Lecture Notes
Logical Frameworks
The Art of Representation
VSTA 2012
Saarbrücken

Carsten Schürmann

September 3, 2012

Introduction

During these lectures I will introduce you to the world of logical frameworks through logic. When I gave these lectures ten years ago, the state of the art was much different than today. Intuitionistic logic formed the foundation of the logical framework LF [?], that was used for example in the Twelf theorem prover. LF was not designed to reason about programming languages and logics per se, instead it was intended to serve as a meta-language for representing complex data. LF is dependently typed, which means that we can define type families that are indexed by other objects in the type theory, for example, judgments that express that a list has n elements, or a derivability judgment for first-order classical logic.

LF surpasses the simply typed λ-calculus in expressiveness. This means that its type structure is so rich that even for complex operational semantics and logics, it is possible to show that they are adequate, which means that there exists a bijection between canonical forms (i.e. β-reduced, η-expanded) of the type theory, and derivations in the object system. What made LF really popular was its ability to capture substitutions and provided substitution principles and theorems for free. This ability is commonly referred to as *higher-order abstract syntax*. In a nut-shell, the Twelf was the first theorem prover that supported reasoning about encodings in LF. Its reasoning engine implements the induction principle derived from the inductive definition of canonical forms of LF, and it circumvented the need for proving substitution lemmas explicitly.

When Frank Pfenning and I built Twelf in 1998, it became quickly evident clear that it would excel on certain kind of theories, namely those that could be nicely represented in LF. There were many examples of such, for example, Andrew Appel's research on foundational proof carrying code, Appel and Felten's work on proof carrying authentication, Crary's work on formalizing a proof for typed assembly language, and the complete formalization of the type preservation theorem for SML. As the result of a humble self evaluation, I would say that Twelf pushed the bar of what was possible. It allowed users to go deep and spared them from boring boiler plate work regarding the properties of substitutions.

There are many mathematical problems, however, that do not fit so well into the philosophy of the Twelf proof assistant. For some object systems Twelf is not expressive enough. For example it is not easily possible in Twelf to reason about languages that are defined with non-standard notion of substitutions, for example those one obtains when reasoning in linear or affine type theories. If the user is interested in studying substitutions in their own right, Twelf may not be the right tool either, just because the built-in free notion of substitution is defined by LF, and variables cannot be compared for equality.

In the concurrent age, where laptop processors consist of multiple cores, we need to be concerned with more expressive logical frameworks, that should be used for representation and provide a foundation for reasoning.

This is why for this summer school, we do not concentrate on LF but focus on linear logic as the foundation for a logical framework that can be used to represent concurrent traces, graphs, protocols, etc. These notes are organized in four lectures. The first lecture is on the judgmental reconstruction of linear logic. We begin from first principles, define two judgments: A is an ephemeral statement that must be used exactly once, and A is a persistant resources that may be used as often as necessary. In the second lecture, we complete the discussion linear logic and prove expansion of initiality (that corresponds to η-expansion), and cut-elimination (that corresponds to β-reduction). We will not discuss all connectives, but leave further investigations to the interested reader. Linear logic, will not only be a system in which represent derivations but computations as well. In logic, computation correspond to proof search, and in order to keep proof search tractable, we discuss inversion and chaining in the third lecture. In the fourth and final lecture, we show on particular way our formulation of linear logic can be turned into a type theory. The type theory is the concurrent logical framework CLF, its implementation is called the Celf system.

Lecture 1: Judgmental Reconstruction

The running example through this course is that of voting protocols. We will show that the logical framework that we develop here is an ideal candidate to model a domain specific programming language to express an election. Let's look at a very difficult example first, the voting protocol that is called single transferable vote (STV). The protocol is used in real elections, for example, small elections, such as trustee election of the CADE conference or other professional organizations, but also in real big elections, such as parliament elections in Ireland, and Australia.

In STV, each voter casts a ballot that lists candidates in order of the voter's preference. To be elected, a candidate must reach a threshold, or quota, of votes. For the purposes of this paper, the particular choice of quota is arbitrary. Because it is commonly used in practice, we choose the Droop quota,

$$\text{quota} = \frac{\#\text{ballots}}{\#\text{seats} + 1} + 1 \ ,$$

however any quota could easily be substituted. Once the quota is computed, the ballots are counted and the following rules are repeated until all open seats are filled.

