Induction and Co-induction in Sequent Calculus

Alberto Momigliano
Department of Mathematics and Computer Science
University of Leicester

Joint work with Dale Miller and Alwen Tiu

Heriot-Watt
Edinburgh, May 23rd, 2003

Overview of the Talk

- Motivations and history.
- Partial Inductive Definitions and $FO\lambda^{\Delta\mathbb{N}}$.
- Induction and Coinduction rules.
- Implementation
- Circular proofs
- Future work

- **Logical Framework:** Meta-language for the specification, implementation and verification of the (meta)theory of deductive systems.

- Deductive systems encoded as logical theories. Properties verified inside the lf possibly via proof search paradigm.

- Example: static and dynamic (natural) semantics of a functional programming language (Mini-ML).

- Reasoning **within** a specified system: Implementation and execution of algorithms. Judgments for evaluation ($e \Downarrow v$) and type inference ($\Gamma \vdash e : \tau$).

- Reasoning **about** the meta-theoretic properties of a deductive system. Example: subject reduction: if $e \Downarrow v$ and $\vdash e : \tau$ then $\vdash v : \tau$.

- The framework must be able to support encodings of informal (mathematical) entities as abstract as possible, do not introduce ad hoc low-level notions.

- This means (to me) supporting **Higher Order Abstract Syntax** (HOAS). Yet, to preserve adequacy of encodings, If is intentionally *weak*, but must support for (co)inductive reasoning.

- The catch: HOAS and (co)induction are prima facie incompatible.

Relevant History of (Co)Induction in LF's

- From Gentzen to iterated inductive defs [Martin-Löf'71]
- Impredicative encoding of inductive types in system F [Böhm & Berarducci'85].
- Mendler-style (co)inductive types [Mendler'87], [Geuvers'92]
- CC^{Ind} [Pfenning & Paulin '90], CC^{CoInd} [Gimenez'96], inductive definitions in HOL [Melham'92,Paulson'97].
- Work on type-based termination [Barthe et al'03], (co)iteration [Abel et al'02], category-theoretic analysis. . .
- Sequent calculus: not much. Back to the drawing board:

Set-theoretic Induction

- A *rule set* $\mathcal{R}$ is set of pairs $\langle G, A\rangle$ (notation: $A \stackrel{\triangle}{=} G$) on an universe $\mathcal{U}$, such that $A \in \mathcal{U}, G \subseteq 2^{\mathcal{U}}$).

- $Id(\mathcal{R})$, the set *inductively defined* by $\mathcal{R}$ is the intersection of all $\mathcal{R}$-*closed* subset of $\mathcal{U}$, i.e. sets S such that for every $A \stackrel{\triangle}{=} G \in \mathcal{R}$, $G \subseteq S$ entails $A \in S$.

- Set-theoretically to prove a property $P \subseteq Id(\mathcal{R})$ by rule induction means showing that P is $\mathcal{R}$-closed.

- Equivalently for $\Phi_{\mathcal{R}} : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$, $\Phi_{\mathcal{R}}$ **monotone** define $\Phi_{\mathcal{R}}(S) = \{A \in \mathcal{U} \mid A \stackrel{\triangle}{=} G \in \mathcal{R}, G \subseteq S\}$. $Id(\mathcal{R})$ [$CoId(\mathcal{R})$] is the least [greatest] fixed point of this operator.

- More interestingly: **Partial** Inductive Definitions [Hallnäs'91]: $A \stackrel{\triangle}{=} C$, where C is not simply a set of atoms, but:

$$\textit{Conditions} \quad C \quad ::= \quad \top \mid \bot \mid A \mid (C_i)_{i \in I} \mid C \to C'$$

- Logically speaking, they correspond to *higher-order* rules [Schroeder-Heister'84] or *parametric-hypothetical judgments* [Martin-Löf'85]. In general, they do not yield monotone inductive operators.

