Separating Functional Computation from Relations

Ulysse Gérard and **Dale Miller**

Inria Saclay & LIX, Ecole Polytechnique
Palaiseau France

31 October 2018, TU Wien

Introduction

Logical foundations of arithmetic usually start with a quantificational logic of relations.

For example: Gentzen's proof of consistency of arithmetic; Church's STT [1940]; Andrews's textbook [2002].

We want a treatment of functional computation based of relations.

Introduction

Logical foundations of arithmetic usually start with a quantificational logic of relations.

For example: Gentzen's proof of consistency of arithmetic; Church's STT [1940]; Andrews's textbook [2002].

We want a treatment of functional computation based of relations.

Application: We wish to extended the Abella theorem prover to have conventional notations, e.g. $(3 * x) + 2 \leq 10$, instead of

$$\exists x_1.\ \mathit{times}\ 3\ x\ x_1 \wedge \exists X_2.\ \mathit{plus}\ x_1\ 2\ x_2 \wedge \mathit{lesseq}\ x_2\ 10$$

We are willing to change the parser and proof automation, but not the logic.

Earlier approaches

- Enhance the equality theory (e.g., Troelstra) : primitive recursive functions are black-boxes and all computation instances (e.g. $23 + 756 = 779$) are added as ground equations.

Earlier approaches

- Enhance the equality theory (e.g., Troelstra) : primitive recursive functions are black-boxes and all computation instances (e.g. $23 + 756 = 779$) are added as ground equations.
- Hybridize the logical calculus with terms and confluent rewriting such as in the $\lambda\Pi$-calculus modulo framework used in Dedukti (Cousineau & Dowek)

Earlier approaches

- Enhance the equality theory (e.g., Troelstra) : primitive recursive functions are black-boxes and all computation instances (e.g. $23 + 756 = 779$) are added as ground equations.
- Hybridize the logical calculus with terms and confluent rewriting such as in the $\lambda\Pi$-calculus modulo framework used in Dedukti (Cousineau & Dowek)
- Add choice operators such as Hilbert's ϵ and Church's ι to coerce relations that encode functions into actual functions.

Earlier approaches

- Enhance the equality theory (e.g., Troelstra) : primitive recursive functions are black-boxes and all computation instances (e.g. $23 + 756 = 779$) are added as ground equations.
- Hybridize the logical calculus with terms and confluent rewriting such as in the $\lambda\Pi$-calculus modulo framework used in Dedukti (Cousineau & Dowek)
- Add choice operators such as Hilbert's ϵ and Church's ι to coerce relations that encode functions into actual functions.

If R is an $n+1$-ary predicate such that

$$\forall \bar{x}.([\exists y.R(\bar{x}, y)] \wedge \forall y \forall z [R(\bar{x}, y) \supset R(\bar{x}, z) \supset y = z])$$

then there exists a n-ary function f_R s.t. $f_R(\bar{x}) = y$ *iff* $R(\bar{x}, y)$. Church formally wrote this using the choice operator ι:

$$\lambda x_1 \ldots \lambda x_n.\iota(\lambda y.R(x_1, \ldots, x_n, y))$$

A new design

We want a new "rule" such that :

$$\frac{\vdash Q(\mathbf{5})}{\vdash Q(\mathbf{2}+\mathbf{3})}$$

A new design

We want a new "rule" such that :

$$\frac{\vdash Q(\mathbf{5})}{\vdash Q(\mathbf{2}+\mathbf{3})}$$

We want to achieve this goal in a purely logical, proof-search oriented setting. We use the following two ideas.

- A focused proof system to synthesize such rules
- A term representation that helps to translate arithmetic expression into expressions involving predicate

Focusing: a top-level perspective

- Proof-search in Gentzen's sequent calculus suffer from a great deal of non-determinancy and redundancy.
- A focused proof system guides proof construction by distinguishing between invertible and non-invertible rules.
- Such proofs contain an alternation of two phases: the negative / invertible / "don't care" phase and the positive / non-invertible / "don't know" phase.
- Focused proof systems have two kinds of sequents to build these two phases.

Road-map

1. We give a presentation of Heyting arithmetic in which fixed points and term equality are logical connectives. The negative phase in its focused proof system is determinate (reading it as a mapping from its conclusion to its premises). Functional computations are computed by such phases.

Road-map

1. We give a presentation of Heyting arithmetic in which fixed points and term equality are logical connectives. The negative phase in its focused proof system is determinate (reading it as a mapping from its conclusion to its premises). Functional computations are computed by such phases.
2. An ambiguity of polarity arises with singletons. If $P(\cdot)$ is a singleton, then,

   $$\forall x[P(x) \supset Q(x)] \equiv \exists x[P(x) \wedge Q(x)] \equiv Q(\epsilon P)$$

   It is then always possible to position P in the negative phase.

Road-map

1. We give a presentation of Heyting arithmetic in which fixed points and term equality are logical connectives. The negative phase in its focused proof system is determinate (reading it as a mapping from its conclusion to its premises). Functional computations are computed by such phases.
2. An ambiguity of polarity arises with singletons. If $P(\cdot)$ is a singleton, then,

$$\forall x[P(x) \supset Q(x)] \equiv \exists x[P(x) \wedge Q(x)] \equiv Q(\epsilon P)$$

   It is then always possible to position P in the negative phase.
3. Ultimately: focusing in logic (not arithmetic) can define administrative normal forms, a term representation which can connect functions-as-constructors to functions-as-relations.