1. If a candidate has enough votes to meet the quota, she is declared elected. Any surplus votes for this candidate are transferred.

2. If all ballots have been assigned to candidates and no candidate meets the quota, then the candidate with the fewest votes is eliminated and her votes are transferred. If several candidates tie for the fewest votes, one is eliminated at random.

3. When a vote is transferred, it is assigned to the hopeful candidate with the next highest preference listed on that ballot. That is, candidates that are already elected or defeated do not receive transferred votes.

4. If, at any point, there are at least as many open seats as hopeful candidates remaining, then all remaining hopefuls become elected.

We will come back to this little example in the last lecture. For now, just noticed, that there is an algorithm hidden within this description. Its definition is not very clear. When we try to express this algorithm in first-order logic, we immediate are confronted with the problem to formalize verbs like *to declare someone elected*, *to declare someone defeated*, *to assign a vote to a candidate*, or *to transfer a vote from one candidate to another*. How shall we model the status of a ballot? Could it be a proposition in intuitionistic logic?

We propose to consider a ballot as an ephemeral resource that must be counted once and exactly once. All we need is to construct a logic that can handle ephemeral resources and ensures their proper usage pattern.

Ephemeral resources are not specific to voting. They can be observed eeverywhere: Messages that are being send over a wire, memory cells that may be updated, destinations in programming, credentials that users may use exactly one time to gain access to another resource, tokens in petri nets. If we look around, we see such ephemeral resources popping up everywhere, warranting a deeper investigation of a phenomenon that has many applications beyond voting.

We will begin to build a logic for ephemeral resources, using a technique that has been dubbed judgmental reconstruction [**?**, **?**]. In Martin-Löf's paper on the meaning of logical constants, he characterizes the basics of a logic system into judgments and evidence. A judgment is a something that can be true, and the evidence provides the argument why it is true. Common examples of judgments include $e \hookrightarrow v$ (e evaluates to v), $e : \tau$ (expression e has type v), A pers (formula is derivable in the logic $\Longrightarrow$), but also A eph meaning that A is an ephemeral resource that may be used exactly once.

Under *evidence* we understand finite trees of small justification steps. Each justification step is an inference rules, with multiples premisses $\mathcal{J}_1$ ... $\mathcal{J}_n$ and one conclusion $\mathcal{J}$.

$$\frac{\mathcal{J}_1 \quad \dots \quad \mathcal{J}_n}{J}\,\mathit{name}$$

Martin-Löf makes a distinction between judgments and *hypothetical judgments.* A hypothetical judgment is one that may be introduced as a hypothesis and such. We will in a moment encounter an example of a hypothetical judgment.

The logic that we will reconstruct for the judgment of ephemeral resources coincides with that of linear logic. It was introduced by Jean-Yves Girard [**?**]. Ephemeral resources must be consumed (exactly once) in a valid proof.

Ephemeral Judgments In our logic resources may be constructed from other resources. This is the intention and meaning of an *ephemeral judgments.* The ephemeral judgment is also hypothetical: If A eph is constructed using each ephemeral resource among A_1 $\mathsf{eph} \dots A_n$ eph exactly once, we write

$$A_1\ \mathsf{eph}, \dots, A_n\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}$$

The list of linear resources to the left of the $\Longrightarrow$ symbol enjoys among the three structural properties only exchange (but neither weakening nor contraction) and will be abbreviated in the remainder of this dpaper by the aforementioned Δ. In the remainder of the paper we refer to A_i eph also as a *linear assumption* and to A_i as a *resource.*

$$\frac{}{A\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}}\,\mathsf{ax}$$

To present the meaning of this judgment via various connectives, we will now develop a specification of voter check-in at a polling place. Prior to election day, each voter receives a voting authorization card in the mail. To check in at her designated polling place on election day, the voter exchanges her voting authorization card for a blank ballot form. Because each voter receives only one authorization card, the card thus helps prevent ballot stuffing.