- Note: logic of ID may be classical or constructive, but logic of PID makes sense only in a strictly constructive setting.

- Sequent calculus for intuitionistic logic $\Gamma \longrightarrow B$ over simply typed λ-terms, but **no** quantification of predicates, with:

- **Definitions**, i.e. finite sets of *clause*

$$\forall \bar{x}[p_1(\bar{t}_1) \stackrel{\triangle}{=} B_1] \quad \ldots \quad \forall \bar{x}[p_n(\bar{t}_n) \stackrel{\triangle}{=} B_n] \quad (n \geq 0)$$

  The symbol $\stackrel{\triangle}{=}$ not a logical connective, just to denote definitional clauses.

- Left and right rules for introducing defined predicates with possibly non-Horn clauses [Halnass'91, S. Heister'93].

- Differently from set(type)-theoretic inductive definitions, case analysis (left rule) is not bungled up with induction principles and monotonicity.

- Fix a definition and a equality theory, here ho-patterns.

- Right rule: as a logic program, this is *backchaining*.

$$\frac{\Gamma \longrightarrow B\theta}{\Gamma \longrightarrow A}\,(\mathit{def}\mathcal{R}), H\theta = A, \text{ for some clause } \forall\bar{x}[H \stackrel{\triangle}{=} B]$$

- Left rule:

$$\frac{\{B\theta, \Gamma\theta \longrightarrow C\theta \mid \ \theta A = \theta H, \text{ for some clause } \forall\bar{x}[H \stackrel{\triangle}{=} B]\}}{A, \Gamma \longrightarrow C}\,(\mathit{def}\mathcal{L})$$

- *Unification* used to select clauses from a definition. Eigenvariables can be instantiated. The set of premises can be empty, finite, or infinite since in some theories CSUs are not effectively computable.

$$\textit{Terms}\quad e \;::=\; x \mid \mathbf{fun}\ x\,.\ e \mid e \mathbin{@} e' \mid \mathbf{fix}\ \mathbf{x}.\ e$$

- $\mathbf{fix}\ \mathbf{x}.\ e$ (and $\mathbf{fun}\ x\,.\ e$) is a **binder**. It induces an equivalence class under α-renaming, and requires capture avoiding substitutions: encoded how?

- This is the crucial choice, as it will dominate the formal development, in particular the representation of judgments.

- Why reinvent the wheel? Use a (weak) λ-calculus as a meta-language [Church'40]. It has a single binder λ, from which any other one is definable: built-in α-equivalence, well-behaved substitutions and more.

- Big step (lazy) evaluation $e \Downarrow v$:

$$\frac{e_1 \Downarrow \mathbf{fun}\ x\ .\ e_1' \qquad [e_2/x]e_1' \Downarrow v}{(e_1\ @\ e_2) \Downarrow v}\ \mathbf{ev\_app}$$

$$\frac{}{\mathbf{fun}\ x\ .\ e \Downarrow \mathbf{fun}\ x\ .\ e}\ \mathbf{ev\_fun} \qquad \frac{[\mathbf{fix\ x.}\ e/x]e \Downarrow v}{\mathbf{fix\ x.}\ e \Downarrow v}\ \mathbf{ev\_fix}$$

- Key notion is **substitution** of expressions, cf. $\mathbf{ev\_fix}, \mathbf{ev\_app}$.

- The latter is a major hassle and so easy to get it wrong.

- Why not use λ-calculus substitution, i.e. β-reduction ?

- Encode it in Full HOAS (no separate category of variables): object-level binder represented via the meta-level binder λ and the function space from the logic.

```
exp :  type.
fun :  (exp -> exp) -> exp.
@ :  exp -> exp -> exp.
fix :  (exp -> exp) -> exp.
```

- For example $\ulcorner \mathbf{fix}\ \ x.\ x\ @\ x \urcorner = \mathtt{fix}(\lambda \mathtt{x}\ .\ \mathtt{x}\ @\ \mathtt{x})$. Note: `exp` **cannot** be a data-type because of negative occurrences, so those are simply constants. Crucially, **freeness** properties must be derivable, wrt inversion on judgments or case analysis on syntax.

