a version of A that is never consumed."

Ensuring that the Card and ID Match

$$\mathit{voting\text{-}auth\text{-}card} \otimes !\mathit{photo\text{-}ID} \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a photo ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Ensuring that the Card and ID Match

$$\mathit{voting\text{-}auth\text{-}card} \otimes !\mathit{photo\text{-}ID} \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a photo ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Solution: Use universal quantification, $\forall x.A$.

- Quantified variables are not resources.

Ensuring that the Card and ID Match

$$\forall v.\, \mathit{voting\text{-}auth\text{-}card}(v) \otimes !\mathit{photo\text{-}ID}(v) \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a matching ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Solution: Use universal quantification, $\forall x.A$.

- Quantified variables are not resources.

Ensuring that the Card and ID Match

$$\mathit{voting\text{-}auth\text{-}card}(V) \otimes !\mathit{photo\text{-}ID}(V) \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a matching ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Solution: Use universal quantification, $\forall x.A$.

- Quantified variables are not resources.

Ensuring that the Card and ID Match

$$\mathit{voting\text{-}auth\text{-}card}(V) \otimes !\mathit{photo\text{-}ID}(V) \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a matching ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Solution: Use universal quantification, $\forall x.A$.

- Quantified variables are not resources.

Problem: Doesn't ensure that the auth. card and ID are mine.

Ensuring that the Card and ID Match

$$\mathit{voting\text{-}auth\text{-}card}(V) \otimes !\mathit{photo\text{-}ID}(V) \multimap \{\mathit{blank\text{-}ballot}\}$$

"If I give an auth. card and show a matching ID, then I get a ballot."

Problem: Doesn't ensure that auth. card and photo ID match.

Solution: Use universal quantification, $\forall x.A$.

- Quantified variables are not resources.

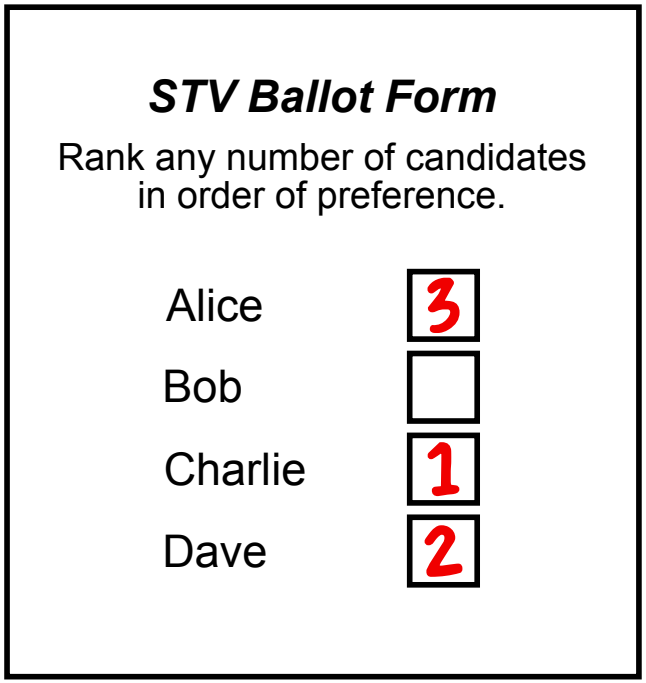
Problem: Doesn't ensure that the auth. card and ID are mine.

Future Work: Use possession modality [Garg$^+$ '06], K has A.

- Location and secrecy of information?
  Homomorphic encryption?

A Linear Logical Specification of Single Transferable Vote

Single Transferable Vote (STV)



Outline of STV Protocol:

0. Calculate the quota of votes.
1. Tally each ballot for its highest pref that is neither elected nor defeated.
   - Surplus votes go to next pref.
2. After all votes have been tallied:
   - If there are more cands. than seats, eliminate cand. with the fewest votes; transfer his votes and re-tally (go to 1).
   - If there are more seats than cands., then all remaining cands. are elected.

