Focusing and higher-order abstract syntax

Noam Zeilberger

Carnegie Mellon University

12 January 2008

Programs $\longleftrightarrow$ Proofs

Programs	$\longleftrightarrow$	Proofs
typed lambda-calculus		natural deduction

Programs	$\longleftrightarrow$	Proofs
typed lambda-calculus		natural deduction
?		sequent calculus

Programs	$\longleftrightarrow$	Proofs
typed lambda-calculus		natural deduction
?		sequent calculus
ML, Haskell, . . .		?

Programs	$\longleftrightarrow$	Proofs
typed lambda-calculus		natural deduction
?		sequent calculus
ML, Haskell, . . .		?

Practical features

1. explicit evaluation order
2. pattern-matching

Programs	$\longleftrightarrow$	Proofs
typed lambda-calculus		natural deduction
?		sequent calculus
ML, Haskell, . . .		?

Practical features

1. explicit evaluation order
2. pattern-matching

Claim 1: **focused** sequent calculus provides both 1 and 2

Claim 2: these features actually *simplify* logic & language

The bureaucracy of syntax

Two equivalent λ-calculus terms:

$$\mathsf{case}(\pi_1\ x, y_1.\mathsf{case}(\pi_2\ x, z_1.e_1, z_2.e_2), y_2.\mathsf{case}(\pi_2\ x, z_1.e_3, z_2.e_4))$$

$$\mathsf{case}(\pi_2\ x, z_1.\mathsf{case}(\pi_1\ x, y_1.e_1, y_2.e_3), z_2.\mathsf{case}(\pi_1\ x, y_1.e_2, y_2.e_4))$$

Same term in ML:

```
case x of
    (Inl y1, Inl z1) => e1
  | (Inl y1, Inr z2) => e2
  | (Inr y2, Inl z1) => e3
  | (Inr y2, Inr z2) => e4
```

Bottom-up proof-search

Naive bottom-up search:

1. Find a rule with goal as conclusion
2. Apply rule (premises become the new goals)
3. Upon failure, backtrack and try applying a different rule

Fact: naive bottom-up search is wildly inefficient for LL

Andreoli's solution: tame proof-search via inversion and focus

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ (\otimes L)$$

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ (\otimes L)$$

Inversion stage: chain application of invertible rules

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ (\otimes L)$$

Inversion stage: chain application of invertible rules

$$\Gamma; (A \oplus B) \otimes (C \oplus D) \vdash E$$

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,(\otimes L)$$

Inversion stage: chain application of invertible rules

$$\frac{\Gamma; A \oplus B, C \oplus D \vdash E}{\Gamma; (A \oplus B) \otimes (C \oplus D) \vdash E}$$

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,(\otimes L)$$

Inversion stage: chain application of invertible rules

$$\frac{\dfrac{\Gamma; A, C \oplus D \vdash E \qquad \Gamma; B, C \oplus D \vdash E}{\Gamma; A \oplus B, C \oplus D \vdash E}}{\Gamma; (A \oplus B) \otimes (C \oplus D) \vdash E}$$

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,(\otimes L)$$

Inversion stage: chain application of invertible rules

$$\dfrac{\dfrac{\Gamma; A, C \vdash E \quad \Gamma; A, D \vdash E}{\Gamma; A, C \oplus D \vdash E} \quad \dfrac{\Gamma; B, C \vdash E \quad \Gamma; B, C \vdash E}{\Gamma; B, C \oplus D \vdash E}}{\dfrac{\Gamma; A \oplus B, C \oplus D \vdash E}{\Gamma; (A \oplus B) \otimes (C \oplus D) \vdash E}}$$

Inversion

A rule is *invertible* if its conclusion implies its premises

Every connective of LL is invertible on either left or right, e.g.:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\&R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,(\otimes L)$$

Inversion stage: chain application of invertible rules

. . . and treat as one big step:

$$\frac{\Gamma, A, C \vdash E \quad \Gamma, A, D \vdash E \quad \Gamma, B, C \vdash E \quad \Gamma, B, D \vdash E}{\Gamma; (A \oplus B) \otimes (C \oplus D) \vdash E}$$

Flashback

Two equivalent λ-calculus terms:

$$\mathsf{case}(\pi_1\ x, y_1.\mathsf{case}(\pi_2\ x, z_1.e_1, z_2.e_2), y_2.\mathsf{case}(\pi_2\ x, z_1.e_3, z_2.e_4))$$

$$\mathsf{case}(\pi_2\ x, z_1.\mathsf{case}(\pi_1\ x, y_1.e_1, y_2.e_3), z_2.\mathsf{case}(\pi_1\ x, y_1.e_2, y_2.e_4))$$