- Hence, PID's **not** (a fragment) of higher-order logic.

Intermission: Free Equality

- There are choices. Standard one (discarded here to keep linearity of heads) is equality as a definition $X = X \stackrel{\triangle}{=} \top$. Freeness follow by *def*$\mathcal{L}$, see injectivity of OL binders:

$$\frac{\dfrac{}{\top \longrightarrow E = E}\ \mathit{def}\mathcal{R}}{\mathbf{fun}\ \lambda x.\, E\ x = \mathbf{fun}\ \lambda x.\, F\ x \longrightarrow E = F}\ \mathit{def}\mathcal{L}$$

- Sequent calculus rendition of free equality [S-Heister'94].

- Exploiting unification, which is now restricted to equality reasoning (definitions as Clark's completion). We adopt it.

$$\frac{\{\Gamma\theta \longrightarrow C\theta \mid \theta \in CSU(s,t)\}}{\Gamma, s = t \longrightarrow C}\ \mathit{eq}\mathcal{L} \qquad \frac{s =_{\beta\eta} t}{\Gamma \longrightarrow s = t}\ \mathit{eq}\mathcal{R}$$

- This corresponds to the following PID (note HOAS):

$$\begin{aligned}
\mathbf{fun}\ E \Downarrow \mathbf{fun}\ E &\stackrel{\triangle}{=} \top \\
(E_1\ @\ E_2) \Downarrow V &\stackrel{\triangle}{=} \exists E' : E_1 \Downarrow \mathbf{fun}\ E' \wedge (E'\ E_2) \Downarrow V \\
\mathbf{fix}\ E \Downarrow V &\stackrel{\triangle}{=} E\ (\mathbf{fix}\ E) \Downarrow V
\end{aligned}$$

- *defR* provides Prolog-like computations:

$$\dfrac{\dfrac{}{\longrightarrow \mathbf{fun}\ \lambda y.\,(\mathbf{fix}\ \lambda x.\,\mathbf{fun}\ \lambda y.\,x) \Downarrow \mathbf{fun}\ \lambda y.\,(\mathbf{fix}\ \lambda x.\,\mathbf{fun}\ \lambda y.\,x)}\ \mathit{defR}}{\longrightarrow \mathbf{fix}\ (\lambda x.\,\mathbf{fun}\ \lambda y.\,x) \Downarrow \mathbf{fun}\ (\lambda y.\,\mathbf{fix}\ \lambda x.\,\mathbf{fun}\ \lambda y.\,(\mathbf{fix}\ \lambda x.\,\mathbf{fun}\ \lambda y.\,x))}\ \mathit{defR}$$

- *defL* provides case analysis:

$$\dfrac{\dfrac{}{\top \longrightarrow \mathbf{fun}\ E = \mathbf{fun}\ E}\ \mathit{defR}}{\mathbf{fun}\ E \Downarrow V \longrightarrow V = \mathbf{fun}\ E}\ \mathit{defL}$$

(Co)Induction in $FO\lambda^{\Delta\mathbb{N}}$

- To prove anything interesting, need some form of induction. In [McDowell'97], only induction on natural numbers:

$$\frac{}{\Gamma \longrightarrow \mathit{nat}\ z}\ \mathit{nat}\mathcal{R} \qquad \frac{\Gamma \longrightarrow \mathit{nat}\ I}{\Gamma \longrightarrow \mathit{nat}\ s\ I}\ \mathit{nat}\mathcal{R}$$

$$\frac{\longrightarrow D\ z \quad D\ j \longrightarrow D\ (s\ j) \quad D\ I, \Gamma \longrightarrow C}{\mathit{nat}\ I, \Gamma \longrightarrow C}\ \mathit{nat}\mathcal{L}$$