The propositional fragment

Propositional intuitionistic logic formulas are given by the logical connectives $\wedge$, $\vee$, and $\supset$, the logical constants t and f, and atomic formulas.

The propositional fragment

Propositional intuitionistic logic formulas are given by the logical connectives $\wedge$, $\vee$, and $\supset$, the logical constants t and f, and atomic formulas.

A polarized formula P is positive if it is a positive atomic formula or its top-level logical connective is either t^+, f, $\wedge^+$, or $\vee$.

A polarized formula N is negative if it is a negative atomic formula or its top-level logical connective is either t^-, $\wedge^-$, or $\supset$.

The propositional fragment

Negative Phase Introduction Rules

$$\frac{\Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow t^+, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \quad \frac{\Gamma \Uparrow \cdot \vdash B_1 \Uparrow \cdot \quad \Gamma \Uparrow \cdot \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \wedge^- B_2 \Uparrow \cdot} \quad \frac{\Gamma \Uparrow B_1 \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \supset B_2 \Uparrow \cdot}$$

$$\frac{\Gamma \Uparrow B_1, B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \wedge^+ B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \quad \frac{\Gamma \Uparrow B_1, \Theta \vdash \Delta_1 \Uparrow \Delta_2 \quad \Gamma \Uparrow B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \vee B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}$$

The propositional fragment

Negative Phase Introduction Rules

$$\frac{\Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow t^+, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow \cdot \vdash B_1 \Uparrow \cdot \quad \Gamma \Uparrow \cdot \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \wedge^- B_2 \Uparrow \cdot} \qquad \frac{\Gamma \Uparrow B_1 \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \supset B_2 \Uparrow \cdot}$$

$$\frac{\Gamma \Uparrow B_1, B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \wedge^+ B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow B_1, \Theta \vdash \Delta_1 \Uparrow \Delta_2 \quad \Gamma \Uparrow B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \vee B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}$$

Positive Phase Introduction Rules

$$\frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow B_2 \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \supset B_2 \vdash \cdot \Downarrow E} \qquad \frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow \cdot \vdash B_2 \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \wedge^+ B_2 \Downarrow \cdot}$$

$$\frac{\Gamma \Downarrow \cdot \vdash B_i \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \vee B_2 \Downarrow \cdot}\ i \in \{1, 2\} \qquad \frac{\Gamma \Downarrow B_i \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \wedge^- B_2 \vdash \cdot \Downarrow E}\ i \in \{1, 2\}$$

The propositional fragment

Structural rules

$$\frac{\Gamma, N \Downarrow N \vdash \cdot \Downarrow E}{\Gamma, N \Uparrow \cdot \vdash \cdot \Uparrow E}\ D_l \qquad \frac{C, \Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow C, \Theta \vdash \Delta_1 \Uparrow \Delta_2}\ S_l \qquad \frac{\Gamma \Uparrow P \vdash \cdot \Uparrow E}{\Gamma \Downarrow P \vdash \cdot \Downarrow E}\ R_l$$

$$\frac{\Gamma \Downarrow \cdot \vdash P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow P}\ D_r \qquad \frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow E}{\Gamma \Uparrow \cdot \vdash E \Uparrow \cdot}\ S_r \qquad \frac{\Gamma \Uparrow \cdot \vdash N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash N \Downarrow \cdot}\ R_r$$

Negative Phase Introduction Rules

$$\frac{\Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow t^+, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow \cdot \vdash B_1 \Uparrow \cdot \quad \Gamma \Uparrow \cdot \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \wedge^- B_2 \Uparrow \cdot} \qquad \frac{\Gamma \Uparrow B_1 \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \supset B_2 \Uparrow \cdot}$$

$$\frac{\Gamma \Uparrow B_1, B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \wedge^+ B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow B_1, \Theta \vdash \Delta_1 \Uparrow \Delta_2 \quad \Gamma \Uparrow B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \vee B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}$$

Positive Phase Introduction Rules

$$\frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow B_2 \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \supset B_2 \vdash \cdot \Downarrow E} \qquad \frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow \cdot \vdash B_2 \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \wedge^+ B_2 \Downarrow \cdot}$$

$$\frac{\Gamma \Downarrow \cdot \vdash B_i \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \vee B_2 \Downarrow \cdot}\ i \in \{1, 2\} \qquad \frac{\Gamma \Downarrow B_i \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \wedge^- B_2 \vdash \cdot \Downarrow E}\ i \in \{1, 2\}$$

Interlude: Bipoles

A bipole is a derivation whose conclusion and premises are all border sequents (of the form $\Gamma \Uparrow \cdot \vdash \cdot \Uparrow E$):

$$
\cfrac{\cfrac{\cfrac{\cfrac{\Gamma, N, \mathcal{N} \Uparrow \cdot \vdash \cdot \Uparrow E}{\cdots}}{\Gamma, N \Uparrow P \vdash \cdot \Uparrow E}\; R_l}{\cfrac{\Gamma, N \Downarrow P \vdash \cdot \Downarrow E}{\cdots}}}{\cfrac{\Gamma, N \Downarrow N \vdash \cdot \Downarrow E}{\Gamma, N \Uparrow \cdot \vdash \cdot \Uparrow E}\; D_l}
$$

Negative phase

Positive phase

These are the synthetic inference rules.

