Reasoning about Deductions in Linear Logic

Frank Pfenning
joint work with Iliano Cervesato and Carsten Schürmann

CADE-15, Linday, Germany
July 6, 1998

Work in Progress!

1. Linear Logic
2. Reasoning About Deductions
3. Linear Type Theory
4. A Meta-Logic
5. Preliminary Experiments
6. Conclusion

Introduction

> We can look at the current field of problem solving by computers as a series of ideas about how to present a problem. If a problem can be cast into one of these representations in a natural way, then it is possible to manipulate it and stand some chance of solving it.

Allen Newell, *Limitations of the Current Stock of Ideas for Problem Solving*, 1965

Universality of Predicate Calculus?

- Classical first-order logic
- Intuitionistic logic
- Temporal, modal logics
- Dynamic logic, Hoare logic
- Arithmetic
- Theory of inductive definitions
- Higher-order logic
- Constructive type theory
- Linear logic
- Linear type theory

Linear Logic as a Logic of State

Suitable description language for

- (Imperative) programming languages
  [Chirimar'95][Cervesato & Pf'96]
- (Abstract) machines [Chirimar'95][Plesko]
- Concurrent systems
  [Girard'89][Lafont'90][Gay'94][Kobayashi'96]
- Protocols [Cervesato & Schürmann]
- Games
  [Lafont & Streicher'91][Blass'92][Abramsky&Jagadeesan'94]
- Planning [Bibel'86]

Legal transition sequences $\Longleftrightarrow$ deductions
Computations $\Longleftrightarrow$ deductions

Reasoning about Deductions

Often, we need to reason *about* computations or transition sequences

- (Imperative) programming languages:
  Type safety
- (Abstract) machines:
  Memory safety (PCC)
- Concurrent systems:
  Deadlock, Bisimulation
- Protocols:
  Security properties
- Games:
  Strategies
- Planning:
  Plan transformation

Talk Outline

1. Linear Logic
2. Reasoning in Linear Logic
3. Linear Type Theory
4. Reasoning about Deductions
5. Preliminary Results
6. Conclusion

Linear Hypotheses as Resources

Basic Judgment

$$\underbrace{(B_1, \ldots, B_n)}_{\Gamma}; \underbrace{(A_1, \ldots, A_m)}_{\Delta} \vdash A$$

Hypothesis Rules

$$\frac{}{\Gamma; A \vdash A} \qquad \frac{}{(\Gamma, A); \cdot \vdash A}$$

- Unrestricted hypotheses $\Gamma \iff$ "logical" assumptions
- Linear hypotheses $\Delta \iff$ resources ("state")
- Conservative extension of (intuitionistic) logic

Linear Implication and Simultaneous Conjunction

$A \multimap B$ *With resource A we can achieve B*

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

$$\frac{\Gamma; \Delta_1 \vdash A \multimap B \qquad \Gamma; \Delta_2 \vdash A}{\Gamma; (\Delta_1, \Delta_2) \vdash B} \multimap \mathsf{E}$$

$A \otimes B$ *We can achieve A and B simultaneously*

$$\frac{\Gamma; \Delta_1 \vdash A \qquad \Gamma; \Delta_2 \vdash B}{\Gamma; (\Delta_1, \Delta_2) \vdash A \otimes B} \otimes\mathsf{I}$$

$$\frac{\Gamma; \Delta \vdash A \otimes B \qquad \Gamma; (\Delta', A, B) \vdash C}{\Gamma; (\Delta, \Delta') \vdash C} \otimes\mathsf{E}$$

Alternative Conjunction and Implication

$A \mathbin{\&} B$ *We can achieve A and B alternatively*

$$\frac{\Gamma; \Delta \vdash A \qquad \Gamma; \Delta \vdash B}{\Gamma; \Delta \vdash A \mathbin{\&} B} \mathbin{\&}\mathsf{I}$$

$$\frac{\Gamma; \Delta \vdash A \mathbin{\&} B}{\Gamma; \Delta \vdash A} \mathbin{\&}\mathsf{E}_1 \qquad \frac{\Gamma; \Delta \vdash A \mathbin{\&} B}{\Gamma; \Delta \vdash B} \mathbin{\&}\mathsf{E}_2$$