- Introduce counters in definitions, derive and use complete induction (doable but clunky):

$$\mathbf{fix}\ E \Downarrow^{(s\ N)} V \ \stackrel{\triangle}{=}\ E\ (\mathbf{fix}\ E) \Downarrow^{N} V \ldots$$

- But what about coinductive predicates? Inductive encoding works only with co-continuity, need direct approach.

$$(E_1 @ E_2) \Uparrow \ \stackrel{\triangle}{=}\ E_1 \Uparrow \vee \exists E' : E_1 \Downarrow \mathbf{fun}\ E' \wedge (E'\ E_2) \Uparrow$$

$$\mathbf{fix}\ E \Uparrow \ \stackrel{\triangle}{=}\ E\ (\mathbf{fix}\ E) \Uparrow$$

- For convenience, we define predicate with a single (disjunctive) body, where pattern matching is a constraint ex: $\mathit{nat}\, x$ has body $(x = z) \vee \exists y.(x = s\, y) \wedge \mathit{nat}\, y$. For this talk, assume also no mutual recursion.

- **Inductive** predicates $\forall \bar{x}.p\ \bar{x} \stackrel{\mu}{=} B$, holding in every **least** fixed point:

$$\frac{I\ \bar{t}, \Gamma \longrightarrow C \qquad B[I]\ \bar{x} \longrightarrow I\ \bar{x}}{p\ \bar{t}, \Gamma \longrightarrow C}\,\mathcal{IL} \qquad \frac{\Gamma \longrightarrow B\ \bar{t}}{\Gamma \longrightarrow p\ \bar{t}}\,\mathcal{IR}$$

  where I is the **Invariant** predicate, $\bar{x}$ fresh eigenvariables and $B[I]$ is B where every occurrence of p is replaced by I.

- Note: the old $\mathit{nat}\mathcal{L}$ rule is now derivable via the invariant $I = \lambda x.[(x = z) \vee \exists y.(x = s\, y)] \wedge D\ x$. In general, $\mathit{def}\mathcal{L}$ is derivable via $\mathcal{IL}$.

- Easiest example in the book: value soundness $e \Downarrow v \longrightarrow value\ v$. Let $I = \lambda xy\ .\ value\ y$.

$$\frac{I\ e\ v \longrightarrow C \qquad B[I]\ x_e\ x_v \longrightarrow I\ x_e\ x_v}{e \Downarrow v \longrightarrow C}\ \mathcal{IL}$$

- Let C be $value\ v$, with the definition: $value\ (\mathbf{fun}\ E) \stackrel{\mu}{=} \top$.

- Invariant implies conclusion $value\ v \longrightarrow value\ v$. Inductive step(s):

  1. $x_e = \mathbf{fun}\ E = x_v \longrightarrow value\ x_v$. $\mathcal{IR}$ on $value\ (\mathbf{fun}\ E)$.

  2. $x_e = E_1\ @\ E_2 \wedge x_v = V \wedge \ldots I\ (E'\ E_2)\ V \longrightarrow value\ x_v$

  3. $x_e = \mathbf{fix}\ F \wedge x_v = V \wedge I\ (F\ (\mathbf{fix}\ F))\ V \longrightarrow value\ x_v$.

- **Conductive** predicates $\forall \bar{x}.p\ \bar{x} \stackrel{\nu}{=} B$, holding in every **greatest** fixed point. S is the purported **Simulation**:

$$\frac{B\ \bar{t}, \Gamma \longrightarrow C}{p\ \bar{t}, \Gamma \longrightarrow C}\,\mathcal{CIL} \qquad \frac{\Gamma \longrightarrow S\ \bar{t} \quad S\ \bar{x} \longrightarrow B[S]\ \bar{x}}{\Gamma \longrightarrow p\ \bar{t}}\,\mathcal{CIR}$$

- Proviso $\mathsf{lvl}(S) \leq \mathsf{lvl}(p)$. Note: the *def*$\mathcal{R}$ rule is derivable.

