A Concurrent Logical Framework

Iliano Cervesato **iliano@itd.nrl.navy.mil**

ITT Industries, inc @ NRL Washington, DC

http://www.cs.stanford.edu/~iliano

(Joint work with Frank Pfenning, David Walker, and Kevin Watkins)

CS Department, UMBC February 27-28 2003

CLF

- Where it comes from
  - Logical Frameworks
  - The LF approach
- What it is
  - True concurrency
  - Monadic encapsulation
  - A canonical approach
- What's next?

All about Logical Frameworks

Represent and reason about object systems

- Languages, logics, …
  - Often semi-formalized as deductive systems
  - Reasoning often informal
- Benefits
  - Formal specification of object system
  - Automate verification of reasoning arguments
  - Feed back into other tools
    - Theorem provers, PCC, …

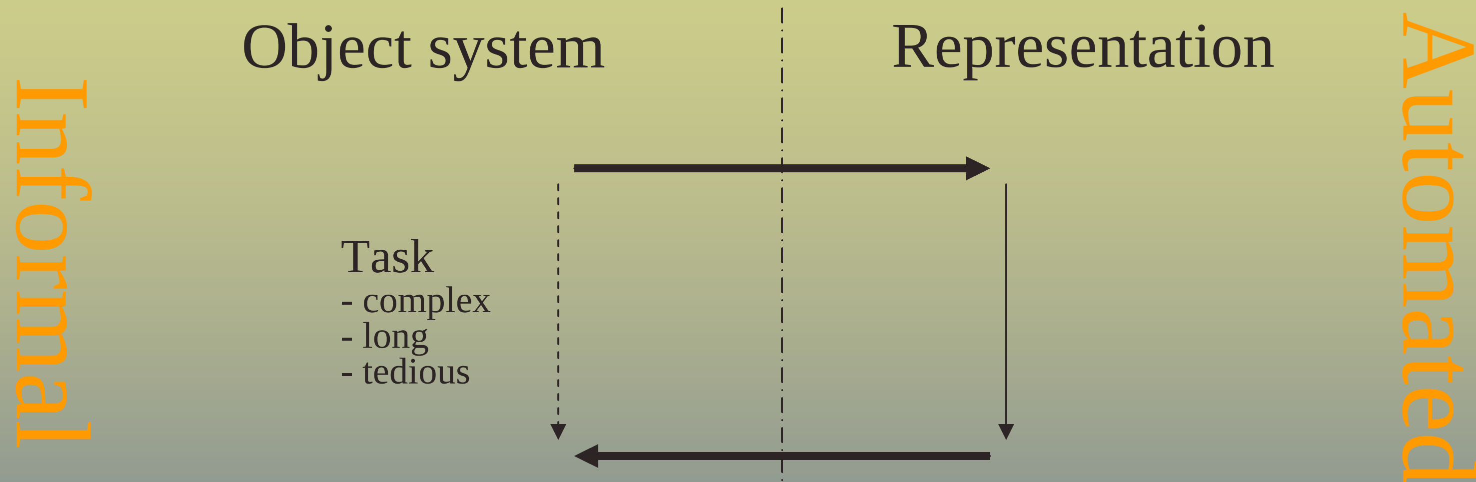
The LF Way

Identify fundamental mechanisms and build them into the framework (soundly!)

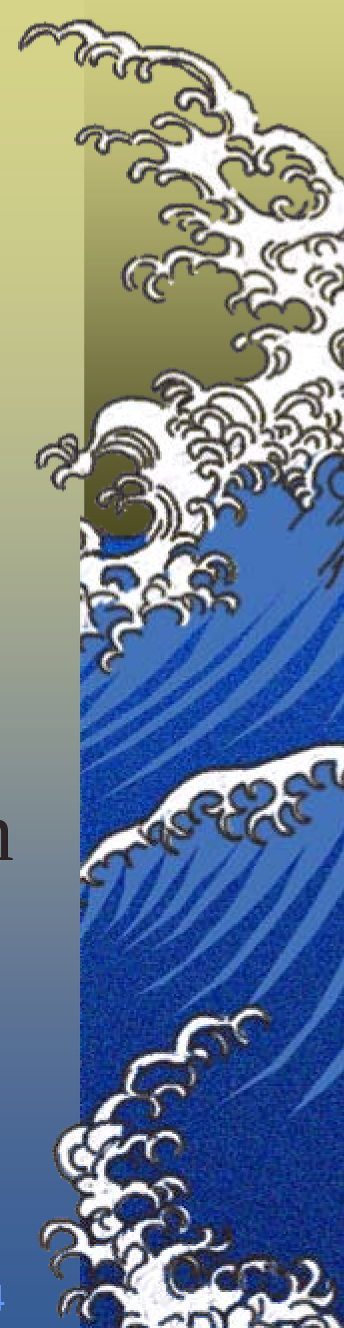
- done (right) once and for all instead of each time