Same term in ML:

```
case x of
    (Inl y1, Inl z1) => e1
  | (Inl y1, Inr z2) => e2
  | (Inr y2, Inl z1) => e3
  | (Inr y2, Inr z2) => e4
```

Focus

Focus stage: chain application of non-invertible rules

Focus

Focus stage: chain application of non-invertible rules

$$\Gamma \vdash [(A \oplus B) \otimes (C \oplus D)]$$

Focus

Focus stage: chain application of non-invertible rules

$$\frac{\Gamma_1 \vdash [A \oplus B] \quad \Gamma_2 \vdash [C \oplus D]}{\Gamma_1, \Gamma_2 \vdash [(A \oplus B) \otimes (C \oplus D)]}$$

Focus

Focus stage: chain application of non-invertible rules

$$\frac{\dfrac{\Gamma_1 \vdash [B]}{\Gamma_1 \vdash [A \oplus B]} \quad \Gamma_2 \vdash [C \oplus D]}{\Gamma_1, \Gamma_2 \vdash [(A \oplus B) \otimes (C \oplus D)]}$$

Focus

Focus stage: chain application of non-invertible rules

$$\frac{\dfrac{\Gamma_1 \vdash [B]}{\Gamma_1 \vdash [A \oplus B]} \quad \dfrac{\Gamma_2 \vdash [C]}{\Gamma_2 \vdash [C \oplus D]}}{\Gamma_1, \Gamma_2 \vdash [(A \oplus B) \otimes (C \oplus D)]}$$

Focus

Focus stage: chain application of non-invertible rules

. . . and treat as one big (nondeterministic) step:

$$\frac{\Gamma_1 \vdash [B] \quad \Gamma_2 \vdash [C]}{\Gamma_1, \Gamma_2 \vdash [(A \oplus B) \otimes (C \oplus D)]}$$

Focus

Focus stage: chain application of non-invertible rules

. . . and treat as one big (nondeterministic) step:

$$\frac{\Gamma_1 \vdash [B] \quad \Gamma_2 \vdash [C]}{\Gamma_1, \Gamma_2 \vdash [(A \oplus B) \otimes (C \oplus D)]}$$

Theorem (Andreoli): focusing proof-search is complete.

Duality^2

1. Inversion and focus are dual stages
2. They divide the connectives into two groups

Together these dualities form a *square of opposition:*

polarity		inversion	focus
positive	$\otimes, \oplus, 1, 0$	left	right
negative	$\&, \parr, \top, \bot$	right	left

Duality^2

1. Inversion and focus are dual stages
2. They divide the connectives into two groups

Together these dualities form a *square of opposition:*

polarity		inversion	focus
positive	$\otimes, \oplus, 1, 0$	left	right
negative	$\&, \parr, \top, \bot$	right	left

Girard '93: dualities extend beyond LL by *polarization*...

Duality^2

1. Inversion and focus are dual stages
2. They divide the connectives into two groups

Together these dualities form a *square of opposition:*

polarity		inversion	focus
positive	$\otimes, \oplus, 1, 0$	left	right
negative	$\&, \parr, \top, \bot$	right	left

Girard '93: dualities extend beyond LL by *polarization*...

...i.e., by fixing evaluation order

What you'll find in the paper

A new formulation of focusing for intuitionistic logic with. . .

- big-step inversion and focus through higher-order rules
- A "connective-blind" proof of cut-elimination!

A Curry-Howard interpretation incorporating. . .

- CBV functions, strict products and sums, recursive types
- Pattern-matching "for free" through a HO encoding:

  $\text{fnc} = \text{pat} \longrightarrow \text{exp} \qquad \text{val} = \text{pat} \times \text{sub}$

- A connective-blind proof of type safety

A formalization in Coq

A very quick sketch

Connectives of the day

Polarized intuitionistic connectives

- Positive: $\otimes, \oplus, 1, 0$ (strict products and sums)
- Negative: $\xrightarrow{v}$ (CBV function space)

(Same entailments as IL, different proofs.)

