Linear Dependent Types and Relative Completeness

Ugo Dal Lago
(Joint Work with Marco Gaboardi)

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

LICS 2011, Toronto, June 22nd 2011

Part I

Program Logics, Type Systems, and Relative Completeness

Floyd-Hoare Logics

- Judgments:

$$\begin{tikzcd}
& \{\mathrm{P}\}\mathrm{C}\{\mathrm{Q}\} \arrow[dl, dotted] \arrow[d, dotted] \arrow[dr, dotted] & \\
\text{precondition} & \text{program} & \text{postcondition}
\end{tikzcd}$$

- Some rules:

$$\frac{}{\{P[E/x]\}\ x := E\ \{P\}} \qquad \frac{}{\{P\}\ \mathbf{skip}\ \{P\}}$$

$$\frac{\{P\}\ C\ \{Q\} \quad \{Q\}\ D\ \{R\}}{\{P\}\ C; D\ \{R\}}$$

$$\frac{R \Rightarrow P \quad \{P\}\ C\ \{Q\} \quad Q \Rightarrow S}{\{R\}\ C\ \{S\}}$$

Floyd-Hoare Logics

- Judgments:

$$\begin{tikzcd}
& \{\mathrm{P}\}\mathrm{C}\{\mathrm{Q}\} \arrow[dl, dotted] \arrow[dd, dotted] \arrow[dr, dotted] & \\
\text{precondition} & & \text{postcondition} \\
& \text{program} &
\end{tikzcd}$$

- Some rules:

$$\frac{}{\{P[E/x]\}\ x := E\ \{P\}} \qquad \frac{}{\{P\}\ \textbf{skip}\ \{P\}}$$

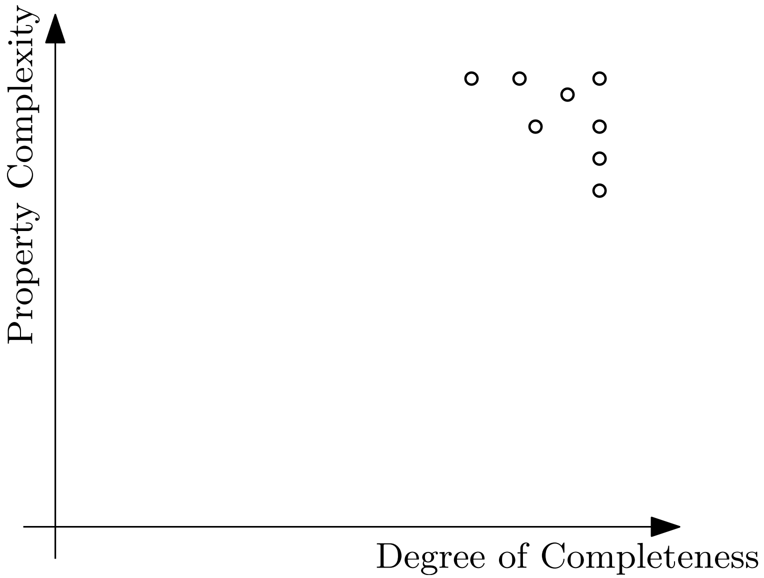
$$\frac{\{P\}\ C\ \{Q\} \quad \{Q\}\ D\ \{R\}}{\{P\}\ C; D\ \{R\}}$$

$$\frac{R \Rightarrow P \quad \{P\}\ C\ \{Q\} \quad Q \Rightarrow S}{\{R\}\ C\ \{S\}}$$