Examples of fixed point definitions

Declare the primitive type i and constants $z : i$ and $s : i \to i$.
z, $(s\ z)$, $(s\ (s\ z))$, $(s\ (s\ (s\ z)))$ are abbreviated by $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$ etc.

As a Horn clause theory

```
nat z.
nat (s X) :- nat X.
plus z X X.
plus (s X) Y (s Z) :- plus X Y Z.
```

Examples of fixed point definitions

Declare the primitive type i and constants $z : i$ and $s : i \to i$.
z, $(s\ z)$, $(s\ (s\ z))$, $(s\ (s\ (s\ z)))$ are abbreviated by $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$ etc.

As a Horn clause theory

```
nat z.
nat (s X) :- nat X.
plus z X X.
plus (s X) Y (s Z) :- plus X Y Z.
```

As fixed point definitions

$$\text{nat} = \mu\lambda N\lambda n(n = \mathbf{0} \vee \exists n'(n = s\ n' \wedge^{+} N\ n'))$$

$$\begin{aligned} plus = & \mu\lambda P\lambda n\lambda m\lambda p.(n = \mathbf{0} \wedge^{+} m = p) \vee \\ & \exists n'\exists p'(n = s\ n' \wedge^{+} p = s\ p' \wedge^{+} P\ n'\ m\ p') \end{aligned}$$

Rules for quantification, term equality and fix-point

Typed first-order quantification rules

$$\frac{\Sigma \vdash t : \tau \qquad \Sigma : \Gamma \Downarrow [t/x]B \vdash \cdot \Downarrow E}{\Sigma : \Gamma \Downarrow \forall x_\tau . B \vdash \cdot \Downarrow E} \qquad \frac{y : \tau, \Sigma : \Gamma \Uparrow \cdot \vdash [y/x]B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \cdot \vdash \forall x_\tau . B \Uparrow \cdot}$$

$$\frac{y : \tau, \Sigma : \Gamma \Uparrow [y/x]B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \exists x_\tau . B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma \vdash t : \tau \quad \Sigma : \Gamma \Downarrow \cdot \vdash [t/x]B \Downarrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \exists x_\tau . B \Downarrow \cdot}$$

Rules for quantification, term equality and fix-point

Typed first-order quantification rules

$$\frac{\Sigma \vdash t : \tau \qquad \Sigma : \Gamma \Downarrow [t/x]B \vdash \cdot \Downarrow E}{\Sigma : \Gamma \Downarrow \forall x_\tau . B \vdash \cdot \Downarrow E} \qquad \frac{y : \tau, \Sigma : \Gamma \Uparrow \cdot \vdash [y/x]B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \cdot \vdash \forall x_\tau . B \Uparrow \cdot}$$

$$\frac{y : \tau, \Sigma : \Gamma \Uparrow [y/x]B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \exists x_\tau . B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma \vdash t : \tau \quad \Sigma : \Gamma \Downarrow \cdot \vdash [t/x]B \Downarrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \exists x_\tau . B \Downarrow \cdot}$$

Equality rules [Girard, Schroeder-Heister]

$$\frac{\Sigma\theta : \Gamma\theta \Uparrow \Theta\theta \vdash \Delta_1\theta \Uparrow \Delta_2\theta}{\Sigma : \Gamma \Uparrow s = t, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \dagger \quad \frac{}{\Sigma : \Gamma \Uparrow s = t, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \ddagger \quad \frac{}{\Sigma : \Gamma \Downarrow \cdot \vdash t = t \Downarrow \cdot}$$

Provisos: (†) θ is the mgu of s and t. (‡) t and s are not unifiable.

Fixed point rules

$$\frac{\Sigma : \Gamma \Uparrow B(\mu B)\bar{t}, \Delta \vdash \cdot \Uparrow E}{\Sigma : \Gamma \Uparrow \mu B\, \bar{t}, \Delta \vdash \cdot \Uparrow E}\ unfoldL \qquad \frac{\Sigma : \Gamma \Downarrow \cdot \vdash B(\mu B)\bar{t} \Downarrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \mu B\, \bar{t} \Downarrow \cdot}\ unfoldR$$

The polarity ambiguity of singleton sets

Let P be a predicate of one argument such that

$$\vdash (\exists x.P(x)) \wedge (\forall x \forall y.P(x) \supset P(y) \supset x = y)$$

The polarity ambiguity of singleton sets

Let P be a predicate of one argument such that

$$\vdash (\exists x.P(x)) \wedge (\forall x \forall y.P(x) \supset P(y) \supset x = y)$$

As a consequence $\exists x.P(x) \wedge Q(x) \equiv \forall x.P(x) \supset Q(x)$.

Assume that P is a purely positive formula.

The polarity ambiguity of singleton sets

Let P be a predicate of one argument such that

$$\vdash (\exists x.P(x)) \wedge (\forall x \forall y.P(x) \supset P(y) \supset x = y)$$

As a consequence $\exists x.P(x) \wedge Q(x) \equiv \forall x.P(x) \supset Q(x)$.

Assume that P is a purely positive formula.

A proof of $\Sigma : \Gamma \Downarrow \cdot \vdash \exists x.P(x) \wedge Q(x) \Downarrow \cdot$ guesses a term t and then proves $\Sigma : \Gamma \Downarrow \cdot \vdash P(t) \Downarrow \cdot$ and $\Sigma : \Gamma \Downarrow \cdot \vdash Q(t) \Downarrow \cdot$.