Notation for polarized formulas:

$$\begin{array}{rcl} P, Q & ::= & X \mid P \otimes Q \mid P \oplus Q \mid 0 \mid 1 \mid N \\ N & ::= & P \xrightarrow{v} Q \end{array}$$

$\leftarrow$implicit coercion

In paper: $P ::= \cdots \mid \mu X.P$

"Hidden" slides (not in paper): $N, M := \cdots \mid N \xrightarrow{n} M \mid N \& M \mid \ldots$

Linear right rules $\approx$ Pattern typing

Linear contexts $\Delta ::= \cdot \mid X, \Delta \mid N, \Delta$

$\boxed{\Delta \Rightarrow P}$

$$\frac{}{X \Rightarrow X} \qquad \frac{}{N \Rightarrow N}$$

$$\frac{}{\cdot \Rightarrow 1} \qquad \frac{\Delta_1 \Rightarrow P \quad \Delta_2 \Rightarrow Q}{\Delta_1, \Delta_2 \Rightarrow P \otimes Q}$$

(no rule for 0) $$\frac{\Delta \Rightarrow P}{\Delta \Rightarrow P \oplus Q} \quad \frac{\Delta \Rightarrow Q}{\Delta \Rightarrow P \oplus Q}$$

Linear right rules $\approx$ Pattern typing

Linear contexts $\Delta ::= \cdot \mid x : X, \Delta \mid f : N, \Delta$

$\boxed{\Delta \Rightarrow P}$

$$\frac{}{X \Rightarrow X} \qquad \frac{}{N \Rightarrow N}$$

$$\frac{}{\cdot \Rightarrow 1} \qquad \frac{\Delta_1 \Rightarrow P \quad \Delta_2 \Rightarrow Q}{\Delta_1, \Delta_2 \Rightarrow P \otimes Q}$$

(no rule for 0) $$\frac{\Delta \Rightarrow P}{\Delta \Rightarrow P \oplus Q} \quad \frac{\Delta \Rightarrow Q}{\Delta \Rightarrow P \oplus Q}$$

Linear right rules $\approx$ Pattern typing

Linear contexts $\Delta ::= \cdot \mid x : X, \Delta \mid f : N, \Delta$

$\boxed{\Delta \Rightarrow P}$

$$\frac{}{X \Rightarrow X} \qquad \frac{}{N \Rightarrow N}$$

$$\frac{}{\cdot \Rightarrow 1} \qquad \frac{\Delta_1 \Rightarrow P \quad \Delta_2 \Rightarrow Q}{\Delta_1, \Delta_2 \Rightarrow P \otimes Q}$$

(no rule for 0) $\qquad \frac{\Delta \Rightarrow P}{\Delta \Rightarrow P \oplus Q} \quad \frac{\Delta \Rightarrow Q}{\Delta \Rightarrow P \oplus Q}$

Linear right rules $\approx$ Pattern typing

Linear contexts $\Delta ::= \cdot \mid x : X, \Delta \mid f : N, \Delta$

$\boxed{\Delta \Rightarrow p : P}$

$$\frac{}{x : X \Rightarrow x : X} \qquad \frac{}{f : N \Rightarrow f : N}$$

$$\frac{}{\cdot \Rightarrow () : 1} \qquad \frac{\Delta_1 \Rightarrow p_1 : P \quad \Delta_2 \Rightarrow p_2 : Q}{\Delta_1, \Delta_2 \Rightarrow (p_1, p_2) : P \otimes Q}$$

(no rule for 0) $\qquad \dfrac{\Delta \Rightarrow p : P}{\Delta \Rightarrow \mathsf{inl}\, p : P \oplus Q} \qquad \dfrac{\Delta \Rightarrow p : Q}{\Delta \Rightarrow \mathsf{inr}\, p : P \oplus Q}$

Usual pattern restrictions emerge!

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash [P]$	right-focus
$\Gamma; P \vdash Q$	left-inversion
$\Gamma \vdash P$	unfocused
$\Gamma \vdash \Delta$	multiple (conjoined) conclusions

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash [P]$

$\Gamma; P \vdash Q$

$\Gamma \vdash P$

$\Gamma \vdash \Delta$

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash V : [P]$ value

$\Gamma; P \vdash Q$

$\Gamma \vdash P$

$\Gamma \vdash \Delta$

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash V : [P]$	value
$\Gamma \vdash F : P > Q$	(CBV) function
$\Gamma \vdash P$	
$\Gamma \vdash \Delta$	

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash V : [P]$	value
$\Gamma \vdash F : P > Q$	(CBV) function
$\Gamma \vdash E : P$	expression
$\Gamma \vdash \Delta$	

Focused [sequent/lambda]-calculus

Unrestricted contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

Judgments:

$\Gamma \vdash V : [P]$	value
$\Gamma \vdash F : P > Q$	(CBV) function
$\Gamma \vdash E : P$	expression
$\Gamma \vdash \sigma : \Delta$	substitution