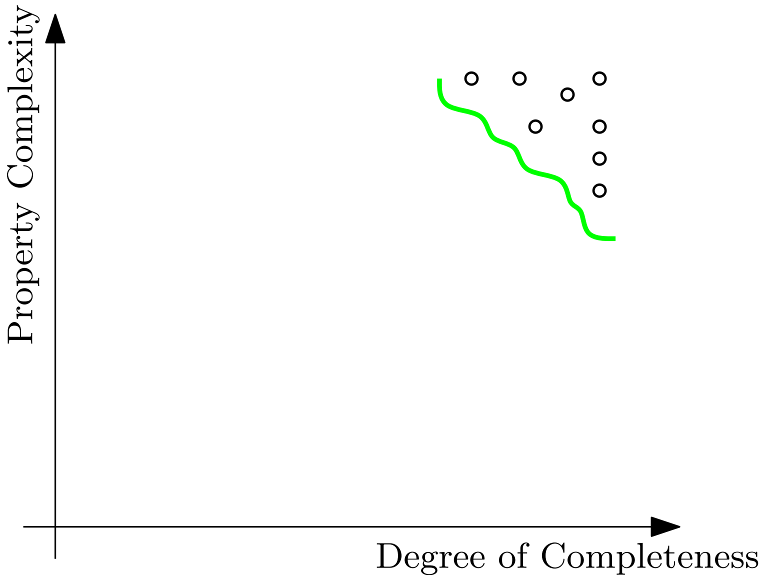
Relative Completeness

- The axiom system is sound.
  - If true formulas of PA are used as side-conditions.
- It's also **relatively complete** [Cook78].
  - All true assertions can be derived if *all true* PA formulas can be used as side-conditions.
- Concrete axiom systems can be derived by throwing in a concrete sound formal system $\mathcal{F}$ for PA.
  - They are sound.
  - They are incomplete, due to Gödel incompleteness.
  - $\mathcal{F}$ is **solely responsible** for their incompleteness.
- A variety of FH logics enjoy the properties above.
  - Including some for higher-order programs [Honda2000]...
  - ... and some in which the complexity of programs and not only their extensional behavior is taken into account.