- Modular constructions: [Σ-Algebras]
  - *app f a*
- Variable binding, α-renaming, substitution [LF]
  - $\lambda x.\ x+1$
- Disposable, updateable cell [LLF]
  - $\lambda\hat{}s'.\ f\ \hat{}\ s$
- True concurrency [CLF]

It's all about *Adequacy*



- Adequacy: correctness of the transcription
- LF: make adequacy as simple as possible



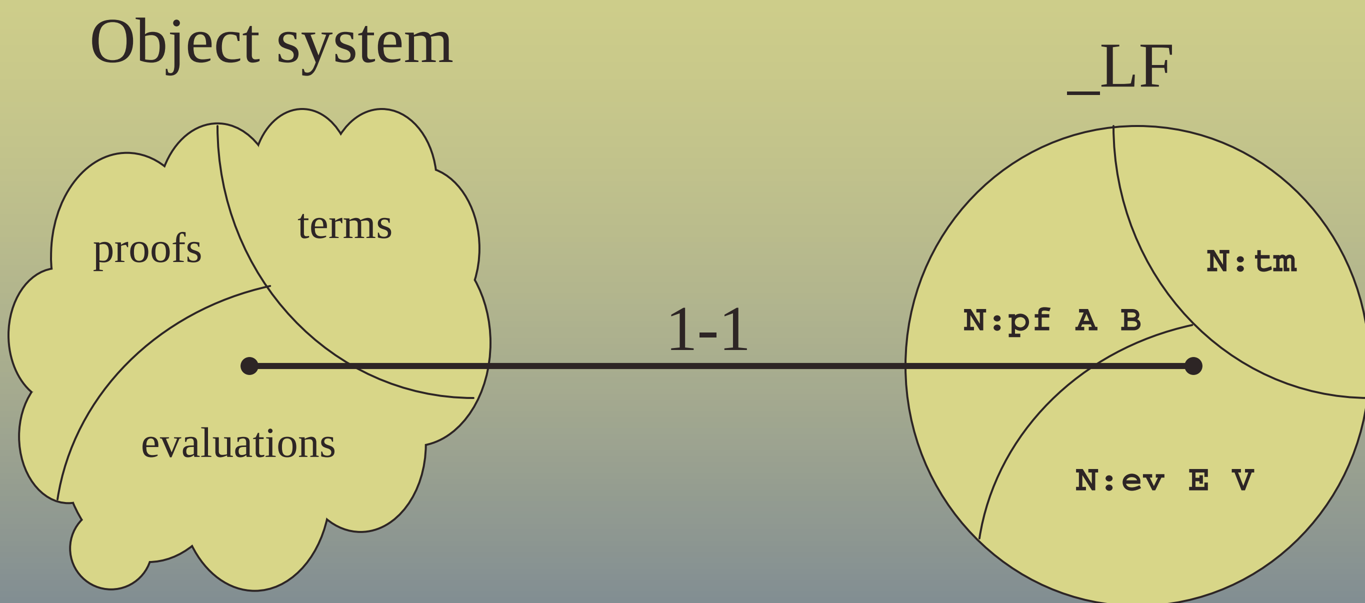
Representation Targets

Mottos, mottos, mottos …

- LF: *judgments-as-types / proofs-as-objects*
  - 3+5 = 8 ⇒ `N : ev (+ 3 5) 8`

    Judgment (a statement we want to make) ⇒ object : type
- LLF: *state-as-linear-hypotheses / imperative-computations-as-linear-functions*
- CLF: *concurrent-computations-as-monadic-expressions / …*
- nextLF: *blablablablablabla-**as**-blablablablablablablabla / blablablablablablablablabl-**as**-blablablablablababablablablablablablabla*