The polarity ambiguity of singleton sets

Let P be a predicate of one argument such that

$$\vdash (\exists x.P(x)) \wedge (\forall x \forall y.P(x) \supset P(y) \supset x = y)$$

As a consequence $\exists x.P(x) \wedge Q(x) \equiv \forall x.P(x) \supset Q(x)$.

Assume that P is a purely positive formula.

A proof of $\Sigma : \Gamma \Downarrow \cdot \vdash \exists x.P(x) \wedge Q(x) \Downarrow \cdot$ guesses a term t and then proves $\Sigma : \Gamma \Downarrow \cdot \vdash P(t) \Downarrow \cdot$ and $\Sigma : \Gamma \Downarrow \cdot \vdash Q(t) \Downarrow \cdot$.

A proof of $\Sigma : \Gamma \Uparrow \cdot \vdash \forall x.P(x) \supset Q(x) \Uparrow \cdot$ computes the value that satisfies P, starting with proving $y, \Sigma : \Gamma \Uparrow P(y) \vdash Q(y) \Uparrow \cdot$. The completed phase has the premise $\Sigma : \Gamma \Uparrow \cdot \vdash \cdot \Uparrow Q(t)$.

Example

Consider a proof of $x, \Sigma : \Gamma \Uparrow \mathit{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$.

Example

Consider a proof of $x, \Sigma : \Gamma \Uparrow \mathit{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$.
Using unfoldL yields

$$x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^{+} \mathbf{3} = x) \vee \exists n' \exists x' (\mathbf{2} = s\ n' \wedge^{+} x = s\ x' \wedge^{+} \mathit{plus}\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x).$$

Example

Consider a proof of $x, \Sigma : \Gamma \Uparrow \mathit{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$.
Using unfoldL yields

$$x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^{+} \mathbf{3} = x) \vee \exists n' \exists x' (\mathbf{2} = s\ n' \wedge^{+} x = s\ x' \wedge^{+} \mathit{plus}\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x).$$

The disjunction introduction rule yields two premises:
(1) $x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^{+} \mathbf{3} = x) \vdash \cdot \Uparrow (Q\ x)$ is proved immediately.

Example

Consider a proof of $x, \Sigma : \Gamma \Uparrow plus\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$.
Using unfoldL yields

$$x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^+ \mathbf{3} = x) \vee \exists n' \exists x' (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ plus\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x).$$

The disjunction introduction rule yields two premises:
(1) $x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^+ \mathbf{3} = x) \vdash \cdot \Uparrow (Q\ x)$ is proved immediately.
(2)

$$\dfrac{\dfrac{x', \Sigma : \Gamma \Uparrow plus\ \mathbf{1}\ \mathbf{3}\ x' \vdash \cdot \Uparrow (Q\ (s\ x'))}{x, n', x', \Sigma : \Gamma \Uparrow (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ plus\ n'\ \mathbf{3}\ x') \vdash \cdot \Uparrow (Q\ x)}}{x, \Sigma : \Gamma \Uparrow (\exists n' \exists x' (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ plus\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x)}$$

Example

Consider a proof of $x, \Sigma : \Gamma \Uparrow \mathit{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$.
Using unfoldL yields

$$x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^{+} \mathbf{3} = x) \vee \exists n' \exists x' (\mathbf{2} = s\ n' \wedge^{+} x = s\ x' \wedge^{+} \mathit{plus}\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x).$$

The disjunction introduction rule yields two premises:
(1) $x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^{+} \mathbf{3} = x) \vdash \cdot \Uparrow (Q\ x)$ is proved immediately.
(2)
$$\dfrac{\dfrac{x', \Sigma : \Gamma \Uparrow \mathit{plus}\ \mathbf{1}\ \mathbf{3}\ x' \vdash \cdot \Uparrow (Q\ (s\ x'))}{x, n', x', \Sigma : \Gamma \Uparrow (\mathbf{2} = s\ n' \wedge^{+} x = s\ x' \wedge^{+} \mathit{plus}\ n'\ \mathbf{3}\ x') \vdash \cdot \Uparrow (Q\ x)}}{x, \Sigma : \Gamma \Uparrow (\exists n' \exists x' (\mathbf{2} = s\ n' \wedge^{+} x = s\ x' \wedge^{+} \mathit{plus}\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x)}$$

The negative phase terminates with the border premise

$$\Sigma : \Gamma \Uparrow \cdot \vdash \cdot \Uparrow (Q\ \mathbf{5})$$

Abstracting away the negative phase, we obtain the following synthetic inference rule :

$$\frac{\vdash Q(\mathbf{5})}{\mathit{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash Q(x)}$$

Abstracting away the negative phase, we obtain the following synthetic inference rule :

$$\frac{\vdash Q(\mathbf{5})}{plus\ \mathbf{2}\ \mathbf{3}\ x \vdash Q(x)}$$

$$\frac{\vdash Q(\mathbf{5})}{\vdash Q(\mathbf{2}+\mathbf{3})}$$

Phases as abstractions

There are two challenges to making abstractions of negative phases.

1. Since there may be many paths to compute the same functional value, the premises of a negative phase may *repeat the same sequents many times*. We can identify the premises of a negative phase as set of border sequents.