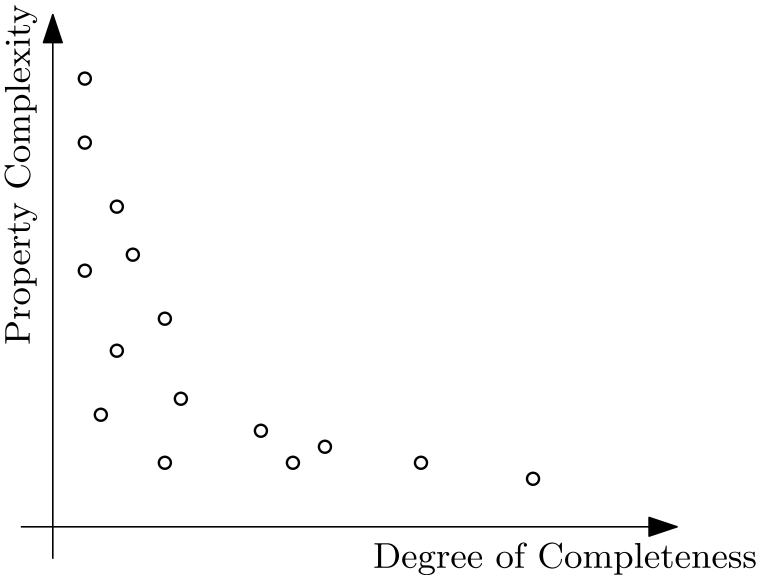
Program Logics



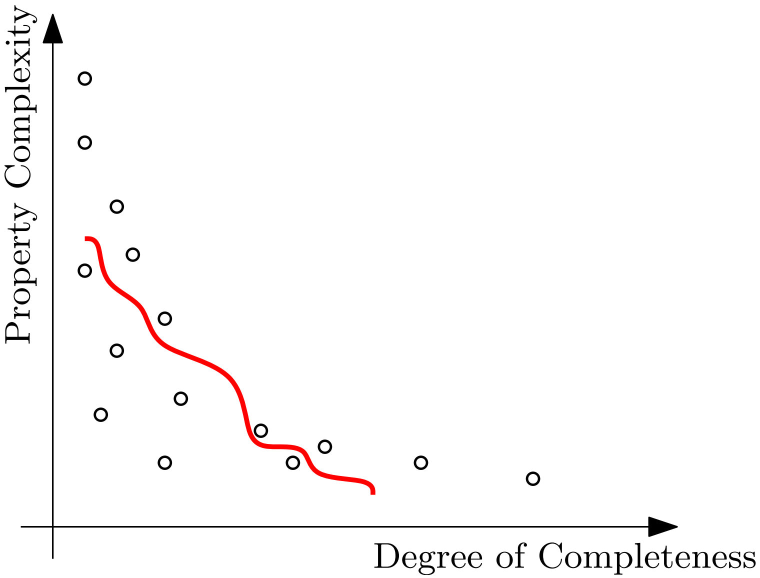
Program Logics



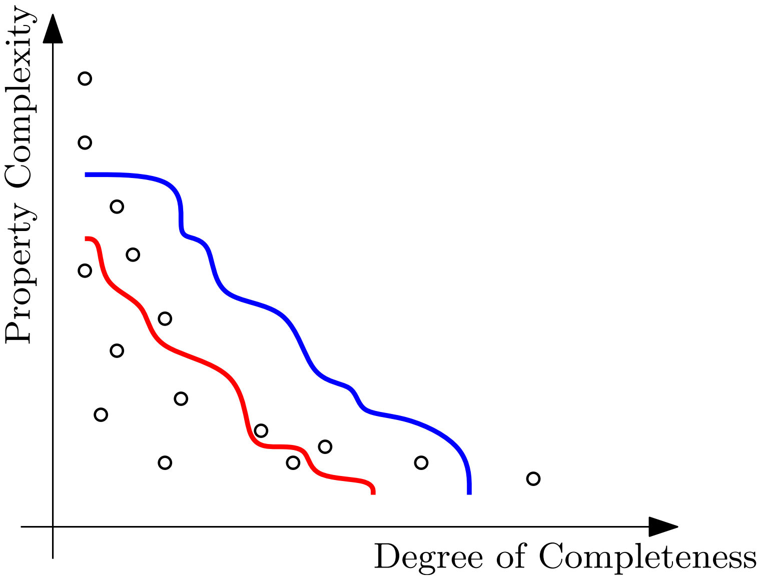
Type Systems



Type Systems



Type Systems



Some Examples

- **Simply Types**
  - “Well-typed programs do not go wrong”.
  - Type inference and type checking are often decidable.
- **Dependent Types**
  - Type checking is decidable.
  - Interesting, extensional properties can be specified.
- **Intersection Types**
  - Sound and complete for termination.
  - Type inference is not decidable.
  - Studying programs as *functions* requires considering an **infinite family** of type derivations.

A Notable Exception: Bounded Linear Logic

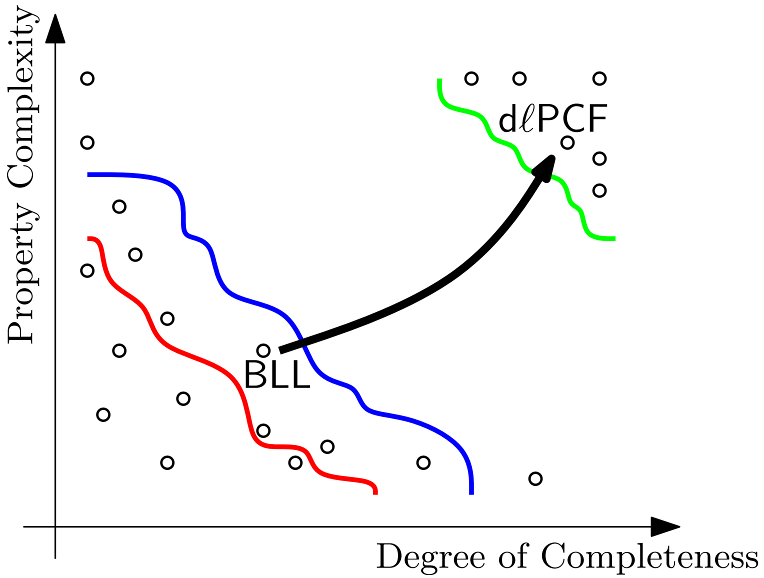
- One of the earliest examples of a system capturing polynomial time **functions** [GSS1992].
  - Extensionally!
  - For every polytime function there is **at least one** proof in BLL computing it.
- Types:

$$A ::= \alpha(p_1, \ldots, p_n) \mid A \otimes A \mid A \multimap A \mid \forall \alpha.A \mid !_{x<p}A$$

- How many "polytime proofs" does BLL capture?
  - There's evidence they are **many** [DLHofmann2010].
- Type checking can be **problematic**. As an example:

$$\frac{\Gamma, !_{x<p}A, !_{y<q}A\{p+y/x\} \vdash B \quad p+q \leqslant r}{\Gamma, !_{x<r}A \vdash B}\ X$$

This Work



Why?