- Example: let $h \equiv \mathbf{fix}\ (\lambda x.\ x\ @\ x)$. It holds $\longrightarrow h \Uparrow$. Let S be $\lambda y.\ y = h \vee y = h\ @\ h$. LHS is
$\longrightarrow h = h \vee h = h\ @\ h$. RHS yields two subgoals
$\longrightarrow h\ @\ h = E_1\ @\ E_2 \wedge (E_1 = h\ @\ h \vee E_1 = h)$ and
$\longrightarrow \exists F.\ h = \mathbf{fix}\ F \wedge (F\ (\mathbf{fix}\ F) = h\ @\ h \vee F\ (\mathbf{fix}\ F) = h)$

- **Regular** predicates $\forall \bar{x}.p\ \bar{x} \stackrel{\triangle}{=} B$, true in **every** fixed point

- Without restrictions, PID may be inconsistent – cut not eliminable – even before (co)inductive rules, see $p \stackrel{\triangle}{=} p \supset \bot$.

- Give predicates and formulas a level and require definitions to be stratified. This brings back monotonicity at the level of definitions, but supports HOAS. Need to go two-level to accommodate reasoning with non-stratifiable hypothetical judgments such as typing [McDowell et al'02, Momigliano et al '03]. Similarly with Twelf and its meta-logic $\mathcal{M}_\omega$.

- Use dependency graph to avoid a predicate to be (mutually) defined both inductively and co-inductively.

- Cut-elimination based on adaptation of Tait-style *reducibility*, generalizes McDowell's proof for $FO\lambda^{\Delta\mathbb{N}}$.

- Well-order on proofs induced by "local" cut reduction, proof by induction on the length of proof of the rightmost derivation, and on the reducibility of the left. Coinduction may need notion of "coreducibility".

- Given $A\bar{y} \stackrel{\mu}{=} B[A]\ \bar{y}$

$$\dfrac{\dfrac{\overset{\Pi_1}{\Delta \longrightarrow B[A]\ \bar{t}}}{\Delta \longrightarrow A\ \bar{t}}\,\mathcal{IR} \qquad \dfrac{\overset{\Pi_B}{B[S]\ \bar{y} \longrightarrow S\ \bar{y}} \qquad \overset{\Pi}{S\ \bar{t}, \Gamma \longrightarrow C}}{A\ \bar{t}, \Gamma \longrightarrow C}\,\mathcal{IL}}{\Delta, \Gamma \longrightarrow C}\,mc \quad .$$

- This reduces to

$$\dfrac{\dfrac{\overset{\Pi'_1}{\Delta \longrightarrow B[S]\ \bar{t}} \qquad \overset{\Pi_B[\bar{t}/\bar{y}]}{B[S]\ \bar{t} \longrightarrow S\ \bar{t}}}{\Delta \longrightarrow S\ \bar{t}}\,mc \qquad \overset{\Pi}{S\ \bar{t}, \Gamma \longrightarrow C}}{\Delta, \Gamma \longrightarrow C}\,mc$$

- Hybrid [Ambler et al'02], a package on top of Isabelle/HOL.

- Generalizing [A. Gordon'94], a deep encoding in Isabelle/HOL of a λ-calculus which provides a form of *logical framework* where the syntax of an object level logic can be adequately represented by HOAS.

- Supports *tactical theorem proving*, *(co)induction*, but *definitional*, so consistent *within a classical type theory*. This gives access to all the features of Isabelle HOL, in particular (co)induction.

- Bottom line: 'Hybrid syntax' provides the user with a form of HOAS, but this is syntactic sugar (more properly a **definition**) for an underlying de Bruijn representation.