Phases as abstractions

There are two challenges to making abstractions of negative phases.

1. Since there may be many paths to compute the same functional value, the premises of a negative phase may *repeat the same sequents many times*. We can identify the premises of a negative phase as set of border sequents.
2. There are *many ways to build a negative phase* but all constructions yield the same border sequents. We will simply ignore how a phase is constructed.

Phases as abstractions

There are two challenges to making abstractions of negative phases.

1. Since there may be many paths to compute the same functional value, the premises of a negative phase may *repeat the same sequents many times*. We can identify the premises of a negative phase as set of border sequents.
2. There are *many ways to build a negative phase* but all constructions yield the same border sequents. We will simply ignore how a phase is constructed.

This latter challenge also holds in confluent rewriting systems: after finding one path to a normal form, no other paths need to be considered.

Need for suspensions

Suspension allows some mixing of functional and symbolic computation. For example, let *times* be

$$\mu\lambda T\lambda n\lambda m\lambda p((n = \mathbf{0}\wedge^{+} p = \mathbf{0})\vee\exists n'\exists p'(n = s\ n'\wedge^{+} T\ n'\ m\ p'\wedge^{+} plus\ p'\ m\ p))$$

To prove $(0 \times (x + 1)) + y = y$, we prove the formula

$$\forall u.\ times\ \mathbf{0}\ (s\ x)\ u \supset \forall v.\ plus\ u\ y\ v \supset v = y$$

$$y, u, v, \Sigma : \cdot \Uparrow times\ \mathbf{0}\ (s\ x)\ u,\ plus\ u\ y\ v \vdash v = y \Uparrow \cdot$$

Need for suspensions

Suspension allows some mixing of functional and symbolic computation. For example, let *times* be

$$\mu\lambda T\lambda n\lambda m\lambda p((n = \mathbf{0} \wedge^+ p = \mathbf{0}) \vee \exists n' \exists p'(n = s\ n' \wedge^+ T\ n'\ m\ p' \wedge^+ plus\ p'\ m\ p))$$

To prove $(0 \times (x+1)) + y = y$, we prove the formula

$$\forall u.\ times\ \mathbf{0}\ (s\ x)\ u \supset \forall v.\ plus\ u\ y\ v \supset v = y$$

$$y, u, v, \Sigma : \cdot \Uparrow times\ \mathbf{0}\ (s\ x)\ u,\ plus\ u\ y\ v \vdash v = y \Uparrow \cdot$$

Schedule the *times* predicate before the *plus* predicate.

Treating the *times* predicate causes the instantiation of u.

Then schedule the *plus* predicate.

Then the negative phase ends with $y, \Sigma : \cdot \Uparrow \cdot \vdash \cdot \Uparrow y = y$.

In general: Suspend *plus* and *times* if their first argument is an eigenvariable.

Suspension restrictions

$\mathcal{S}$ is defined at the mathematics level over the $(\mu B \bar{t})$ expression.

Examples

1. The μ-expression contains more than 100 symbols
2. The first term in the list $\bar{t}$ is an eigenvariable

Suspension restrictions

$\mathcal{S}$ is defined at the mathematics level over the $(\mu B \bar{t})$ expression.

Examples

1. The μ-expression contains more than 100 symbols
2. The first term in the list $\bar{t}$ is an eigenvariable

We need a restriction to enforce determinancy

> ($*$) *For all μ-expressions $(\mu B \bar{t})$ and for all substitutions θ defined on the eigenvariables free in that expression, if $\mathcal{S}$ holds for $(\mu B \bar{t})\theta$ then $\mathcal{S}$ holds for $(\mu B \bar{t})$.*

Suspensions during the positive phase

A suspension predicate $\mathcal{S}$ is defined only on μ-expressions.

Suspensions during the positive phase

A suspension predicate $\mathcal{S}$ is defined only on μ-expressions. If $\mathcal{S}$ holds for $(\mu B\bar{t})$, computation is suspended as the *unfoldL* rule will not unfold a suspended fixed point.

Suspensions during the positive phase

A suspension predicate $\mathcal{S}$ is defined only on μ-expressions. If $\mathcal{S}$ holds for $(\mu B \bar{t})$, computation is suspended as the *unfoldL* rule will not unfold a suspended fixed point.

$$\frac{\Sigma : \Gamma \Uparrow B(\mu B)\bar{t}, \Delta \vdash \cdot \Uparrow E}{\Sigma : \Gamma \Uparrow \mu B\, \bar{t}, \Delta \vdash \cdot \Uparrow E}\ \mathit{unfoldL}\dagger$$

Suspensions during the positive phase

A suspension predicate $\mathcal{S}$ is defined only on μ-expressions. If $\mathcal{S}$ holds for $(\mu B \bar{t})$, computation is suspended as the *unfoldL* rule will not unfold a suspended fixed point.

$$\frac{\Sigma : \Gamma \Uparrow B(\mu B)\bar{t}, \Delta \vdash \cdot \Uparrow E}{\Sigma : \Gamma \Uparrow \mu B\, \bar{t}, \Delta \vdash \cdot \Uparrow E}\ \textit{unfoldL}\dagger$$

$\Downarrow$-sequents need a new multiset zone Ω.

$$\Gamma \Downarrow \Theta; \Omega \vdash \Delta_1 \Downarrow \Delta_2.$$

Formulas in Ω are not "stored" just "suspended".