- dℓPCF captures both:
    - **Extensional properties of programs**: what function a program computes.
    - **Intensional properties of programs**: the time complexity of programs.
- Implicit Computational Complexity
    - Many type-theoretical characterizations of complexity classes.
    - Most of them have decidable type inference...
    - ... and poor expressive power.
- **Idea**: drop decidability constraints, and concentrate on expressivity.
    - Recover decidability by considering proper fragments.

Why?

- dℓPCF captures both:
  - **Extensional properties of programs**: what function a program computes.
  - **Intensional properties of programs**: the time complexity of programs.
- Implicit Computational Complexity
  - Many type-theoretical characterizations of complexity classes.
  - Most of them have decidable type inference...
  - ... and poor expressive power.
- **Idea**: drop decidability constraints, and concentrate on expressivity.
  - Recover decidability by considering proper fragments.

Why?

- dℓPCF captures both:
    - **Extensional properties of programs**: what function a program computes.
    - **Intensional properties of programs**: the time complexity of programs.
- Implicit Computational Complexity
    - Many type-theoretical characterizations of complexity classes.
    - Most of them have decidable type inference...
    - ... and poor expressive power.
- **Idea**: drop decidability constraints, and concentrate on expressivity.
    - Recover decidability by considering proper fragments.

Why?

- dℓPCF captures both:
  - **Extensional properties of programs**: what function a program computes.
  - **Intensional properties of programs**: the time complexity of programs.
- Implicit Computational Complexity
  - Many type-theoretical characterizations of complexity classes.
  - Most of them have decidable type inference...
  - ... and poor expressive power.
- **Idea**: drop decidability constraints, and concentrate on expressivity.
  - Recover decidability by considering proper fragments.

Why?

- $\mathsf{d\ell PCF}$ captures both:
  - **Extensional properties of programs**: what function a program computes.
  - **Intensional properties of programs**: the time complexity of programs.
- Implicit Computational Complexity
  - Many type-theoretical characterizations of complexity classes.
  - Most of them have decidable type inference...
  - ... and poor expressive power.
- **Idea**: drop decidability constraints, and concentrate on expressivity.
  - Recover decidability by considering proper fragments.

Part II

dℓPCF

dℓPCF: a Bird's Eye View

- A type system for the lambda calculus with constants and full higher-order recursion. (i.e. PCF).
- Greatly inspired by BLL.
- Indices are not necessarily polynomials, but terms from a signature Σ.
  - Symbols in Σ are given a meaning by an equational program $\mathcal{E}$.
  - Side conditions in the form:

$$\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \leqslant \mathrm{J}$$

- Types and modal types are defined as follows:

$$\begin{aligned} \sigma, \tau &::= \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \mid A \multimap \sigma && \text{basic types} \\ A, B &::= [a < \mathrm{I}] \cdot \sigma && \text{modal types} \end{aligned}$$

dℓPCF: a Bird's Eye View

- A type system for the lambda calculus with constants and full higher-order recursion. (i.e. PCF).
- Greatly inspired by BLL.
- Indices are not necessarily polynomials, but terms from a signature Σ.
  - Symbols in Σ are given a meaning by an equational program $\mathcal{E}$.
  - Side conditions in the form:

$$\phi; \mathrm{K}_1 \leqslant \mathrm{H}_1, \ldots, \mathrm{K}_n \leqslant \mathrm{H}_n \models^{\mathcal{E}} \mathrm{I} \leqslant \mathrm{J}$$

- Types and modal types are defined as follows:

$$\begin{array}{rll} \sigma, \tau ::= & \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \mid A \multimap \sigma & \text{basic types} \\ A, B ::= & [a < \mathrm{I}] \cdot \sigma & \text{modal types} \end{array}$$

The Meaning of Types

$$[a < \mathrm{I}] \cdot \sigma \multimap \tau$$

$$\Downarrow$$

$$(\sigma\{0/a\} \otimes \ldots \otimes \sigma\{\mathrm{I} - 1/a\}) \multimap \tau$$