- Start with a de Bruijn data-type for the the untyped λ calculus with constants:

$$\texttt{CON} :: con \Rightarrow expr \qquad \texttt{VAR} :: nat \Rightarrow expr$$

$$\texttt{\$\$} :: expr \Rightarrow expr \Rightarrow expr \qquad \texttt{Abs} :: expr \Rightarrow expr$$

- From that **define** a function $\texttt{LAM} :: (expr \Rightarrow expr) \Rightarrow expr$,

- a predicate $proper :: expr \Rightarrow bool$ to restrict to well-formed first order terms and a predicate $abstr :: (expr \Rightarrow expr) \Rightarrow bool$ to rule out exotic λ-term.

- Example: $\bigwedge V_1.\, \bigwedge V_2.\, V_1\ V_2$ is coded as $\mathsf{LAM}\, v_1.\, (\mathsf{LAM}\, v_2.\, (v_1 \texttt{\$\$} v_2))$ and reduced by rewriting to the de Bruijn term $Abs\ (Abs\ (Bnd\ 1\ \texttt{\$\$}\ Bnd\ 0))$.

- Since stratified PID are monotonic, implement them as a Isabelle HOL(co)inductive definition (**abstr** annotations to isolate the parametric function space, to be internalized):

$$\begin{array}{rcl} \mathit{coinductive}\ \mathit{divrg}\ \_ & :: & \mathit{exp} \Rightarrow \mathit{bool} \\ [\![\ \mathit{divrg}\ E_1\ ]\!] & \Longrightarrow & \mathit{divrg}\ (E_1\$E_2) \\ [\![\ E_1 >\!\!>\!\!> \mathbf{fun}\ E';\ \mathsf{abstr}\ \mathsf{E}';\ \mathit{divrg}\ (E'\ E_2)\ ]\!] & \Longrightarrow & \mathit{divrg}\ (E_1\$E_2)\ldots \end{array}$$

- **Prove** (by comprehension) the set-theoretic soundness of the $\mathcal{CIR}$ rule, by Isabelle HOL's co-induction, where B_{div} is the completion body of the definition, abstracted over S:

$$\mathit{Goal}\ ''(\exists S.\ S\ t \wedge (\forall y.\ S\ y \rightarrow B_{div}\ S\ y)) \rightarrow \mathit{divrg}\ t''\,;$$

- Use the latter to prove:

$$\mathit{Goal}\ ''\mathit{divrg}\ (\mathbf{fix}\ x.\ x\ \$\ x)''\,;$$

```
B_div == (%S x. (EX E1 E2. x = E1 $ E2 & (S E1) |
(EX E E1 E2.  x = E1 $ E2 & E1 >>> lam x. E x & abstr E & S (E E2))
(EX E. x = fix E & abstr E & S(E (fix E)))))"

    Goal "divrg (fix x. x $ x)";
br co_div 1;

 1. EX S. S (fix x. x $ x) & (ALL y. S y --> B_div S y)
by(res_inst_tac [("x"," (%s . s = (h $ h) | s = h)")] exI 1);

 1. (fix x. x $ x = h $ h | fix x. x $ x = h) &
    (ALL y. y = h $ h | y = h --> B_div (%s. s = h $ h | s = h) y)
br conjI 1;

 1. fix x. x $ x = h $ h | fix x. x $ x = h
 2. ALL y. y = h $ h | y = h --> B_div (%s. s = h $ h | s = h) y
by(simp_tac(simpset() addsimps [h_def])1);
```

```
 1. ALL y. y = h $ h | y = h --> B_div (%s. s = h $ h | s = h) y
bw B_div_def; by(strip_tac 1);

 1. !!y. y = h $ h | y = h (* first disjunct *)
      ==> EX e1 e2. y = e1 $ e2 & (e1 = h $ h | e1 = h) |
          (EX E e1 e2. y = e1 $ e2 &
               e1 >>> bLam E & abstr E & (E e2 = h $ h | E e2 = h))
  (EX E. y = fix E & abstr E & (E (fix E) = h $ h | E (fix E) = h))
be disjE 1; by(fast_tac(HOL_cs addss (simpset() addsimps [h_def])) 1)

 1. !!y. y = h
      ==> ...
  (EX E. y = fix E & abstr E & (E (fix E) = h $ h | E (fix E) = h))
bw h_def;by(blast_tac(HOL_cs addIs [abstr_xx])1);
...
Level 10
divrg (fix x. x $ x)
No subgoals!
```

Circular Proofs

- Using $\mathcal{CIR}$ requires the invention of a simulation: An alternative lazy way is to drop the rule and do proof search and circularity checking (guarded induction), as suggested by Coquand and implemented in CC^{CoInd}.