Only the decide, release, and initial rules deal with this context. It only exists in the positive phase.

Term representation using the $\lambda\kappa$-calculus (Brock-Nannestad, Guenot & Gustafsson)

$$\begin{array}{rrcl} \textit{Terms}: & t, u & ::= & \lambda x.t \mid x\ k \mid \uparrow p \\ \textit{Values}: & p, q & ::= & x \mid \downarrow t \\ \textit{Continuations}: & k & ::= & \varepsilon \mid p :: k \mid \kappa x.t \end{array}$$

Term representation using the $\lambda\kappa$-calculus (Brock-Nannestad, Guenot & Gustafsson)

$$\begin{array}{rrcl} \textit{Terms}: & t, u & ::= & \lambda x.t \mid x\ k \mid \uparrow p \\ \textit{Values}: & p, q & ::= & x \mid \downarrow t \\ \textit{Continuations}: & k & ::= & \varepsilon \mid p :: k \mid \kappa x.t \end{array}$$

$$\frac{}{\Gamma, x : a^+ \Downarrow \cdot \vdash x : a^+ \Downarrow \cdot}\ I_r$$

$$\frac{}{\Gamma \Downarrow a^- \vdash \cdot \Downarrow \varepsilon : a^-}\ I_l$$

Term representation using the $\lambda\kappa$-calculus (Brock-Nannestad, Guenot & Gustafsson)

$$
\begin{array}{rrl}
\textit{Terms}: & t, u ::= & \lambda x.t \mid x\ k \mid \uparrow p \\
\textit{Values}: & p, q ::= & x \mid \downarrow t \\
\textit{Continuations}: & k ::= & \varepsilon \mid p :: k \mid \kappa x.t
\end{array}
$$

$$
\frac{\Gamma \Uparrow \cdot \vdash t : N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash \downarrow t : N \Downarrow \cdot}\ R_r
\qquad
\frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Uparrow \cdot \vdash t : E \Uparrow \cdot}\ S_r
$$

$$
\frac{\Gamma \Downarrow \cdot \vdash p : P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow \uparrow p : P}\ D_r
\qquad
\frac{}{\Gamma, x : a^+ \Downarrow \cdot \vdash x : a^+ \Downarrow \cdot}\ I_r
$$

$$
\frac{}{\Gamma \Downarrow a^- \vdash \cdot \Downarrow \varepsilon : a^-}\ I_l
$$

Term representation using the $\lambda\kappa$-calculus (Brock-Nannestad, Guenot & Gustafsson)

$$
\begin{array}{rrl}
\textit{Terms}: & t, u ::= & \lambda x.t \mid x\ k \mid \uparrow p \\
\textit{Values}: & p, q ::= & x \mid \downarrow t \\
\textit{Continuations}: & k ::= & \varepsilon \mid p :: k \mid \kappa x.t
\end{array}
$$

$$
\frac{\Gamma \Uparrow \cdot \vdash t : N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash \downarrow t : N \Downarrow \cdot}\ R_r
\qquad
\frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Uparrow \cdot \vdash t : E \Uparrow \cdot}\ S_r
$$

$$
\frac{\Gamma \Downarrow \cdot \vdash p : P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow \uparrow p : P}\ D_r
\qquad
\frac{}{\Gamma, x : a^+ \Downarrow \cdot \vdash x : a^+ \Downarrow \cdot}\ I_r
$$

$$
\frac{\Gamma, x : P \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Downarrow P \vdash \cdot \Downarrow \kappa x.t : E}\ R_l/S_l
\qquad
\frac{\Gamma, x : N \Downarrow N \vdash \cdot \Downarrow k : E}{\Gamma, x : N \Uparrow \cdot \vdash \cdot \Uparrow x\ k : E}\ D_l
\qquad
\frac{}{\Gamma \Downarrow a^- \vdash \cdot \Downarrow \varepsilon : a^-}\ I_l
$$

Term representation using the $\lambda\kappa$-calculus (Brock-Nannestad, Guenot & Gustafsson)

$$\begin{array}{rrcl} \textit{Terms}: & t, u & ::= & \lambda x.t \mid x\ k \mid \uparrow p \\ \textit{Values}: & p, q & ::= & x \mid \downarrow t \\ \textit{Continuations}: & k & ::= & \varepsilon \mid p :: k \mid \kappa x.t \end{array}$$

$$\frac{\Gamma \Uparrow \cdot \vdash t : N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash \downarrow t : N \Downarrow \cdot}\ R_r \qquad \frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Uparrow \cdot \vdash t : E \Uparrow \cdot}\ S_r$$

$$\frac{\Gamma \Downarrow \cdot \vdash p : P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow \uparrow p : P}\ D_r \qquad \frac{}{\Gamma, x : a^+ \Downarrow \cdot \vdash x : a^+ \Downarrow \cdot}\ I_r$$

$$\frac{\Gamma, x : P \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Downarrow P \vdash \cdot \Downarrow \kappa x.t : E}\ R_l/S_l \qquad \frac{\Gamma, x : N \Downarrow N \vdash \cdot \Downarrow k : E}{\Gamma, x : N \Uparrow \cdot \vdash \cdot \Uparrow x\ k : E}\ D_l \qquad \frac{}{\Gamma \Downarrow a^- \vdash \cdot \Downarrow \varepsilon : a^-}\ I_l$$

$$\frac{\Gamma, x : A \Uparrow \cdot \vdash t : B \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash \lambda x.t : A \supset B \Uparrow \cdot}\ \supset_r / S_l \qquad \frac{\Gamma \Downarrow \cdot \vdash p : A \Downarrow \cdot \qquad \Gamma \Downarrow B \vdash \cdot \Downarrow k : E}{\Gamma \Downarrow A \supset B \vdash \cdot \Downarrow p :: k : E}\ \supset_l$$