In traditional logic, one might try to specify this check-in process by taking as an axiom the formula

$$\mathit{voting\text{-}auth\text{-}card} \to \mathit{blank\text{-}ballot} \quad :$$

if a voter has a voting authorization card, then she may have a blank ballot form. However, this specification would allow proofs of such nonsense as

$\mathit{voting\text{-}auth\text{-}card} \rightarrow \mathit{blank\text{-}ballot} \wedge \mathit{voting\text{-}auth\text{-}card}$: if a voter has a voting authorization card, she can receive a blank ballot and keep her authorization card. By iterating this proof, one can show that, under this specification, ballot stuffing is possible: $\mathit{voting\text{-}auth\text{-}card} \rightarrow \mathit{blank\text{-}ballot} \wedge \cdots \wedge \mathit{blank\text{-}ballot} \wedge \mathit{voting\text{-}auth\text{-}card}$. Therefore, this specification of the check-in procedure is clearly unsound.

Linear Implication, $\multimap$. The problem is one of expressivity—traditional implication does not express that the check-in process *consumes* the voter's authorization card. But, as a logic of resources, linear logic provides just the right expressive power. It includes the linear implication formula $A \multimap B$, which, like the traditional implication, is a procedure for producing resource B if given A; unlike the traditional implication, however, this procedure consumes resource A as part of the production.

Thus, a sound specification of voter check-in is given by taking as an axiom the linear logical formula

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

the check-in process consumes the voter's authorization card and gives her a blank ballot in exchange.

The connective $\multimap$ is defined by the following rules.

$$\frac{(\Delta, A \;\mathsf{eph}) \Longrightarrow B \;\mathsf{eph}}{\Delta \Longrightarrow A \multimap B \;\mathsf{eph}} \multimap \mathsf{R} \qquad \frac{\Delta_1 \Longrightarrow A \;\mathsf{eph} \quad \Delta_2, B \;\mathsf{eph} \Longrightarrow C \;\mathsf{eph}}{(\Delta_1, \Delta_2, A \multimap B \;\mathsf{eph}) \Longrightarrow C \;\mathsf{eph}} \multimap \mathsf{L}$$

This is the core of a *sequent calculus* formulation for linear logic. The letter R and L tell us if it is a right or a left rule, meaning that the principal formula formed by $\multimap$ either occurs on the right (as the conclusion) or on the left within the context of assumptions.

Alternatively, we could have also written the $\multimap \mathsf{R}$ as an introduction rule $\multimap \mathsf{I}$ in natural deduction formulation. To emphasize the difference to the sequent formulation we write $\vdash$ for the judgment of derivability in natural deduction:

$$\frac{(\Delta, A \;\mathsf{eph}) \vdash B \;\mathsf{eph}}{\Delta \vdash A \multimap B \;\mathsf{eph}} \multimap \mathsf{I}$$

The natural deduction formulation of the $\multimap \mathsf{L}$ is the following elimination rule $\multimap \mathsf{E}$:

$$\frac{\Delta_1 \vdash A \multimap B \;\mathsf{eph} \quad \Delta_2 \vdash B \;\mathsf{eph}}{(\Delta_1, \Delta_2) \vdash B \;\mathsf{eph}} \multimap \mathsf{E}.$$

Natural deduction formulations of logics are traditionally easier to understand and comprehend than their sequent counterparts. Formulating theorems and proofs, however is much easier in the sequent formulation. Both formulations are equivalent in the sense of provability.

Whenever, we introduce a new connective, we are interested in two properties, initiality expansion, and cut-elimination. Initiality expansion corresponds roughly speaking to η-expansion: Assuming that $A\ \mathsf{eph}$ is given, we can take it apart and reconstruct it. The only rule we may rely on is that the axiom rules hold for atoms:

$$\frac{}{P\ \mathsf{eph} \Longrightarrow P\ \mathsf{eph}}\ \mathit{pax}$$

Cut-admissibility corresponds to β-reduction: If we have a proof of $A\ \mathsf{eph}$ and a proof of $C\ \mathsf{eph}$ hypothetical in $A\ \mathsf{eph}$, then we can cut out the intermediate $A\ \mathsf{eph}$ and prove $C\ \mathsf{eph}$ directly. Pfenning often refers to those three properties of harmony: They somehow guarantee that the left and the right rules for a connectives fit to each other, the left rule does not loose information and non the right rule does not introduce anything phony.

Theorem 1 (Initiality-expansion) *If* $A\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}$ *with the connective specific axiom rules then* $A\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}$ *with the* ax *rule.*

Theorem 2 (Cut-admissibility) *If* $\Delta_1 \Longrightarrow A\ \mathsf{eph}$ *and* $(\Delta_2, A\ \mathsf{eph}) \Longrightarrow B\ \mathsf{eph}$ *then* $(\Delta_1, \Delta_2) \Longrightarrow B\ \mathsf{eph}$.