Single Transferable Vote on a Single Slide

begin/1 :
$begin(S, H, U) \otimes$
$!(Q = U/(S{+}1) + 1)$
$\multimap \{!quota(Q) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/1 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 < Q)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$hopeful(C, N{+}1) \otimes$
$tally\text{-}votes(S, H, U{-}1)\}$

tally/2 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 \geq Q) \otimes$
$!(S \geq 1)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$!elected(C) \otimes$
$tally\text{-}votes(S{-}1, H{-}1, U{-}1)\}$

tally/3 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [C' \mid L]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{uncounted\text{-}ballot(C', L) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/4 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [\,]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{tally\text{-}votes(S, H, U{-}1)\}$

tally/5 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S < H)$
$\multimap \{defeat\text{-}min(S, H, 0)\}$

tally/6 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S \geq H)$
$\multimap \{!elect\text{-}all\}$

defeat-min/1 :
$defeat\text{-}min(S, H, M) \otimes$
$hopeful(C, N)$
$\multimap \{minimum(C, N) \otimes$
$defeat\text{-}min(S, H{-}1, M{+}1)\}$

defeat-min/2 :
$defeat\text{-}min(S, 0, M)$
$\multimap \{defeat\text{-}min'(S, 0, M)\}$

defeat-min′/1 :
$defeat\text{-}min'(S, H, M) \otimes$
$minimum(C_1, N_1) \otimes$
$minimum(C_2, N_2) \otimes$
$!(N_1 \leq N_2)$
$\multimap \{minimum(C_1, N_1) \otimes$
$hopeful(C_2, N_2) \otimes$
$defeat\text{-}min'(S, H{+}1, M{-}1)\}$

defeat-min′/2 :
$defeat\text{-}min'(S, H, 1) \otimes$
$minimum(C, N)$
$\multimap \{!defeated(C) \otimes$
$transfer(C, N, S, H, 0)\}$

transfer/1 :
$transfer(C, N, S, H, U) \otimes$
$counted\text{-}ballot(C, L)$
$\multimap \{uncounted\text{-}ballot(C, L) \otimes$
$transfer(C, N{-}1, S, H, U{+}1)\}$

transfer/2 :
$transfer(C, 0, S, H, U)$
$\multimap \{tally\text{-}votes(S, H, U)\}$

elect-all/1 :
$!elect\text{-}all \otimes$
$hopeful(C, N)$
$\multimap \{!elected(C)\}$

Single Transferable Vote on a Single Slide

begin/1 :
$begin(S, H, U) \otimes$
$!(Q = U/(S{+}1) + 1)$
$\multimap \{!quota(Q) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/1 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 < Q)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$hopeful(C, N{+}1) \otimes$
$tally\text{-}votes(S, H, U{-}1)\}$

tally/2 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 \geq Q) \otimes$
$!(S \geq 1)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$!elected(C) \otimes$
$tally\text{-}votes(S{-}1, H{-}1, U{-}1)\}$

tally/3 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [C' \mid L]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{uncounted\text{-}ballot(C', L) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/4 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [\,]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{tally\text{-}votes(S, H, U{-}1)\}$

tally/5 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S < H)$
$\multimap \{defeat\text{-}min(S, H, 0)\}$

tally/6 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S \geq H)$
$\multimap \{!elect\text{-}all\}$

defeat-min/1 :
$defeat\text{-}min(S, H, M) \otimes$
$hopeful(C, N)$
$\multimap \{minimum(C, N) \otimes$
$defeat\text{-}min(S, H{-}1, M{+}1)\}$

defeat-min/2 :
$defeat\text{-}min(S, 0, M)$
$\multimap \{defeat\text{-}min'(S, 0, M)\}$

defeat-min′/1 :
$defeat\text{-}min'(S, H, M) \otimes$
$minimum(C_1, N_1) \otimes$
$minimum(C_2, N_2) \otimes$
$!(N_1 \leq N_2)$
$\multimap \{minimum(C_1, N_1) \otimes$
$hopeful(C_2, N_2) \otimes$
$defeat\text{-}min'(S, H{+}1, M{-}1)\}$

defeat-min′/2 :
$defeat\text{-}min'(S, H, 1) \otimes$
$minimum(C, N)$
$\multimap \{!defeated(C) \otimes$
$transfer(C, N, S, H, 0)\}$

transfer/1 :
$transfer(C, N, S, H, U) \otimes$
$counted\text{-}ballot(C, L)$
$\multimap \{uncounted\text{-}ballot(C, L) \otimes$
$transfer(C, N{-}1, S, H, U{+}1)\}$

transfer/2 :
$transfer(C, 0, S, H, U)$
$\multimap \{tally\text{-}votes(S, H, U)\}$

elect-all/1 :
$!elect\text{-}all \otimes$
$hopeful(C, N)$
$\multimap \{!elected(C)\}$