- Derivations (with $def\mathcal{L}def\mathcal{R}$) may include cycles. The derivation of a sequent can be closed if there is a leaf which is an instance of another sequent in the derivation graph.

- Circular branch has to be *guarded*, i.e., there has to be an application of $def\mathcal{R}$ in the branch (up to permutation). Equivalent in sequents to *productivity* in type theory.

- Restriction: circularity across cut rule must be restricted. Otherwise we lose cut-elimination.

- Divergence (as before), for $h \equiv \mathbf{fix}\ (\lambda x.\ x\ @\ x)$:

$$\dfrac{\dfrac{\longrightarrow h \Uparrow}{\longrightarrow h\ @\ h \Uparrow}\ \mathit{defR}}{\longrightarrow h \Uparrow}\ \mathit{defR}$$

- Applicative simulation [Abramski'90]:

$$R \leq S \ \stackrel{\nu}{=}\ \forall T.\ R \Downarrow \mathbf{fun}\ T \rightarrow (\exists U. S \Downarrow \mathbf{fun}\ U \wedge \forall p.\ (T\ p) \leq (U\ p)$$

- Transitivity $m_1 \leq m_2, m_2 \leq m_3 \longrightarrow m_1 \leq m_3$. Two *defL* and one of *defR* and quantifier eliminations (on the right): get $(m_1\ p) \leq (m_2\ p), (m_2\ p) \leq (f_3\ p) \longrightarrow (m_1\ p) \leq (f_3\ p)$ and close.

- $S = \lambda\ xy.\ \exists w\,.\, x \leq w \wedge w \leq y$ constructed via proof-search.

- Cut free circular proofs **not** complete for infinite behavior. For example take *simulation* in CCS:

$$\mathit{sim}\ P\ Q \stackrel{\triangle}{=} \forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{sim}\ P\ '\ Q'$$

- *sim* P Q, where $P = \mu x.(a.x)$ and $Q = \mu x.((a.x \mid a.x))$.

$$P \xrightarrow{a} P \xrightarrow{a} P \xrightarrow{a} \ldots$$

$$Q \xrightarrow{a} (Q \mid a.Q) \xrightarrow{a} (Q \mid Q) \xrightarrow{a} ((Q \mid a.Q) \mid Q) \xrightarrow{a} \ldots$$

  but there is no cut free circular proof of it.

- But you can prove it via $\mathcal{CIR}$ with the simulation

$$S := \lambda P \lambda Q.(P = \mu x.a.x) \wedge \exists Q'.Q \xrightarrow{a} Q \mid Q'.$$

Extensions: ∇

- Eigenvariables have two roles: freshness and site for instantiation (inversion and cut-elim). Mapping object-level abstraction to meta-level universal quantif. blurs them.

- Example: $\nu xy.\ [x = y]\ \bar{x}z.0$ has no transitions.
  $\nu\lambda x.\ \lambda y.\ \bar{x}z.0 \stackrel{A}{\longrightarrow} \bot$ true iff there are two distinct names.

- Introduce another layer of abstraction into sequent, writing signatures explicitly in the sequent, and to each formula there is an associated local signature:

  $$\Sigma : \sigma_1 \triangleright B_1, \ldots, \sigma_n \triangleright B_n \longrightarrow \sigma \triangleright C$$

  where Σ is the global eigenvariable which has scope over the entire sequent and σ_i is the local variables. An intuitive reading of $x, y \triangleright Bxy$ is $\lambda x \lambda y.Bxy$. Local variables cannot be instantiated.