We will give the proofs of thee important properties in the second lecture.

Simultaneous Conjunction, $\otimes$, and Its Unit, 1. Now suppose that voters are also required to present a photo ID during check-in. The specification will have the same basic structure: 'a voting authorization card and a photo ID' $\multimap$ *blank-ballot*. But how can we express the 'a voting authorization card *and* a photo ID' resource as a formula of linear logic?

Fortunately, linear logic provides a simultaneous conjunction, $A \otimes B$ (read 'both resources A and B'). Thus, a specification of the revised check-in process can be given by the formula

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

when a voter gives a voting authorization card and a photo ID, she receives a blank ballot form in exchange. (Note that $\otimes$ binds more tightly than $\multimap$.)

The connective $\multimap$ is defined by the following rules.

$$\frac{\Delta_1 \Longrightarrow A\ \mathsf{eph} \quad \Delta_2 \Longrightarrow B\ \mathsf{eph}}{\Delta \Longrightarrow A \otimes B\ \mathsf{eph}}\ \otimes\mathsf{R}$$

$$\frac{\Delta, A\ \mathsf{eph}, B\ \mathsf{eph} \Longrightarrow C\ \mathsf{eph}}{(\Delta, A \otimes B\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ \otimes\mathsf{L}$$

Linear logic also includes a unit for simultaneous conjunction, $\mathbf{1}$ (read 'nothing'), which represents the empty collection of resources. The proposition $\mathbf{1}$ is primarily used in the idiom $A \multimap \mathbf{1}$, which consumes resource A and produces nothing in return. 1 is the unit of the tensor, and the rules can be easily derived from the tensor rules.

$$\frac{}{\cdot \Longrightarrow \mathbf{1}\ \mathsf{eph}}\ \mathsf{1R}$$

$$\frac{\Delta \Longrightarrow C\ \mathsf{eph}}{(\Delta, \mathbf{1}\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ \mathsf{1L}$$

Unrestricted Modality, !. The prior specification of the check-in process, *voting-auth-card* $\otimes$ *photo-ID* $\multimap$ *blank-ballot*, is not fully satisfactory, however. Because *photo-ID* is treated as a resource and linear implication (which consumes the resources it is given) is used, this axiom specifies a check-in process in which voters must relinquish their photo IDs to vote! This is not the intent; voters should always retain their photo IDs. And so, at first glance, *photo-ID* does not appear to fit into the resource discipline of linear logic.

However, the unrestricted modality, $!A$, of linear logic provides a way out. The proposition $!A$ is a version of A that is not subject to the resource discipline—an assumption $!A$ can be used an unlimited number of times (including none). Alternatively, one may think of $!A$ as stating that A is a fact that will remain true regardless of how the system evolves.

[Using the ! modality, the revised specification can therefore be given by

$$\textit{voting-auth-card} \otimes\, !\textit{photo-ID} \multimap \textit{blank-ballot} \quad :$$

when a voter gives an authorization card and shows a photo ID, she receives a blank ballot form. (Note that ! binds more tightly than $\otimes$ and $\multimap$.)

Our judgment that we have been so far is not general enough to capture resources that are persistant as the the ! suggests. We will need to introduce a new judgment, A pers if A is a persistant hypothesis and extend our logic, by a context Γ that stands for the persistant hypotheses $x_1 : A_1\ \mathsf{pers}, \ldots, x_n : A_n\ \mathsf{pers}$. In contrast the ephemeral context Δ, Γ enjoys all structural properties, i.e. weakening, exchange, and contraction. The meaning of persistance is best summarized by the following rule

$$\frac{\Gamma; \cdot \Longrightarrow A\ \mathsf{eph}}{\Gamma \Longrightarrow A\ \mathsf{pers}}\ \mathit{pers}$$

which we don't really need since we will just use the premiss in any rule that would mention persistent conclusion otherwise.

$$\frac{\Gamma, A\ \mathsf{pers}; \Delta, A\ \mathsf{eph} \vdash C\ \mathsf{eph}}{\Gamma, A\ \mathsf{pers}; \Delta \vdash C\ \mathsf{eph}}\ \mathit{copy}$$

This rules does everything we need it to do. It allows us to create arbitrary many instances from persistant resources. We need to go back and extend all inference that we have introduced so far with a leading Γ;. It is easy to see that

Theorem 3 $\Gamma, A\ \mathsf{pers}; \cdot \Longrightarrow A\ \mathsf{eph}$

Proof: Immediate by *ax* followed by *copy*. □

Therefore, we don't need to consider an additional axiom rule and the initiality expansion theorem from above should not be affected. We restate it, this time with the context for persistant assumptions added.