The Meaning of Types

$$[a < \mathrm{I}] \cdot \sigma \multimap \tau$$

$$\Downarrow$$

$$(\sigma\{0/a\} \otimes \ldots \otimes \sigma\{\mathrm{I} - 1/a\}) \multimap \tau$$

dℓPCF: Subtyping

$$\frac{\begin{array}{c}\phi; \Phi \models^{\mathcal{E}} \mathrm{K} \leqslant \mathrm{I} \\ \phi; \Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{H}\end{array}}{\phi; \Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \sqsubseteq \mathtt{Nat}[\mathrm{K}, \mathrm{H}]}$$

$$\frac{\begin{array}{c}\phi; \Phi \vdash^{\mathcal{E}} B \sqsubseteq A \\ \phi; \Phi \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau\end{array}}{\phi; \Phi \vdash^{\mathcal{E}} A \multimap \sigma \sqsubseteq B \multimap \tau}$$

$$\frac{\begin{array}{c}\phi, a; \Phi, a < \mathrm{J} \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau \\ \phi; \Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{I}\end{array}}{\phi; \Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathrm{J}] \cdot \tau}$$

dℓPCF: Subtyping

$$\frac{\phi; \Phi \models^{\mathcal{E}} \mathrm{K} \leqslant \mathrm{I} \qquad \phi; \Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{H}}{\phi; \Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \sqsubseteq \mathtt{Nat}[\mathrm{K}, \mathrm{H}]}$$

$$\frac{\phi; \Phi \vdash^{\mathcal{E}} B \sqsubseteq A \qquad \phi; \Phi \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau}{\phi; \Phi \vdash^{\mathcal{E}} A \multimap \sigma \sqsubseteq B \multimap \tau}$$

$$\frac{\phi, a; \Phi, a < \mathrm{J} \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau \qquad \phi; \Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{I}}{\phi; \Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathrm{J}] \cdot \tau}$$

dℓPCF: Some Rules

Constraints

$$\frac{\phi; \Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathbf{1}] \cdot \tau}{\phi; \Phi; \Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathsf{0}} x : \tau\{\mathsf{0}/a\}}\ V$$

Weight

dℓPCF: Some Rules

$$\frac{\begin{array}{c}\phi;\Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I}+1,\mathrm{J}+1] \sqsubseteq \mathtt{Nat}[\mathrm{K},\mathrm{H}] \\ \phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} t : \mathtt{Nat}[\mathrm{I},\mathrm{J}]\end{array}}{\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} \mathtt{s}(t) : \mathtt{Nat}[\mathrm{K},\mathrm{H}]}\ S$$

dℓPCF: Some Rules

$$\frac{\phi; \Phi; \Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{J}} t : \tau}{\phi; \Phi; \Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} \lambda x.t : [a < \mathrm{I}] \cdot \sigma \multimap \tau}\ L$$

dℓPCF: Some Rules

$$\frac{\begin{array}{c}\phi;\Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \Gamma \uplus \sum_{a<\mathrm{I}} \Delta \\ \phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} t : [a<\mathrm{I}]\cdot\sigma \multimap \tau \\ \phi,a;\Phi,a<\mathrm{I};\Delta \vdash^{\mathcal{E}}_{\mathrm{K}} u : \sigma\end{array}}{\phi;\Phi;\Sigma \vdash^{\mathcal{E}}_{\mathrm{J}+\sum_{a\leqslant\mathrm{I}}\mathrm{K}+\mathrm{I}} tu : \tau}\ A$$

dℓPCF: Some Rules

Sum of Modal Types

$$\dfrac{\begin{array}{c}\phi;\Phi\vdash^{\mathcal{E}}\Sigma\sqsubseteq\Gamma\uplus\sum_{a<\mathrm{I}}\Delta\\ \phi;\Phi;\Gamma\vdash^{\mathcal{E}}_{\mathrm{J}} t:[a<\mathrm{I}]\cdot\sigma\multimap\tau\\ \phi,a;\Phi,a<\mathrm{I};\Delta\vdash^{\mathcal{E}}_{\mathrm{K}} u:\sigma\end{array}}{\phi;\Phi;\Sigma\vdash^{\mathcal{E}}_{\mathrm{J}+\sum_{a\leqslant\mathrm{I}}\mathrm{K}+\mathrm{I}} tu:\tau}\ A$$

dℓPCF: Some Rules

Bounded Sum of Modal Types

$$\frac{\begin{array}{c}\phi;\Phi\vdash^{\mathcal{E}}\Sigma\sqsubseteq\Gamma\uplus\sum_{a<\mathrm{I}}\Delta\\ \phi;\Phi;\Gamma\vdash^{\mathcal{E}}_{\mathrm{J}} t:[a<\mathrm{I}]\cdot\sigma\multimap\tau\\ \phi,a;\Phi,a<\mathrm{I};\Delta\vdash^{\mathcal{E}}_{\mathrm{K}} u:\sigma\end{array}}{\phi;\Phi;\Sigma\vdash^{\mathcal{E}}_{\mathrm{J}+\sum_{a\leqslant\mathrm{I}}\mathrm{K}+\mathrm{I}} tu:\tau}\ A$$

dℓPCF: Intended Meaning

$$a; \emptyset; \emptyset \vdash_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}]$$

What does this mean?