$A \rightarrow B$ *With A as unrestricted resource, we can achieve B*

$$\frac{(\Gamma, A); \Delta \vdash B}{\Gamma; \Delta \vdash A \rightarrow B} \rightarrow\mathsf{I}$$

$$\frac{\Gamma; \Delta \vdash A \rightarrow B \qquad \Gamma; \cdot \vdash A}{\Gamma; \Delta \vdash B} \rightarrow\mathsf{E}$$

Unit and Truth

1 *We have no resources*

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}}\ \mathbf{1}\mathsf{I}$$

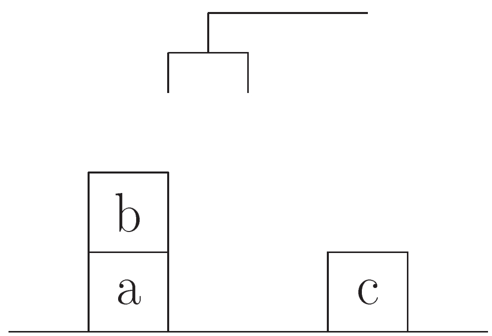
$$\frac{\Gamma; \Delta \vdash \mathbf{1} \qquad \Gamma; \Delta' \vdash C}{\Gamma; (\Delta, \Delta') \vdash C}\ \mathsf{1E}$$

$\top$ *Consumes all resources*

$$\frac{}{\Gamma; \Delta \vdash \top}\ \top\mathsf{I}$$

no $\top\mathsf{E}$ *rule*

Example: Blocks World



Primitive propositions:

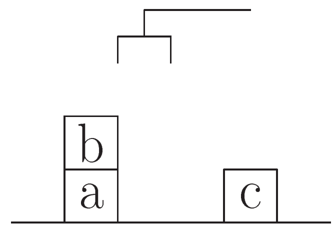
on(x, y)	block x is on block y
tb(x)	block x is on the table
holds(x)	robot hand holds block x
empty	robot hand is empty
clear(x)	top of block x is clear

Problem Representation

$$\Gamma_0; \Delta_0 \vdash A_0$$

- Γ_0 represents legal moves
- Δ_0 represents current state
- A_0 represents goal state
- A deduction corresponds to a solution

Example Problem



$$
\begin{aligned}
\Delta_0 &= \mathrm{tb}(a), \mathrm{on}(b, a), \mathrm{clear}(b),\\
&\quad \mathrm{tb}(c), \mathrm{clear}(c),\\
&\quad \mathrm{empty}.\\
\Gamma_0 &= \forall x.\forall y.\mathrm{empty} \otimes \mathrm{clear}(x) \otimes \mathrm{on}(x, y) \multimap \mathrm{holds}(x) \otimes \mathrm{clear}(y),\\
&\quad \forall x.\mathrm{empty} \otimes \mathrm{clear}(x) \otimes \mathrm{tb}(x) \multimap \mathrm{holds}(x),\\
&\quad \forall x.\forall y.\mathrm{holds}(x) \otimes \mathrm{clear}(y) \multimap \mathrm{empty} \otimes \mathrm{on}(x, y) \otimes \mathrm{clear}(x),\\
&\quad \forall x.\mathrm{holds}(x) \multimap \mathrm{empty} \otimes \mathrm{clear}(x) \otimes \mathrm{tb}(x).\\
A_0 &= \mathrm{on}(a, b) \otimes \top.
\end{aligned}
$$