Extensions: ∇ 2/3

- A new quantifier ∇ manipulates the local signatures.

$$\frac{\Sigma, \sigma \vdash t : \tau \quad \Sigma : \sigma \triangleright B[t/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \forall_\tau x.B, \Gamma \longrightarrow \mathcal{C}} \forall\mathcal{L} \qquad \frac{\Sigma, h : \Gamma \longrightarrow \sigma \triangleright B[(h\ \sigma)/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \forall x.B}$$

$$\frac{\Sigma, h : \sigma \triangleright B[(h\ \sigma)/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \exists x.B, \Gamma \longrightarrow \mathcal{C}} \exists\mathcal{L} \qquad \frac{\Sigma, \sigma \vdash t : \tau \quad \Sigma : \Gamma \longrightarrow \sigma \triangleright B[t/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \exists_\tau x.B}$$

$$\frac{\Sigma : (\sigma, y) \triangleright B[y/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \nabla x.B, \Gamma \longrightarrow \mathcal{C}} \nabla\mathcal{L} \qquad \frac{\Sigma : \Gamma \longrightarrow (\sigma, y) \triangleright B[y/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \nabla x.B} \nabla\mathcal{R}$$

- The interaction (scoping) between global and local variables in $\forall$- and $\exists$-rules. Achieved via *raising*.

Extensions: ∇

- The name-restriction in π-calculus transition is now interpreted as ∇.

$$\nu P \xrightarrow{\alpha} \nu\ Q \stackrel{\triangle}{=} \nabla x.(P \xrightarrow{\alpha} Q)$$

- Let $\Sigma = \{x, z, \alpha, Q\}$. We can now prove the judgment in the previous example:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma : w \triangleright ([x = w](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \mathit{def}\mathcal{L}}{\Sigma : . \triangleright \nabla y.([x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \nabla\mathcal{L}}{\Sigma : . \triangleright (\nu\lambda y.\,[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \mathit{def}\mathcal{L}}{\Sigma :\longrightarrow . \triangleright (\nu\lambda y.\,[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \supset \bot}\ \supset\mathcal{R}$$

- In applying *def*$\mathcal{L}$ rule, the local signature is mapped to λ-abstraction. The proof above relies on the failure of the unification $\lambda w.x = \lambda w.w$.

Conclusions

- Partial inductive definitions give an independent account of introduction and elimination rules for predicates.
- They allow HOAS at the syntax level, which is not required to be a data-type, orthogonally from HOL.
- Stratification is sufficient to ensure cut-elimination, while another layer (explicitly reference provability) is needed to reason about non-stratifiable hypo-judgments.
- (Co)Induction rules greatly adds to specification power.
- Circular proofs are a promising way to lazily search for (co)inductive proofs.

Future Work

- Take circular proofs seriously. Show soundness w.r.t. the $\mathcal{CIR}$ rule. Study restrictions that allow certain cuts. Investigate circular (well-founded) proofs in the **inductive** setting [Sprenger & Dam'03] and the relation with fixed point logics.

- Play a similar game (wrt coinduction) with LF and *Twelf*:

  1. coverage and guardedness checking at the LF level

  2. add ν to the meta-logic $\mathcal{M}_\omega$.

- Look at Mendler-style (co)induction, also to ensure guardedness as-you-go [Gimemenez'98].

- Semantical considerations: (co)-induction rules are sound wrt higher order logic. Should be complete too.

- Which semantics for circular proofs?

- Implementation:

  1. Use Hybrid, which is currently been extended in a way which could support ∇ [Ambler et al'03].

  2. A dedicated system for (normalized) PID's, on top of an interactive proof assistant (Isabelle, Coq).

- Experiment with one big case-study, say congruence of bisimulation in the higher-order π-calculus.

That's all, folks!