Untyped syntax

Canonical fragment (β-reduced, η-long):

$$
\begin{array}{lll}
V & ::= & p[\sigma] \\
F & ::= & \lambda\phi \\
 & & \quad \text{where } \phi ::= (p_1 \mapsto E_1 \mid \cdots \mid p_n \mapsto E_n) \\
E & ::= & V \mid F(g(V)) \\
\sigma & ::= & \cdot \mid (y/x, \sigma) \mid (F/f, \sigma)
\end{array}
$$

Identity and cut principles:

$$
\begin{array}{lll}
F & ::= & \cdots \mid \mathsf{id} \mid f \\
E & ::= & \cdots \mid F(V) \mid F(E)
\end{array}
$$

Right-focus $\approx$ Value typing

$$\boxed{\Gamma \vdash [P]}$$

$$\frac{\Delta \Rightarrow P \quad \Gamma \vdash \Delta}{\Gamma \vdash [P]}$$

Right-focus $\approx$ Value typing

$$\boxed{\Gamma \vdash V : [P]}$$

$$\frac{\Delta \Rightarrow P \quad \Gamma \vdash \Delta}{\Gamma \vdash [P]}$$

Right-focus $\approx$ Value typing

$$\boxed{\Gamma \vdash V : [P]}$$

$$\frac{\Delta \Rightarrow p : P \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p[\sigma] : [P]}$$

Right-focus $\approx$ Value typing

$$\boxed{\Gamma \vdash V : [P]}$$

$$\frac{\Delta \Rightarrow p : P \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p[\sigma] : [P]}$$

Yes, this is really a “value” in the ML sense…

```
(fn x => x*x, fn x => x-3)
```

Right-focus $\approx$ Value typing

$$\boxed{\Gamma \vdash V : [P]}$$

$$\frac{\Delta \Rightarrow p : P \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p[\sigma] : [P]}$$

Yes, this is really a “value” in the ML sense…

```
(f,g)[(fn x => x*x)/f,(fn x => x-3)/g]
```

Left-inversion $\approx$ CBV function typing

$\boxed{\Gamma; P \vdash Q}$

$$\frac{\forall(\Delta \Rightarrow P) : \quad \Gamma, \Delta \vdash Q}{\Gamma; P \vdash Q}$$

Left-inversion $\approx$ CBV function typing

$$\boxed{\Gamma \vdash F : P > Q}$$

$$\frac{\forall(\Delta \Rightarrow P) : \quad \Gamma, \Delta \vdash Q}{\Gamma; P \vdash Q}$$

Left-inversion $\approx$ CBV function typing

$\boxed{\Gamma \vdash F : P > Q}$

$$\frac{\forall(\Delta \Rightarrow p : P) : \quad \Gamma, \Delta \vdash \phi(p) : Q}{\Gamma \vdash (\lambda\phi) : P > Q}$$

Left-inversion $\approx$ CBV function typing

$$\boxed{\Gamma \vdash F : P > Q}$$

$$\frac{\forall(\Delta \Rightarrow p : P) : \quad \Gamma, \Delta \vdash \phi(p) : Q}{\Gamma \vdash (\lambda\phi) : P > Q}$$

ϕ a (partial) map from patterns to expressions!

In Coq, define $\mathtt{Lam} : (\mathsf{pat} \longrightarrow \mathsf{exp}) \longrightarrow \mathsf{fnc}$

Left-inversion $\approx$ CBV function typing

$$\boxed{\Gamma \vdash F : P > Q}$$

$$\frac{\forall(\Delta \Rightarrow p : P) : \quad \Gamma, \Delta \vdash \phi(p) : Q}{\Gamma \vdash (\lambda\phi) : P > Q}$$

ϕ a (partial) map from patterns to expressions!

In Coq, define $\mathtt{Lam} : (\mathsf{pat} \longrightarrow \mathsf{exp}) \longrightarrow \mathsf{fnc}$

Hmm. . . ?!

A more abstract syntax

The basic premise of this higher-order encoding is that the static and dynamic semantics of a language shouldn't care so much about the details of how a function is written down, but just get to the point about what expression is evaluated for each pattern. (Note this is *not* the same as only caring about the mapping from values to results!)

Several people have complained that calling this sort of encoding "higher-order abstract syntax" is misleading, since HOAS has a well-established usage referring to meta-level *substitution functions* used to encode object-level binders. They are probably right that my choice of terminology was confusing—sorry! Maybe it should be "abstract higher-order syntax"?. . .