Some Problem Statements

- Resources $\Delta = P_1, \ldots, P_n \Longleftrightarrow$ state formula $A = P_1 \otimes \cdots \otimes P_n$
- Some questions:
  - Can we achieve B from A?
    Does there exist a deduction $\mathcal{D}$ of $\Gamma_0; \cdot \vdash A \multimap B$?
  - Can we achieve B from A without ever placing block b on the table?
    Does there exist a deduction $\mathcal{D}$ of $\Gamma_0; \cdot \vdash A \multimap B$ such that safe$(\mathcal{D})$?
  - Can we transform the solution to a shorter one?
    For a deduction $\mathcal{D}$ of $\Gamma_0; \cdot \vdash A \multimap B$ does there exists a related $\mathcal{D}'$ of $\Gamma_0; \cdot \vdash A \multimap B$ such that $\mathcal{D}' < \mathcal{D}$?

Example: Imperative Programming Languages

[Chirimar'95] [Cervesato & Pf'96] [Plesko]

$$\Gamma_0; \Delta \vdash \mathrm{exec}(C)$$

$$\begin{array}{rcl}
\Gamma_0 & \Longleftrightarrow & \text{operational semantics} \\
\Delta & \Longleftrightarrow & \text{memory state} \\
C & \Longleftrightarrow & \text{command} \\
\text{Deduction } \mathcal{D} & \Longleftrightarrow & \text{computation} \\
\text{Property of } \mathcal{D} & \Longleftrightarrow & \text{safety condition}
\end{array}$$

- Memory safety of a program C corresponds to a property of all derivations of $\mathrm{exec}(C)$.
- Type safety of the programming language corresponds to a property of all programs C, typing derivations $\mathcal{P}$ and computation derivations $\mathcal{D}$.

Example: Protocols

$$\Gamma_0; \Delta \vdash A$$

$$
\begin{array}{rcl}
\Gamma_0 & \Longleftrightarrow & \text{protocol rules} \\
\Delta & \Longleftrightarrow & \text{state of principles and communication} \\
C & \Longleftrightarrow & \text{goal (e.g., authentication)} \\
\text{Deduction } \mathcal{D} & \Longleftrightarrow & \text{legal computation} \\
\text{Property of } \mathcal{D} & \Longleftrightarrow & \text{security condition}
\end{array}
$$

Example: Concurrent Systems

$$\Gamma_0; \Delta \vdash \mathbf{1}$$

$$\begin{array}{rcl}
\Gamma_0 & \Longleftrightarrow & \text{transition rules} \\
\Delta & \Longleftrightarrow & \text{state of processes and channels} \\
\text{Deduction } \mathcal{D} & \Longleftrightarrow & \text{trace} \\
\text{Property of } \mathcal{D} & \Longleftrightarrow & \text{trace property}
\end{array}$$

- Trace properties correspond to properties of deductions.

Reasoning about Deductions

- Automated deduction answers questions:
  Does there exist a (linear) deduction $\mathcal{D}$ of judgment J?

- Often, we would like to answer questions such as:
  For every deduction $\mathcal{D}$ of J, does there exist a deduction $\mathcal{D}'$ of J^*?

- Traditional approach:
  Does there exist a proof of

  $$\forall \mathsf{J}.\ \forall \mathsf{D}.\ \mathit{ded}(\mathsf{D}, \mathsf{J}) \supset \exists \mathsf{D}'.\mathit{ded}(\mathsf{D}', \mathsf{J}^*)$$

  in a meta-logic?

- This talk:
  Design a suitable meta-logic.

Traditional Logics as Meta-Logics

- Theorem proving methods developed for the logic must be coded on the representation in the meta-logic.
- Coding of judgments and deductions is often awkward when the meta-logic is not expressive enough.
- A decidable property ($\mathcal{D}$ is a derivation of J) is typically mapped to a proposition (ded(D, J)) subject to theorem proving.
- No support for hypothetical, linear hypothetical, or schematic judgments, whose frequently recurring properties must be formulated and proved in the meta-logic.

Goals for a Meta-Theory of Deductions

- Reason directly about deductions.
- Approach general enough for linear logic.
- Inherit theorem proving methods.
- Suitable for automation.
- Natural expression of informal meta-theoretic proofs.