Two normal forms for simply typed terms

1. When atoms are given a negative polarity then the terms annotating proofs are in $\beta\eta$-long normal form :

$$\lambda x_1 \ldots \lambda x_n . h \; t_1 \ldots t_m$$

Two normal forms for simply typed terms

1. When atoms are given a negative polarity then the terms annotating proofs are in $\beta\eta$-long normal form :

$$\lambda x_1 \ldots \lambda x_n.h\ t_1 \ldots t_m$$

Written in $\lambda\kappa$-terms :

$$\lambda x_1 \ldots \lambda x_n h.\ (\downarrow[\![t_1]\!] :: \cdots :: \downarrow[\![t_m]\!] :: \varepsilon)$$

Two normal forms for simply typed terms

1. When atoms are given a negative polarity then the terms annotating proofs are in $\beta\eta$-long normal form :

   $$\lambda x_1 \ldots \lambda x_n . h \; t_1 \ldots t_m$$

   Written in $\lambda\kappa$-terms :

   $$\lambda x_1 \ldots \lambda x_n h. \; (\downarrow [\![t_1]\!] :: \cdots :: \downarrow [\![t_m]\!] :: \varepsilon)$$

2. When atoms are given a positive polarity the terms annotating proofs are in administrative normal form (ANF):

   $\lambda x_1 \ldots \lambda x_n . h \, (p_1 {::} \cdots {::} p_m {::} \kappa y . t)$ (with t a term in ANF form)

Two normal forms for simply typed terms

1. When atoms are given a negative polarity then the terms annotating proofs are in $\beta\eta$-long normal form :

$$\lambda x_1 \dots \lambda x_n . h\ t_1 \dots t_m$$

   Written in $\lambda\kappa$-terms :

$$\lambda x_1 \dots \lambda x_n h .\ (\downarrow [\![t_1]\!] :: \dots :: \downarrow [\![t_m]\!] :: \varepsilon)$$

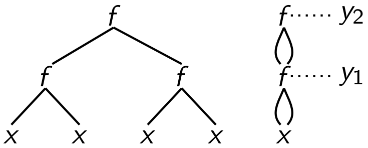
2. When atoms are given a positive polarity the terms annotating proofs are in administrative normal form (ANF):

$$\lambda x_1 \dots \lambda x_n . h\,(p_1 :: \dots :: p_m :: \kappa y . t) \quad \text{(with } t \text{ a term in ANF form)}$$

   With some syntactic sugar :

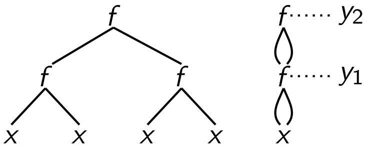
$$\lambda x_1 \dots \lambda x_n .\ \textbf{name}\ y = h\ (p_1, \dots, p_m)\ \textbf{in}\ t$$

Example: ANF and sharing



$f : i \; \rightarrow \; i \; \rightarrow \; i$ and $x : i$

Example: ANF and sharing

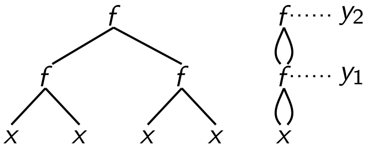


$f : i \ \rightarrow \ i \ \rightarrow \ i$ and $x : i$
When i is negative:

$$f \ (\downarrow(f \ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \downarrow(f \ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \varepsilon)$$

$$f \ (f \ (x, \ x), \ f \ (x, \ x))$$

Example: ANF and sharing



$f : i \;\rightarrow\; i \;\rightarrow\; i$ and $x : i$

When i is negative:

$$f\ (\downarrow(f\ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \downarrow(f\ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \varepsilon)$$

$$f\ (f\ (x,\ x),\ f\ (x,\ x))$$

When i is positive:

$$f\ x :: x :: \kappa y_1.(f\ y_1 :: y_1 :: \kappa y_2.y_2)$$

$$\textbf{name}\ y_1 = (f\ x\ x)\ \textbf{in}\ \textbf{name}\ y_2 = (f\ y_1\ y_1)\ \textbf{in}\ y_2$$

Mixed term representations

Add the binary infix term constructor $+$ of type $i \to i \to i$.

The expression $P(2+2)$ can be presented as :

$$\textbf{name}\ u = (s\ z)\ \textbf{in}\ \textbf{name}\ v = (s\ u)\ \textbf{in}\ \textbf{name}\ x = v + v\ \textbf{in}\ P(x)$$

We now have a mix of

- uninterpreted term constructors (e.g., z and s) and
- interpreted term constructors ($+$) which will be interpreted by predicates.