Small-step semantics as cut-elimination

$$\mathsf{id}(V) \rightsquigarrow V$$

$$\boxed{\frac{\phi(p)\ \text{defined}}{(\lambda\phi)(p[\sigma]) \rightsquigarrow \phi(p)[\sigma]}} \qquad \frac{E \rightsquigarrow E'}{F(E) \rightsquigarrow F(E')}$$

Theorem: Type Safety

Proof does not mention (positive) connectives!

Cf. Gentzen/Prawitz/Dummett proof-theoretic semantics

Coq encoding

"Curry-style" (untyped syntax, type system, opsem, type safety):

`http://www.cs.cmu.edu/~noam/research/focusing.tar`

"Church-style" (typed syntax only):

`http://www.cs.cmu.edu/~noam/research/focus-church.v`

"Church v2.0" (with uniform treatment of negative types):

`http://www.cs.cmu.edu/~noam/research/focusing2.v`

Define functions using Coq's built-in pattern-matching!

(But use de Bruijn indices for binding, alas. . .)

Coq example: `plus`

```
Definition plus : fnc :=
  Fix (Lam (fun p => match p with
    | Pair m Ze => Return (Val m (Sub nil))
    | Pair m (Su n) =>
        EComp
         (Lam (fun n' =>
                 Return (Val (Su n')
                             (Sub nil))))
         (1,0) (Val (Pair m n) (Sub nil))
    | _ => EFail
    end)).
```

Conclusions

Focusing is awesome!

Focusing is awesome!

Why?

1. **Pattern-matching** $\approx$ focus and inversion
2. **Explicit evaluation order** $\approx$ polarity

Future Work

The computational world of polarized logic & type theory

- Mixed polarities = mixed evaluation strategies (Haskell?)
- Polymorphism & intersection/union types
- Dependent types

Higher-order encodings

- What is the relationship between AHOS and HOAS?
- Can they be combined in one logical framework?
- See Brigitte's talk! And paper with Licata & Harper. . .

Thanks!

Recursion and recursive types

Recursion:

$$\frac{\Gamma, f : P \xrightarrow{v} Q \vdash F : P > Q}{\Gamma \vdash \mathsf{fix}\, f.F : P > Q} \qquad (\mathsf{fix}\, f.F)(V) \rightsquigarrow ([\mathsf{fix}\, f.F/f]F)(V)$$

Recursive types:

$$\frac{\Delta \Rightarrow p : P[\mu X.P/X]}{\Delta \Rightarrow \mathsf{fold}(p) : \mu X.P}$$

A uniform treatment of negative types

Make coercions explicit: $\downarrow N$ positive, $\uparrow P$ negative

$$P \xrightarrow{v} Q = P \to \uparrow Q \qquad N \xrightarrow{n} M = \downarrow N \to M$$

Define negative connectives through linear left rules:

$$\boxed{\Delta; N \Rightarrow \gamma}$$

$$\frac{}{\cdot; \uparrow P \Rightarrow P} \qquad \frac{\Delta_1 \Rightarrow P \quad \Delta_2; N \Rightarrow \gamma}{\Delta_1, \Delta_2; P \to N \Rightarrow \gamma}$$

$$\frac{\Delta; N \Rightarrow \gamma}{\Delta; N \& M \Rightarrow \gamma} \qquad \frac{\Delta; M \Rightarrow \gamma}{\Delta; N \& M \Rightarrow \gamma}$$

Curry-Howard: destructor patterns

General intuitionistic focusing

Right-focus and left-inversion:

$$\frac{\Delta \Rightarrow P \quad \Gamma \vdash \Delta}{\Gamma \vdash [P]} \qquad \frac{\forall(\Delta \Rightarrow P) : \quad \Gamma, \Delta \vdash \gamma}{\Gamma; P \vdash \gamma}$$

Right-inversion and left-focus:

$$\frac{\forall(\Delta; N \Rightarrow \gamma) : \quad \Gamma, \Delta \vdash \gamma}{\Gamma \vdash N} \quad \frac{\Delta; N \Rightarrow \gamma_0 \quad \Gamma \vdash \Delta \quad \Gamma; \gamma_0 \vdash \gamma}{\Gamma; [N] \vdash \gamma}$$

Unfocused sequents:

$$\frac{\Gamma \vdash [P]}{\Gamma \vdash P} \qquad \frac{N \in \Gamma \quad \Gamma; [N] \vdash \gamma}{\Gamma \vdash \gamma}$$