Overview of Approach

- Use (linear) type theory to reify deductions.
  (Linear) logical framework
- Separate meta-logic from logical framework.
  Avoid complications of reflection
- Keep meta-logic simple.
  Exploit expressiveness of logical framework

Reifying Deductions

Problem: many deductions correspond to one computation or transition sequence.

- Approach I: Consider equivalence classes of derivations (*proof nets*).
  - Works well for classical, multiplicative linear logic.
  - Some difficulties for exponential and additives.
  - Is there an implementable theory of proof nets?

- Approach II: Use linear λ-terms to represent deductions (LLF).
  - Works well for fragment of intuitionistic linear logic.
  - Tractable equational theory ($\beta\eta$-conversion).

A Linear Logical Framework (LLF)

Basic Judgment

$$\Gamma; \Delta \vdash_{\Sigma} M : A$$

- Type $A \Longleftrightarrow$ judgment J
- Object M of type $A \Longleftrightarrow$ deduction $\mathcal{D}$ of J
- Signature $\Sigma \Longleftrightarrow$ deductive system

$$\begin{array}{rcl} A & ::= & P \mid \Pi x{:}A_1.A_2 \mid A_1 \to A_2 \\ & & \mid A_1 \multimap A_2 \mid A_1 \,\&\, A_2 \mid \top \end{array}$$

Properties of LLF

- Canonical forms exist and are unique.
- Equality and validity are decidable [Cervesato & Pf'96].
- Expressive enough to directly embed intuitionistic and classical linear logic and examples from this talk.
- Canonical objects are in bijective correspondence with deductions.
- Some "sequentialization" is necessary due to the absence of $\otimes$ (avoids commuting conversions).

Blocks World in LLF

```
block : type.
a : block.
b : block.
c : block.

on : block -> block -> type.            % on x y -- x is on y
tb : block -> type.                     % tb x -- x is on table
clear : block -> type.                  % clear x -- top of x is clear
empty : type.                           % empty -- robot hand is empty
holds : block -> type.                  % holds x -- robot hand holds x

move : type.

pick : empty
       -o on X Y
       -o clear X
       -o (clear Y -o holds X -o move)
       -o move.
```

Example: A Solution (Deduction)

```
win : type.

winm : (tb a -o on b a -o clear b        % b on a on table
          -o tb c -o clear c             % c on table
          -o empty                       % hand empty
          -o (<T> -o on a b -o move)     % win if a on b
          -o move)
        -o win.

% ex1: pick up b, put b on table, pick up a, put a on b
ex1 : win
  = winm ^ ([oa^tb a] [oba^on b a] [cb^clear b]
              [oc^tb c] [cc^clear c] [e^empty]
              [success^move o- on a b o- <T>]
              pick ^ e ^ oba ^ cb
              ^ ([ca^clear a] [hx^holds b]
                   puttb ^ hx
                   ^ ([ob^tb b] [cb^clear b] [e^empty]
                        picktb ^ e ^ oa ^ ca
                        ^ ([ha^holds a]
                             put ^ ha ^ cb
                             ^ ([oab^on a b] [ca^clear a] [e^empty]
                                  success ^ () ^ oab))))).
```

Examples: Safety Condition

```
% A derivation is safe, if block b is never put on the table.

safe : block -> type.
%name safe S.

sfa : safe a.
% b is not safe.
sfc : safe c.

okm : move -> type.
%name okm K.

okpick : ({cy:clear Y} {hx:holds X} okm (M ^ cy ^ hx))
          -> okm (pick ^ E ^ O ^ C ^ M).

okpicktb : ({hx:holds X} okm (M ^ hx))
            -> okm (picktb ^ E ^ O ^ C ^ M).

okput : ({oxy:on X Y} {cx:clear X} {e:empty} okm (M ^ oxy ^ cx ^ e))
         -> okm (put ^ H ^ C ^ M).

okputtb : ({ox:tb X} {cx:clear X} {e:empty} okm (M ^ ox ^ cx ^ e))
           -> safe X
           -> okm (puttb ^ H ^ M).
```