- t computes a function from natural numbers to natural numbers.
- Something **extensional**:
  - On input a natural number n, t returns a natural number $\mathrm{K}\{n/a\}$
- Something more **intensional**:
  - The cost of evaluation of t on an input n is $(\mathrm{I} + \mathrm{J})\{n/a\}$.
- Two questions:
  - Is this **correct**?
  - How many programs can be captured this way?

dℓPCF: Intended Meaning

$$a;\varnothing;\varnothing \vdash_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}]$$

What does this mean?

- t computes a function from natural numbers to natural numbers.
- Something **extensional**:
  - On input a natural number n, t returns a natural number $\mathrm{K}\{n/a\}$
- Something more **intensional**:
  - The cost of evaluation of t on an input n is $(\mathrm{I}+\mathrm{J})\{n/a\}$.
- Two questions:
  - Is this **correct**?
  - How many programs can be captured this way?

dℓPCF: Intended Meaning

$$a;\varnothing;\varnothing \vdash_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}]$$

What does this mean?

- t computes a function from natural numbers to natural numbers.
- Something **extensional**:
  - On input a natural number n, t returns a natural number $\mathrm{K}\{n/a\}$
- Something more **intensional**:
  - The cost of evaluation of t on an input n is $(\mathrm{I}+\mathrm{J})\{n/a\}$.
- Two questions:
  - Is this **correct**?
  - How many programs can be captured this way?

dℓPCF: Intended Meaning

$$a; \varnothing; \varnothing \vdash_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}]$$

What does this mean?

- t computes a function from natural numbers to natural numbers.
- Something **extensional**:
    - On input a natural number n, t returns a natural number $\mathrm{K}\{n/a\}$
- Something more **intensional**:
    - The cost of evaluation of t on an input n is $(\mathrm{I} + \mathrm{J})\{n/a\}$.
- Two questions:
    - Is this **correct**?
    - How many programs can be captured this way?

Intensional Soundness

- A generalization of KAM which takes constants and fixpoints into account.
- Lift the type system to closures, stack and environments.

Lemma (Measure Decreasing)

Suppose $(t, \epsilon, \epsilon) \to^* D \to E$ *and let* D *have weight* I. *Then one of the following holds:*

1. E *has weight* J, $\phi; \Phi \models \mathrm{I} = \mathrm{J}$ *but* $|D| > |E|$;
2. E *has weight* J, $\phi; \Phi \models \mathrm{I} > \mathrm{J}$ *and* $|E| < |D| + |t|$;

Theorem

Let $\emptyset; \emptyset; \emptyset \vdash_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J}, \mathrm{K}]$ *and* $t \Downarrow^n \underline{\mathtt{m}}$. *Then* $n \leqslant |t| \cdot [\![\mathrm{I}]\!]^{\mathcal{E}}_{\rho}$

Intensional Soundness

- A generalization of KAM which takes constants and fixpoints into account.
- Lift the type system to closures, stack and environments.

Lemma (Measure Decreasing)

Suppose $(t, \epsilon, \epsilon) \rightarrow^* D \rightarrow E$ *and let* D *have weight* I. *Then one of the following holds:*

1. E *has weight* J, $\phi; \Phi \models \mathrm{I} = \mathrm{J}$ *but* $|D| > |E|$;
2. E *has weight* J, $\phi; \Phi \models \mathrm{I} > \mathrm{J}$ *and* $|E| < |D| + |t|$;

Theorem

Let $\varnothing; \varnothing; \varnothing \vdash_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J}, \mathrm{K}]$ *and* $t \Downarrow^n \underline{\mathtt{m}}$. *Then* $n \leqslant |t| \cdot [\![\mathrm{I}]\!]^{\mathcal{E}}_{\rho}$

Completeness for Programs

- The following holds only when $\mathcal{E}$ is **universal**.
- $(\!|\sigma|\!)$ is the PCF type underlying σ, i.e. its skeleton.