Single Transferable Vote on a Single Slide

begin/1 :
$begin(S, H, U) \otimes$
$!(Q = U/(S{+}1) + 1)$
$\multimap \{!quota(Q) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/1 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 < Q)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$hopeful(C, N{+}1) \otimes$
$tally\text{-}votes(S, H, U{-}1)\}$

tally/2 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 \geq Q) \otimes$
$!(S \geq 1)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$!elected(C) \otimes$
$tally\text{-}votes(S{-}1, H{-}1, U{-}1)\}$

tally/3 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [C' \mid L]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{uncounted\text{-}ballot(C', L) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/4 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [\,]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{tally\text{-}votes(S, H, U{-}1)\}$

tally/5 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S < H)$
$\multimap \{defeat\text{-}min(S, H, 0)\}$

tally/6 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S \geq H)$
$\multimap \{!elect\text{-}all\}$

defeat-min/1 :
$defeat\text{-}min(S, H, M) \otimes$
$hopeful(C, N)$
$\multimap \{minimum(C, N) \otimes$
$defeat\text{-}min(S, H{-}1, M{+}1)\}$

defeat-min/2 :
$defeat\text{-}min(S, 0, M)$
$\multimap \{defeat\text{-}min'(S, 0, M)\}$

defeat-min′/1 :
$defeat\text{-}min'(S, H, M) \otimes$
$minimum(C_1, N_1) \otimes$
$minimum(C_2, N_2) \otimes$
$!(N_1 \leq N_2)$
$\multimap \{minimum(C_1, N_1) \otimes$
$hopeful(C_2, N_2) \otimes$
$defeat\text{-}min'(S, H{+}1, M{-}1)\}$

defeat-min′/2 :
$defeat\text{-}min'(S, H, 1) \otimes$
$minimum(C, N)$
$\multimap \{!defeated(C) \otimes$
$transfer(C, N, S, H, 0)\}$

transfer/1 :
$transfer(C, N, S, H, U) \otimes$
$counted\text{-}ballot(C, L)$
$\multimap \{uncounted\text{-}ballot(C, L) \otimes$
$transfer(C, N{-}1, S, H, U{+}1)\}$

transfer/2 :
$transfer(C, 0, S, H, U)$
$\multimap \{tally\text{-}votes(S, H, U)\}$

elect-all/1 :
$!elect\text{-}all \otimes$
$hopeful(C, N)$
$\multimap \{!elected(C)\}$

Single Transferable Vote on a Single Slide

begin/1 :
$begin(S, H, U) \otimes$
$!(Q = U/(S{+}1) + 1)$
$\multimap \{!quota(Q) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/1 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 < Q)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$hopeful(C, N{+}1) \otimes$
$tally\text{-}votes(S, H, U{-}1)\}$

tally/2 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, L) \otimes$
$hopeful(C, N) \otimes$
$!quota(Q) \otimes !(N{+}1 \geq Q) \otimes$
$!(S \geq 1)$
$\multimap \{counted\text{-}ballot(C, L) \otimes$
$!elected(C) \otimes$
$tally\text{-}votes(S{-}1, H{-}1, U{-}1)\}$

tally/3 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [C' \mid L]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{uncounted\text{-}ballot(C', L) \otimes$
$tally\text{-}votes(S, H, U)\}$

tally/4 :
$tally\text{-}votes(S, H, U) \otimes$
$uncounted\text{-}ballot(C, [\,]) \otimes$
$(!elected(C) \oplus !defeated(C))$
$\multimap \{tally\text{-}votes(S, H, U{-}1)\}$

tally/5 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S < H)$
$\multimap \{defeat\text{-}min(S, H, 0)\}$

tally/6 :
$tally\text{-}votes(S, H, 0) \otimes$
$!(S \geq H)$
$\multimap \{!elect\text{-}all\}$

defeat-min/1 :
$defeat\text{-}min(S, H, M) \otimes$
$hopeful(C, N)$
$\multimap \{minimum(C, N) \otimes$
$defeat\text{-}min(S, H{-}1, M{+}1)\}$