Theorem 4 (Initiality-expansion) *If* $\Gamma; A\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}$ *with the connective specific axiom rules then* $\Gamma; A\ \mathsf{eph} \Longrightarrow A\ \mathsf{eph}$ *with the* ax *rule.*

We will check all of this in detail in Lecture 2. For cut, we are less lucky, because we will need to consider cutting out a persistant assumption as the following reformulation of the the cut admissibility theorem shows:

We will consider an extension of the cut theorem above though:

Theorem 5 (Cut-admissibility)

1. *If* $\Gamma; \Delta_1 \Longrightarrow A\ \mathsf{eph}$ *and* $\Gamma; (\Delta_2, A\ \mathsf{eph}) \Longrightarrow B\ \mathsf{eph}$ *then* $\Gamma; (\Delta_1, \Delta_2) \Longrightarrow B\ \mathsf{eph}$.
2. *If* $\Gamma; \cdot \Longrightarrow A\ \mathsf{eph}$ *and* $(\Gamma, A\ \mathsf{pers}); \Delta \Longrightarrow B\ \mathsf{eph}$ *then* $\Gamma; \Delta \Longrightarrow B\ \mathsf{eph}$.

Next we turn to the rules defining the modal connective !, which will fall in place quite naturally.

$$\frac{\Gamma; \cdot \Longrightarrow A\ \mathsf{eph}}{\Gamma; \cdot \Longrightarrow !A\ \mathsf{eph}}\ !\mathsf{R}$$

$$\frac{(\Gamma, A\ \mathsf{pers}); \Delta \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, !A\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ !\mathsf{L}$$

Universal Quantification, $\forall x{:}\tau$. Strictly speaking, the current specification of voter check-in does not capture the requirement that the name on the authorization card must match the name on the photo ID.

This problem can be resolved using universal quantification. In linear logic, multi-sorted universal quantification, $\forall x{:}\tau.\, A$, behaves just as in traditional logic. In particular, the members of the domain of quantification are not subject to a resource discipline. Thus, the specification may be revised to

$$\forall v{:}\mathsf{voter}.\ \big(\textit{voting-auth-card}(v) \otimes !\textit{photo-ID}(v) \multimap \textit{blank-ballot}\big) \quad :$$

when a voter v gives *her* authorization card and shows *her* photo ID, she receives a blank ballot form.

The left and right rules defining universal quantification are as follows:

$$\frac{\Gamma; \Delta \Longrightarrow A[a/x]\ \mathsf{eph}}{\Gamma; \Delta \Longrightarrow \forall x{:}\tau.A\ \mathsf{eph}}\ \forall\mathsf{R}^(a : \tau)$$

$$\frac{\Gamma; (\Delta, A[t/x]\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, \forall x{:}\tau.A\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ \forall\mathsf{L}, \text{where } t \text{ has sort } \tau$$

$$\frac{}{\Gamma; P\ \mathsf{eph} \Longrightarrow P\ \mathsf{eph}}\ \mathit{pax}$$

$$\frac{\Gamma; (\Delta, A\ \mathsf{eph}) \Longrightarrow B\ \mathsf{eph}}{\Gamma; \Delta \Longrightarrow A \multimap B\ \mathsf{eph}} \multimap \mathsf{R} \qquad \frac{\Gamma; \Delta_1 \Longrightarrow A\ \mathsf{eph} \quad \Gamma; \Delta_2, B\ \mathsf{eph} \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta_1, \Delta_2, A \multimap B\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}} \multimap \mathsf{L}$$

$$\frac{\Gamma; \Delta_1 \Longrightarrow A\ \mathsf{eph} \quad \Gamma; \Delta_2 \Longrightarrow B\ \mathsf{eph}}{\Gamma; \Delta \Longrightarrow A \otimes B\ \mathsf{eph}} \otimes\mathsf{R} \qquad \frac{\Gamma; (\Delta, A\ \mathsf{eph}, B\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, A \otimes B\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}} \otimes\mathsf{L}$$