Searching for Deductions in LLF

- Theorem proving in linear logic
  [Tammet'94] [Lincoln & Shankar'94] [Harland & Pym'97]
- Constraint logic programming in LLF
  [Cervesato'96] [Cervesato, Hodas & Pf'96]
- LLF theorem proving (PTTP-style)
  [Schürmann & Pf'98]
- Model checking?

The Architecture

Meta-Logic	$\mathcal{M}_\omega$
Logical Framework	LLF
Problem Domain	

Minimal Requirements for Meta-Logic

- Existential quantifiers over LLF objects $\exists x{:}A.F$.
- Truth $\top$.

Theorem Proving: $\exists x{:}A.\top$ searches for a deduction M of A.

- Universal quantifiers over LLF objects $\forall x{:}A.F$.
- Conjunction $F_1 \wedge F_2$ (for simultaneous induction).

Others ($M < M'$, $M = M'$) may be possible, but not necessary in many examples.

Meta-Logic Judgment in M_ω

$$(\Gamma; \cdot); \Psi \vdash_\Sigma F$$

$(\Gamma; \cdot)$ is a pure LLF context
Ψ contains meta-logical assumptions (lemmas, ind. hyp.)
F is a meta-logical formulas

Some Introduction Rules

$$\frac{}{(\Gamma; \cdot); \Psi \vdash_\Sigma \top}\ \top\mathsf{I}$$

$$\frac{\Gamma; \cdot \vdash^{LLF}_\Sigma M : A \qquad (\Gamma; \cdot); \Psi \vdash_\Sigma [M/x]F}{(\Gamma; \cdot); \Psi \vdash_\Sigma \exists x{:}A.F}\ \exists\mathsf{I}$$

$$\frac{((\Gamma, x{:}A); \cdot); \Psi \vdash_\Sigma F}{(\Gamma; \cdot); \Psi \vdash_\Sigma \forall x{:}A.F}\ \forall\mathsf{I}$$

Splitting

- Splitting handles inversion and proof by cases.
- Central also for induction proofs.
- Inspired by definitional reflection
  [Schroeder-Heister'93] [McDowell & Miller'97]

$$\frac{\Gamma(x) = A \qquad (\Gamma; \cdot); \Psi \vdash_{\Sigma} \text{split } x{:}A \text{ over} F}{(\Gamma; \cdot); \Psi \vdash_{\Sigma} F} \,\mathsf{split}$$

Splitting and Unification

- Consider each constant $c : B$ if it can be the head of an object of type A.
- Check this condition by linear unification.
  [Cervesato & Pfenning'97]
- In practice allow splitting only in finitary cases.
- Cannot restrict to unitary case (patterns) because of linearity constraints.

Induction

- Must handle structural induction.
- Fixed induction schemas are difficult (impossible?) because of schematic, hypothetical, and linear hypothetical judgments.
- Decompose into splitting and well-founded recursion.
- Requires a proof term calculus for meta-logic.

Meta-Logical Judgment Revisited

Revised Judgment

$$(\Gamma; \cdot); \Psi \vdash_{\Sigma} P \in F$$

Recursion Rule

$$\frac{(\Gamma; \cdot); (\Psi, \mathbf{x} \in F) \vdash_{\Sigma} P \in F}{(\Gamma; \cdot); \Psi \vdash_{\Sigma} (\mu\mathbf{x} \in F.P) \in F}\ \mathsf{rec}$$

where $\mu\mathbf{x} \in F.P$ terminates in $\mathbf{x}$.

Proof terms for other rules are straightforward.