defeat-min/2 :
$defeat\text{-}min(S, 0, M)$
$\multimap \{defeat\text{-}min'(S, 0, M)\}$

defeat-min′/1 :
$defeat\text{-}min'(S, H, M) \otimes$
$minimum(C_1, N_1) \otimes$
$minimum(C_2, N_2) \otimes$
$!(N_1 \leq N_2)$
$\multimap \{minimum(C_1, N_1) \otimes$
$hopeful(C_2, N_2) \otimes$
$defeat\text{-}min'(S, H{+}1, M{-}1)\}$

defeat-min′/2 :
$defeat\text{-}min'(S, H, 1) \otimes$
$minimum(C, N)$
$\multimap \{!defeated(C) \otimes$
$transfer(C, N, S, H, 0)\}$

transfer/1 :
$transfer(C, N, S, H, U) \otimes$
$counted\text{-}ballot(C, L)$
$\multimap \{uncounted\text{-}ballot(C, L) \otimes$
$transfer(C, N{-}1, S, H, U{+}1)\}$

transfer/2 :
$transfer(C, 0, S, H, U)$
$\multimap \{tally\text{-}votes(S, H, U)\}$

elect-all/1 :
$!elect\text{-}all \otimes$
$hopeful(C, N)$
$\multimap \{!elected(C)\}$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

- Correspondence between legal text and logical formula is plain!
- Linearity: count ballots only once and update running tallies.

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}$$

- Correspondence between legal text and logical formula is plain!
- Linearity: count ballots only once and update running tallies.

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Tally the votes, assigning each ballot to its highest preference candidate who is neither elected nor defeated."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
the quota wouldn't be reached by this vote,
then mark the ballot as counted and
update C's tally to $N{+}1$ votes and
tally the remaining $U{-}1$ ballots.

tally/1 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}$$

- Correspondence between legal text and logical formula is plain!
- Linearity: count ballots only once and update running tallies.

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
this vote would meet the quota and
there is at least one seat left,
then mark the ballot as counted and
declare candidate C to be elected and
tally the remaining $U-1$ ballots among
the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
this vote would meet the quota and
there is at least one seat left,
then mark the ballot as counted and
declare candidate C to be elected and
tally the remaining $U-1$ ballots among
the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
this vote would meet the quota and
there is at least one seat left,
then mark the ballot as counted and
declare candidate C to be elected and
tally the remaining $U-1$ ballots among
the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and there is an uncounted vote for C and C is a hopeful with running tally N and this vote would meet the quota and there is at least one seat left, then mark the ballot as counted and declare candidate C to be elected and tally the remaining $U-1$ ballots among the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}
$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and there is an uncounted vote for C and C is a hopeful with running tally N and this vote would meet the quota and there is at least one seat left,
then mark the ballot as counted and declare candidate C to be elected and tally the remaining $U-1$ ballots among the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
this vote would meet the quota and
there is at least one seat left,
then mark the ballot as counted and
declare candidate C to be elected and
tally the remaining $U-1$ ballots among
the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and
there is an uncounted vote for C and
C is a hopeful with running tally N and
this vote would meet the quota and
there is at least one seat left,
then mark the ballot as counted and
declare candidate C to be elected and
tally the remaining $U-1$ ballots among
the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"If a candidate reaches the quota, he is declared elected."

Detailed Reading

Otherwise, if we are tallying votes and there is an uncounted vote for C and C is a hopeful with running tally N and this vote would meet the quota and there is at least one seat left, then mark the ballot as counted and declare candidate C to be elected and tally the remaining $U-1$ ballots among the $H-1$ hopefuls and $S-1$ seats left.

tally/2 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \otimes \\
&!(S \geq 1) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad !\mathit{elected}(C) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S{-}1, H{-}1, U{-}1)\}
\end{aligned}$$

- ! modality: once declared elected, always declared elected.

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Any surplus votes go to the next preference listed on the ballot."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is either elected or defeated,
then transfer it to the next pref. C' and
tally the remaining U ballots.

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Any surplus votes go to the next preference listed on the ballot."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is either elected or defeated,
then transfer it to the next pref. C' and
tally the remaining U ballots.