Make it Canonical, Sam



Each object of interest has exactly 1 representation

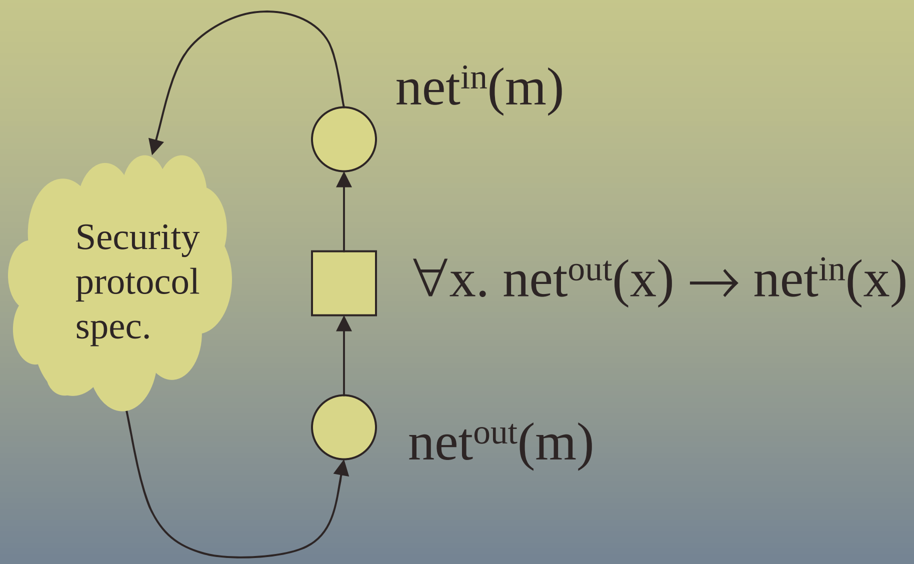
- Canonical objects:
  - η-long, β-normal _LF term
  - Decidable, computable

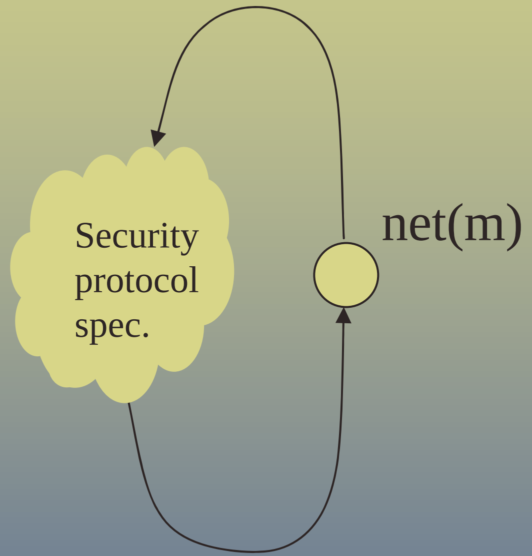
But what is LLF?

- Types ("asynchronous" constructors of ILL)
  - $A ::= a \mid \Pi x{:}A.\ B \mid A \multimap B \mid A \mathbin{\&} B \mid \top$
- Terms
  - $N ::= x \mid \lambda x{:}A.\ N \mid N_1\ N_2$
    $\lambda\hat{}x{:}A.\ N \mid N_1\hat{}N_2 \mid$
    $\langle N_1, N_2 \rangle \mid \mathsf{fst}\ N \mid \mathsf{snd}\ N \mid$
    $\langle\rangle$
- Main judgment
  - $\Gamma\ ;\ \Delta \vdash N : A$

CLF

An Example





Many instances can be executing concurrently

LLF Encoding

$\texttt{net : } \textit{step} \quad \texttt{o- net}^{\texttt{out}}\texttt{ m}$
$\qquad\qquad\qquad \texttt{o- (net}^{\texttt{in}}\texttt{ m -o } \textit{step}\texttt{).}$

- LLF forces continuation-passing style
- Consider 2 independent applications:
  - $\lambda \mathbf{n}^{\mathbf{i}}_{1}\texttt{. net \^{} } \mathbf{n}^{\mathbf{o}}_{1}\texttt{ \^{} (}\lambda \mathbf{n}^{\mathbf{i}}_{2}\texttt{. net \^{} } \mathbf{n}^{\mathbf{o}}_{2}\texttt{ \^{} C)}$
  - $\lambda \mathbf{n}^{\mathbf{i}}_{2}\texttt{. net \^{} } \mathbf{n}^{\mathbf{o}}_{2}\texttt{ \^{} (}\lambda \mathbf{n}^{\mathbf{i}}_{1}\texttt{. net \^{} } \mathbf{n}^{\mathbf{o}}_{1}\texttt{ \^{} C)}$

  Should be indistinguishable (***true concurrency***)
- Equate them at the meta-level

  $\texttt{same-trace } T_1\ T_2\texttt{ o- …}$

  Never-ending even for small system!