Termination

- Applications of recursion are annotated with a termination order.
- Intuitionistic (non-linear) prototype allows lexicographic and simultaneous extensions of a subterm ordering.
- Subterm ordering on higher-order functions is not trivial. [Rohwedder & Pf'96]
- Other termination orders possible. [Kahrs'95] [van de Pol & Schwichtenberg'95] [Lysne & Piris'95]
- Can linearity play a role?

Meta-Logic Summary

- Universal and existential quantification over closed LLF objects.
- Direct search to find witnesses for existentials.
  Inherit theorem proving techniques.
- Induction decomposes into case analysis and well-founded recursion.
- Simplicity possible due to expressive power of LLF.

Soundness of Meta-Logic

- If $(\cdot;\cdot);\cdot \vdash_\Sigma \exists x{:}A.F$ then for some $\cdot;\cdot \vdash_\Sigma M : A$ we have $(\cdot;\cdot);\cdot \vdash [M/x]F$.
- If $(\cdot;\cdot);\cdot \vdash_\Sigma \forall x{:}A.F$ then for all $\cdot;\cdot \vdash_\Sigma M : A$ we have $(\cdot;\cdot);\cdot \vdash_\Sigma [M/x]F$.
- Proof by showing totality of proof terms as functions.
  Completed only for restricted non-linear case
- Is there useful notion of completeness?

Limitations

- Main limitation: restriction to closed objects and fixed signature.
- Required for soundness of splitting.
- Approach I: reify contexts Γ and Δ in the meta-logical judgments. [McDowell'97]
- Approach II: allow "regular" context classes and generalize splitting [ongoing work]
- Example: memory state in programming languages, world state in planning.

Preliminary Results

- Implementation of the linear logical framework, including type reconstruction and linear constraint logic programming interpreter.
  `http://www.stanford.edu/~iliano/LLF`

- Implementation of the logical framework (LF), including type reconstruction, logic programming, meta-logic and simple meta-theorem prover
  [Schürmann & Pfenning, this CADE]
  `http://www.cs.cmu.edu/~twelf/`

- Theory and implementation restricted to $\forall\exists$ formulas.

Twelf Experiments

Experiment	Front	Fill	Split	Rec	Total
Cartesian Closed Categories	0.058	1.000	0.004	0.036	1.099
CPM Completeness	0.900	0.916	0.010	0.117	1.134
Horn LP Soundness	0.112	4.336	0.004	0.049	4.501
Horn LP Completeness	0.137	0.015	0.005	0.039	0.195
Mini-ML Value soundness	0.055	0.016	0.041	0.061	0.172
Mini-ML Type preservation	0.066	0.062	0.521	0.150	0.799
Mini-ML Evaluation/Reduction	0.064	25.397	0.007	0.078	25.546
Hilbert's abstraction theorem	0.111	0.197	0.004	0.010	0.322
Associativity of +	0.026	0.009	0.012	0.016	0.063
Commutativity of +	0.037	0.092	0.609	4.139	4.877

Linux 2.30, SML/NJ 110, Twelf 1.2 on Pentium II (300 Mhz)

Related Work

- Many experiments in reasoning about derivations using "traditional" logics as meta-logics.
  [Shankar'87] [Matthews'94] [Basin & Constable'93] [McKinna & Pollack'93] [Barras & Werner'97] ...
- $FO\lambda^{\Delta\mathbb{N}}$ over hereditary Harrop formulas [McDowell & Miller'97]
  - definitional reflection (splitting)
  - does not reify deductions, no dependent types
  - only natural number induction
  - interactive
  - applies to linear case [McDowell'97]
- RLF linear logical framework [Ishtiaq & Pym'98]

Summary

- Proposed reasoning about deductions as a paradigm in theorem proving.
- Illustrated the value of linearity within this paradigm.
- Sketched a linear logical framework (LLF) for problem representation.
- Proposed a meta-logic for reasoning about deductions in LLF.
- Presented some preliminary results.
- LLF: `http://www.stanford.edu/~iliano/LLF`
- Twelf: `http://www.cs.cmu.edu/~twelf/`