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Any surplus votes go to the next preference listed on the ballot."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is either elected or defeated,
then transfer it to the next pref. C' and
tally the remaining U ballots.

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Any surplus votes go to the next preference listed on the ballot."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is either elected or defeated,
then transfer it to the next pref. C' and
tally the remaining U ballots.

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

Aspect 1: From Legal Text to Formal Specification

Legal Text

"Any surplus votes go to the next preference listed on the ballot."

Detailed Reading

If we are tallying votes and
there is an uncounted vote for C and
C is either elected or defeated,
then transfer it to the next pref. C' and
tally the remaining U ballots.

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

Aspect 2: From Formal Specification to Implementation

tally/1 :

$$
\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, L) \otimes \\
&\mathit{hopeful}(C, N) \otimes \\
&!\mathit{quota}(Q) \otimes !(N{+}1 < Q) \\
&\quad \multimap \{\mathit{counted\text{-}ballot}(C, L) \otimes \\
&\qquad \mathit{hopeful}(C, N{+}1) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U{-}1)\}
\end{aligned}
$$

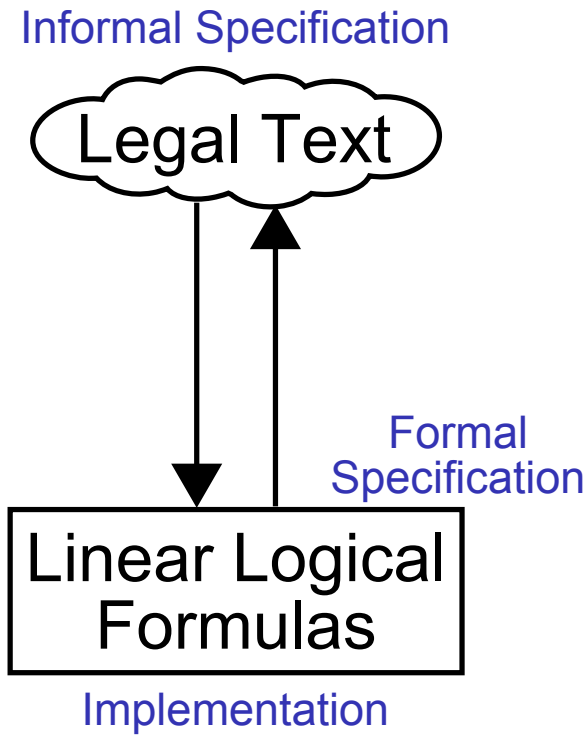
Celf Linear Logic Program

```
tally-votes S H (s U') *
uncounted-ballot C L *
hopeful C N *
!quota Q * !nat-less (s N) Q
   -o {counted-ballot C L *
       hopeful C (s N) *
       tally-votes S H U'}
```

- Transliteration of formula to logic programming syntax
- Complete Celf source code of STV available online at `http://www.itu.dk/~carsten/files/voteid2011.tgz`

Conclusion

Conclusion



Summary

Linear logic is well-suited to the trustworthy specification and implementation of voting protocols, including STV.

begin/1 :

$$begin(S, H, U) \otimes$$
$$!(Q = U/(S{+}1) + 1)$$
$$\multimap \{!quota(Q) \otimes$$
$$tally\text{-}votes(S, H, U)\}$$

tally/1 :

$$tally\text{-}votes(S, H, U) \otimes$$
$$uncounted\text{-}ballot(C, L) \otimes$$
$$hopeful(C, N) \otimes$$
$$!quota(Q) \otimes !(N{+}1 < Q)$$
$$\multimap \{counted\text{-}ballot(C, L) \otimes$$
$$hopeful(C, N{+}1) \otimes$$
$$tally\text{-}votes(S, H, U{-}1)\}$$

tally/2 :

$$tally\text{-}votes(S, H, U) \otimes$$
$$uncounted\text{-}ballot(C, L) \otimes$$
$$hopeful(C, N) \otimes$$
$$!quota(Q) \otimes !(N{+}1 \geq Q) \otimes$$
$$!(S \geq 1)$$
$$\multimap \{counted\text{-}ballot(C, L) \otimes$$
$$!elected(C) \otimes$$
$$tally\text{-}votes(S{-}1, H{-}1, U{-}1)\}$$