Lemma (Weighted Subject Expansion)

If D has weight I and type σ and C is typable with type $(\!|\sigma|\!)$. Then, $C \rightarrow D$ implies that C has weight J and type σ, where $\phi; \Phi \models \mathrm{J} \leqslant \mathrm{I} + 1$.

Theorem (Relative Completeness for Programs)

Let t be a PCF program such that $t \Downarrow^n \underline{\mathtt{m}}$. Then, there exist two index terms I and J such that $[\![\mathrm{I}]\!]^{\mathcal{U}} \leqslant n$ and $[\![\mathrm{J}]\!]^{\mathcal{U}} = m$ and such that the term t is typable in $\mathsf{d}\ell\mathsf{PCF}$ as $\emptyset; \emptyset; \emptyset \vdash^{\mathcal{U}}_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J}]$.

Completeness for Programs

- The following holds only when $\mathcal{E}$ is **universal**.
- $(\!|\sigma|\!)$ is the PCF type underlying σ, i.e. its skeleton.

Lemma (Weighted Subject Expansion)

If D has weight I and type σ and C is typable with type $(\!|\sigma|\!)$. Then, $C \to D$ implies that C has weight J and type σ, where $\phi; \Phi \models \mathrm{J} \leqslant \mathrm{I} + 1$.

Theorem (Relative Completeness for Programs)

Let t be a PCF program such that $t \Downarrow^n \underline{\mathtt{m}}$. Then, there exist two index terms I and J such that $[\![\mathrm{I}]\!]^{\mathcal{U}} \leqslant n$ and $[\![\mathrm{J}]\!]^{\mathcal{U}} = m$ and such that the term t is typable in $\mathsf{d}\ell\mathsf{PCF}$ as $\emptyset; \emptyset; \emptyset \vdash^{\mathcal{U}}_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J}]$.

Completeness for Functions

- It strongly relies on the universality of $\mathcal{U}$.
- Suppose that $\{\pi_n\}_{n\in\mathbb{N}}$ is an r.e. family of type derivations:
  - For the same term t;
  - Having the same PCF skeleton (as type derivations);

  Then we can turn them into a **single**, parametric type derivation.

Theorem (Relative Completeness for Functions)

Suppose that t is a PCF *term such that* $\vdash t : \mathtt{Nat} \rightarrow \mathtt{Nat}$. *Moreover, suppose that there are two (total and computable) functions* $f, g : \mathbb{N} \rightarrow \mathbb{N}$ *such that* $t\ \underline{\mathtt{n}} \Downarrow^{g(n)} \underline{\mathtt{f(n)}}$. *Then there are terms* $\mathrm{I}, \mathrm{J}, \mathrm{K}$ *with* $[\![\mathrm{I}+\mathrm{J}]\!] \leqslant g$ *and* $[\![\mathrm{K}]\!] = f$, *such that*

$$a; \emptyset; \emptyset \vdash^{\mathcal{U}}_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}].$$

Completeness for Functions

- It strongly relies on the universality of $\mathcal{U}$.
- Suppose that $\{\pi_n\}_{n\in\mathbb{N}}$ is an r.e. family of type derivations:
  - For the same term t;
  - Having the same PCF skeleton (as type derivations);

  Then we can turn them into a **single**, parametric type derivation.

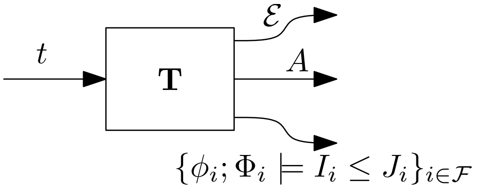
Theorem (Relative Completeness for Functions)

Suppose that t is a PCF term such that $\vdash t : \mathtt{Nat} \to \mathtt{Nat}$. Moreover, suppose that there are two (total and computable) functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $t\ \underline{\mathtt{n}} \Downarrow^{g(n)} \underline{\mathtt{f(n)}}$. Then there are terms $\mathrm{I}, \mathrm{J}, \mathrm{K}$ with $[\![\mathrm{I}+\mathrm{J}]\!] \leqslant g$ and $[\![\mathrm{K}]\!] = f$, such that

$$a; \varnothing; \varnothing \vdash^{\mathcal{U}}_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}].$$

Conclusions

- A relatively complete type system $\mathsf{d}\ell\mathsf{PCF}$.
- Type inference, type checking and derivation checking are undecidable, in general.
  - ... but can become manageable if $\mathcal{E}$ is simple enough.
  - Light Logics!
- **Current work**: relative decidability of type inference.


t
$\mathbf{T}$
$\mathcal{E}$
A
$\{\phi_i; \Phi_i \models I_i \leq J_i\}_{i \in \mathcal{F}}$

Thank you!