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{1}\ \mathsf{eph}}\ \mathbf{1}\mathsf{R} \qquad \frac{\Gamma; \Delta \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, \mathbf{1}\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ \mathbf{1}\mathsf{L}$$

$$\frac{\Gamma, A\ \mathsf{pers}; \Delta, A\ \mathsf{eph} \vdash C\ \mathsf{eph}}{\Gamma, A\ \mathsf{pers}; \Delta \vdash C\ \mathsf{eph}}\ \mathit{copy}$$

$$\frac{\Gamma; \cdot \Longrightarrow A\ \mathsf{eph}}{\Gamma; \cdot \Longrightarrow\ !A\ \mathsf{eph}}\ !\mathsf{R} \qquad \frac{(\Gamma, A\ \mathsf{pers}); \Delta \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, !A\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ !\mathsf{L}$$

$$\frac{\Gamma; \Delta \Longrightarrow A[a/x]\ \mathsf{eph}}{\Gamma; \Delta \Longrightarrow \forall x{:}\tau.A\ \mathsf{eph}}\ \forall\mathsf{R}^{(}a : \tau)$$

$$\frac{\Gamma; (\Delta, A[t/x]\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}{\Gamma; (\Delta, \forall x{:}\tau.A\ \mathsf{eph}) \Longrightarrow C\ \mathsf{eph}}\ \forall\mathsf{L}, \text{where } t \text{ has sort } \tau$$

Figure 1: Sequent Calculus for Intuitionistic Linear Logic

With the appearance of a substitution, we must define an appropriate substitution lemma:

Lemma 6 (Substitution) *If* $\Gamma(a : \tau); \Delta(a : \tau) \vdash C(a : \tau)\ \mathsf{eph}$ *and* t *has sort* τ*, then* $\Gamma(t); \Delta(t) \vdash C(t)\ \mathsf{eph}$.

Figure 1 summarizes all our rules that define the meaning of the connectives purely in terms of availability.

Lecture 2: Initiality and Cut-Admissibility

In this lecture, we will prove Theorems 4 and 5 and Lemma 6. We discuss several cases to illustrate that the theorems are really true.

Example 7 (First past the post) We give an example of the specification of the first past the post voting protocol in linear logic. Assuming that there are n ballots and k candidates, this protocol will determine the winner of the election. We first give the rules and then explain them, assuming that there are three three sorts, nat denoting natural numbers and cand denoting candidates.

1. $\forall N : \mathsf{nat}. \forall C : \mathsf{cand}. \forall M : \mathsf{nat}.$

   $\mathsf{count}(N) \otimes \mathsf{ballot}(C) \otimes \mathsf{hopeful}(C, M) \multimap \mathsf{hopeful}(C, M+1) \otimes \mathsf{count}(N-1)$

2. $\forall C_1 : \mathsf{cand}. \forall M_1 : \mathsf{nat}. \forall C_2 : \mathsf{cand}. \forall M_2 : \mathsf{nat}.$

   $\mathsf{count}(0) \otimes \mathsf{hopeful}(C_1, M_1) \otimes \mathsf{hopeful}(C_2, M_2) \otimes !(M_1 < M2)$

   $\multimap \mathsf{hopeful}(C_2, M_2) \otimes \mathsf{count}(0) \otimes !\mathsf{defeated}(C_1)$

3. $\forall C : \mathsf{cand}. \forall M : \mathsf{nat}$

   $\mathsf{count}(0) \otimes \mathsf{hopeful}(C, M) \multimap !\mathsf{elected}(C)$

A candidate is elected if and only if the following sequent is provable. Let Γ consist of the three rules.

$$\begin{aligned}\Gamma; \cdot \Longrightarrow\ & \mathsf{ballot}(C_1) \otimes \ldots \otimes \mathsf{ballot}(C_n) \\ & \otimes \mathsf{hopeful}(C_1, 0) \otimes \ldots \otimes \mathsf{hopeful}(C_k, 0) \otimes \mathsf{count}(n) \\ & \multimap !\mathsf{elected}(C)\end{aligned}$$

to figure out who is elected, we need to run a theorem prover. And thus we need to determine a good strategy on how to apply the rules. This is what we are going to do in the next lecture.