Encoding in Linear logic

$$\forall m.\ net^{out}\ m\ \multimap\ net^{in}\ m$$

- Much simpler
- In general, requires "synchronous" operators
  - $\otimes$ and **1**
- Concurrency given by "commuting conversions"

$$\text{let } x_1 \otimes y_1 = N_1 \text{ in } (\text{let } x_2 \otimes y_2 = N_2 \text{ in } M)$$

$$= \text{let } x_2 \otimes y_2 = N_2 \text{ in } (\text{let } x_1 \otimes y_1 = N_1 \text{ in } M) \qquad \text{if } x_i, y_i \notin FV(R_{2-i})$$

- … looks like what we want …

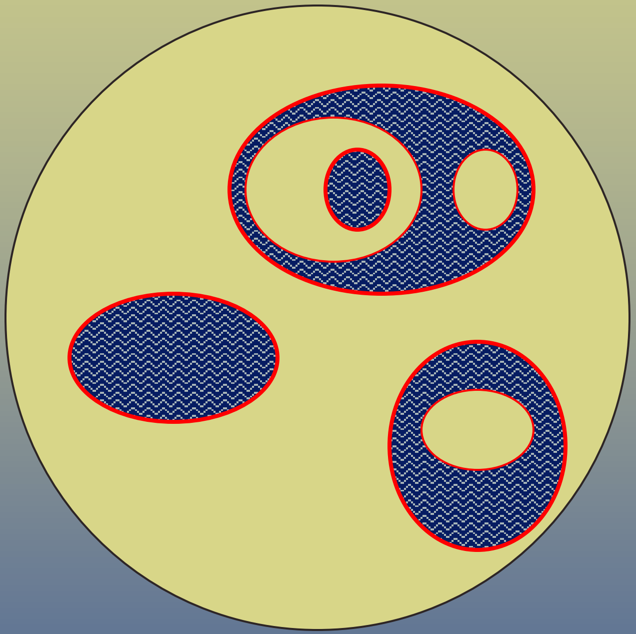
However …

- Commuting conversions are too wild
  - Allow permutations we don't care for
- Synchronous types destroy uniqueness of canonical forms
  - **`nat:type.   z:nat.   s:nat->nat.   c:1.`**
  - Natural numbers: **`z`**, **`s z`**, **`s`** (**`s z`**), …
  - What about **`let 1 = c in z`** ? What if **`c`** is linear?
- No good! ☹

Monadic Encapsulation

Separate synchronous and asynchronous types

- *Outside* the monad
  - LLF types (asynchronous)
  - η-long, β-normal forms
- *Inside* the monad
  - Synchronous types
  - Commuting conversions
    - *Concurrency equation*
  - η-long, β-normal forms
- Monad is a sandbox for synchronous behavior



CLF

- Types
  - A ::= a | Π x:A. B | A –o B | A & B | T | {S}
  - S ::= A | !A | $S_1 \otimes S_2$ | **1** | ∃x:A. S
- Terms
  - N ::= x | λ x:A. N | N_1 N_2 | λ^x:A. N | N_1^N_2 | $<N_1,N_2>$ | fst N | snd N | $<>$ | {E}
  - E ::= M | let {p} = N in E
  - M ::= N | !N | $M_1 \otimes M_2$ | 1 | [N,M]
  - p ::= x | !x | $p_1 \otimes p_2$ | 1 | [x,p]

Example in CLF

$$\texttt{net : net}^{\texttt{in}}\ \texttt{m -o \{ net}^{\texttt{out}}\ \texttt{m \}.}$$

- Relating the 2 specifications
  - 2 sets of CLF declarations
  - Meta-level definition of trace transformation

    $$\texttt{simplify-net \{T}^{\texttt{i/o}}\texttt{\} \{T\}}$$

    - Trivial mapping
    - Permutations handled automatically
      - No need to take action
      - Critical for more complex examples

Examples and Applications

- π-calculus
  - Synchronous
  - Asynchronous
- Concurrent ML
- Petri nets
  - Execution-sequence semantics
  - Trace semantics
- MSR security protocol specification language

- No implementation … yet …

Conclusions

CLF

- A logical framework that internalizes true concurrency
- Monadic encapsulation tames commuting conversions
- Canonical approach to meta-theory
- Good number of examples

- *This is just the beginning … plenty more to do!*