tally/3 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [C' \mid L]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{uncounted\text{-}ballot}(C', L) \otimes \\
&\qquad \mathit{tally\text{-}votes}(S, H, U)\}
\end{aligned}$$

tally/4 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, U) \otimes \\
&\mathit{uncounted\text{-}ballot}(C, [\,]) \otimes \\
&(!\mathit{elected}(C) \oplus !\mathit{defeated}(C)) \\
&\quad \multimap \{\mathit{tally\text{-}votes}(S, H, U-1)\}
\end{aligned}$$

tally/5 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, 0) \otimes \\
&!(S < H) \\
&\quad \multimap \{\mathit{defeat\text{-}min}(S, H, 0)\}
\end{aligned}$$

tally/6 :

$$\begin{aligned}
&\mathit{tally\text{-}votes}(S, H, 0) \otimes \\
&!(S \geq H) \\
&\quad \multimap \{!\mathit{elect\text{-}all}\}
\end{aligned}$$

defeat-min/1 :

$$\mathit{defeat\text{-}min}(S, H, M) \otimes \mathit{hopeful}(C, N) \multimap \{\mathit{minimum}(C, N) \otimes \mathit{defeat\text{-}min}(S, H{-}1, M{+}1)\}$$

defeat-min/2 :

$$\mathit{defeat\text{-}min}(S, 0, M) \multimap \{\mathit{defeat\text{-}min}'(S, 0, M)\}$$

defeat-min′/1 :

$$\mathit{defeat\text{-}min}'(S, H, M) \otimes \mathit{minimum}(C_1, N_1) \otimes \mathit{minimum}(C_2, N_2) \otimes !(N_1 \leq N_2) \multimap \{\mathit{minimum}(C_1, N_1) \otimes \mathit{hopeful}(C_2, N_2) \otimes \mathit{defeat\text{-}min}'(S, H{+}1, M{-}1)\}$$

defeat-min′/2 :

$$\mathit{defeat\text{-}min}'(S, H, 1) \otimes \mathit{minimum}(C, N) \multimap \{!\mathit{defeated}(C) \otimes \mathit{transfer}(C, N, S, H, 0)\}$$

transfer/1 :

$$\mathit{transfer}(C, N, S, H, U) \otimes \mathit{counted\text{-}ballot}(C, L) \multimap \{\mathit{uncounted\text{-}ballot}(C, L) \otimes \mathit{transfer}(C, N{-}1, S, H, U{+}1)\}$$

transfer/2 :

$$\mathit{transfer}(C, 0, S, H, U) \multimap \{\mathit{tally\text{-}votes}(S, H, U)\}$$

elect-all/1 :

$$!\mathit{elect\text{-}all} \otimes \mathit{hopeful}(C, N) \multimap \{!\mathit{elected}(C)\}$$

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \rightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \longrightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \rightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \longrightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \longrightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

- Not quite right. This formula allows the voter to keep his card:

$$\vdash \textit{voting-auth-card} \longrightarrow \textit{blank-ballot} \wedge \textit{voting-auth-card}$$

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \longrightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

- Not quite right. This formula allows the voter to keep his card:

$$\vdash \textit{voting-auth-card} \longrightarrow \textit{blank-ballot} \wedge \textit{voting-auth-card}$$

- Worse, it allows ballot stuffing:

$$\vdash \textit{voting-auth-card} \longrightarrow \textit{blank-ballot} \wedge \cdots \wedge \textit{blank-ballot} \wedge \textit{voting-auth-card}$$

Why not use traditional first-order logic?

To motivate the use of linear logic and illustrate its connectives, let's develop a specification of the voter check-in process.

- Let's try to use a traditional logical formula:

$$\textit{voting-auth-card} \longrightarrow \textit{blank-ballot}$$

"If I have an authorization card, then I can have a blank ballot."

- Not quite right. This formula allows the voter to keep his card:

$$\vdash \textit{voting-auth-card} \longrightarrow \textit{blank-ballot} \wedge \textit{voting-auth-card}$$

- Worse, it allows ballot stuffing:

$$\vdash \textit{voting-auth-card} \longrightarrow \textit{blank-ballot} \wedge \cdots \wedge \textit{blank-ballot} \wedge \textit{voting-auth-card}$$

Solution: Use linear logic!