Interpreting term constructors

The formal introduction of a new interpreted binary term constructor such as $+ : i \to i \to i$ must be tied to a 3-ary μ-expression R and a formal proof that R encodes a function:

$$\forall x, y([\exists z. R(x, y, z)] \wedge \forall z \forall z' [R(x, y, z) \supset R(x, y, z') \supset z = z']).$$

Interpreting term constructors

The formal introduction of a new interpreted binary term constructor such as $+ : i \to i \to i$ must be tied to a 3-ary μ-expression R and a formal proof that R encodes a function:

$$\forall x, y([\exists z.R(x, y, z)] \wedge \forall z \forall z'[R(x, y, z) \supset R(x, y, z') \supset z = z']).$$

Then the *formula* (**name** $z = x + y$ **in** B) is interpreted as either $\forall z(R\ x\ y\ z \supset B)$ or $\exists z(R\ x\ y\ z \wedge^+ B)$.

Interpreting term constructors

The formal introduction of a new interpreted binary term constructor such as $+ : i \to i \to i$ must be tied to a 3-ary μ-expression R and a formal proof that R encodes a function:

$$\forall x, y([\exists z.R(x, y, z)] \wedge \forall z \forall z'[R(x, y, z) \supset R(x, y, z') \supset z = z']).$$

Then the *formula* (**name** $z = x + y$ **in** B) is interpreted as either $\forall z(R\ x\ y\ z \supset B)$ or $\exists z(R\ x\ y\ z \wedge^{+} B)$.

$$\frac{\Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \textbf{name}\ z = f\ \bar{x}\ \textbf{in}\ B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, \Theta \vdash B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \Theta \vdash \textbf{name}\ z = f\ \bar{x}\ \textbf{in}\ B \Uparrow \cdot}$$

Conclusion

$$\dfrac{\dfrac{\dfrac{\vdash Q(\mathbf{5})}{\mathit{plus}\ 2\ 3\ x \vdash Q(x)}\ \text{Negative Phase}}{\vdash \textbf{name}\ x = 2+3\ \textbf{in}\ Q(x)}\ \text{Interpret}}{\vdash Q(2+3)}\ \text{Parse/Translate}$$

Conclusion

We have presented a treatment of functional computation based on relations providing:

- a method for moving expressions denoting embedded computation into naming-combinators of the logic (ANF normal form)
- a mean of organizing introduction rules so that functional computations can be identified as one specific phase of computation (the negative phase).

Possible future work:

- Treat more datatypes than numerals; also higher-order expressions.
- Extend this project to include "functional-up-to-equivalence".
- Design this into Abella. See: LFMTP 2018 paper by Chaudhuri, Gérard, and M.

Thank you

$$\frac{y, \Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \textbf{name}\ y = f\ \bar{x}\ \textbf{in}\ B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{y, \Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, \Theta \vdash B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \Theta \vdash \textbf{name}\ y = f\ \bar{x}\ \textbf{in}\ B \Uparrow \cdot}$$

$$\frac{\Sigma : \Gamma \Uparrow \cdot \vdash \textbf{name}\ x = f\ \bar{x}\ \textbf{in}\ B \Uparrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \textbf{name}\ x = f\ \bar{x}\ \textbf{in}\ B \Downarrow \cdot} \qquad \frac{\Sigma : \Gamma \Uparrow \textbf{name}\ x = t\ \textbf{in}\ B \vdash \cdot \Uparrow \Delta}{\Sigma : \Gamma \Downarrow \textbf{name}\ x = t\ \textbf{in}\ B \vdash \cdot \Downarrow \Delta}$$

Figure : Introduction rules for interpreted constructors

The incorporation of the *naming* context Ψ.

Name binding rules: the variable x is not bound in Σ nor in Ψ.

$$\frac{\Sigma : x := t, \Psi; \Gamma \Uparrow B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Psi; \Gamma \Uparrow \textbf{name}\ x = t\ \textbf{in}\ B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma : x := t, \Psi; \Gamma \Uparrow \cdot \vdash B \Uparrow \cdot}{\Sigma : \Psi; \Gamma \Uparrow \cdot \vdash \textbf{name}\ x = t\ \textbf{in}\ B \Uparrow \cdot}$$

$$\frac{\Sigma : x := t, \Psi; \Gamma \Downarrow \cdot \vdash B \Downarrow \cdot}{\Sigma : \Psi; \Gamma \Downarrow \cdot \vdash \textbf{name}\ x = t\ \textbf{in}\ B \Downarrow \cdot} \qquad \frac{\Sigma : x := t, \Psi; \Gamma \Downarrow B \vdash \cdot \Downarrow E}{\Sigma : \Psi; \Gamma \Downarrow \textbf{name}\ x = t\ \textbf{in}\ B \vdash \cdot \Downarrow E}$$

Positive phase quantifier rules

$$\frac{\Sigma, \Sigma(\Psi) \Uparrow \cdot \vdash t : \tau \Uparrow \cdot \quad \Sigma : \Psi; \Gamma \Downarrow [t/x]B \vdash \cdot \Downarrow E}{\Sigma : \Psi; \Gamma \Downarrow \forall x_\tau . B \vdash \cdot \Downarrow E}$$

$$\frac{\Sigma, \Sigma(\Psi) \Uparrow \cdot \vdash t : \tau \Uparrow \cdot \quad \Sigma : \Psi; \Gamma \Downarrow \cdot \vdash [t/x]B \Downarrow \cdot}{\Sigma : \Psi; \Gamma \Downarrow \cdot \vdash \exists x_\tau . B \Downarrow \cdot}$$