Questions?

dℓPCF: Some Rules

$$\frac{\begin{array}{c}\phi, b; \Phi, b < \mathrm{L}; \Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{K}} t : \tau \\ \phi; \Phi \vdash^{\mathcal{E}} \tau\{0/b\} \sqsubseteq \mu \\ \phi, a, b; \Phi, a < \mathrm{I}, b < \mathrm{L} \vdash^{\mathcal{E}} \tau\{\textcircled{$\triangle$}_{b}^{b+1,a}\, \mathrm{I} + 1/b\} \sqsubseteq \sigma \\ \phi; \Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \sum_{b<\mathrm{L}+1} \Gamma \\ \phi; \Phi \models^{\mathcal{E}} \textcircled{$\triangle$}_{b}^{0,1}\, \mathrm{I} \leqslant \mathrm{L}\end{array}}{\phi; \Phi; \Sigma \vdash^{\mathcal{E}}_{\mathrm{L}+\sum_{b<\mathrm{L}} \mathrm{K}} \mathtt{fix}\ x.t : \mu}\ R$$

dℓPCF: Some Rules

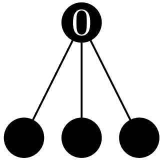
Forest Cardinalities

$$\frac{\begin{array}{c}\phi, b; \Phi, b < \mathrm{L}; \Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{K}} t : \tau \\ \phi; \Phi \vdash^{\mathcal{E}} \tau\{0/b\} \sqsubseteq \mu \\ \phi, a, b; \Phi, a < \mathrm{I}, b < \mathrm{L} \vdash^{\mathcal{E}} \tau\{\bigcirc\!\!\!\!\triangle_{b}^{b+1,a}\, \mathrm{I} + 1/b\} \sqsubseteq \sigma \\ \phi; \Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \sum_{b<\mathrm{L}+1} \Gamma \\ \phi; \Phi \models^{\mathcal{E}} \bigcirc\!\!\!\!\triangle_{b}^{0,1}\, \mathrm{I} \leqslant \mathrm{L}\end{array}}{\phi; \Phi; \Sigma \vdash^{\mathcal{E}}_{\mathrm{L}+\sum_{b<\mathrm{L}}\mathrm{K}} \mathtt{fix}\ x.t : \mu}\ R$$

Forest Cardinalities

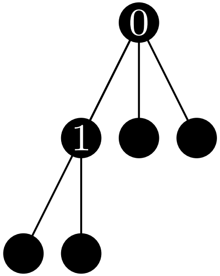
$$I\{0/a\} = 3$$

Forest Cardinalities



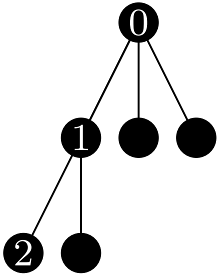
$$I\{1/a\} = 2$$

Forest Cardinalities



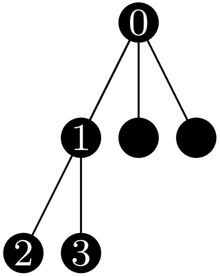
$$I\{2/a\} = 0$$

Forest Cardinalities



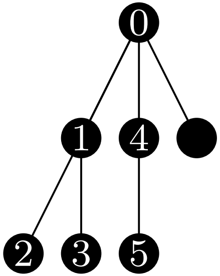
$$I\{3/a\} = 0$$

Forest Cardinalities



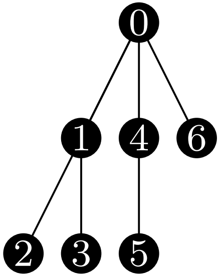
$$I\{4/a\} = 1$$

Forest Cardinalities



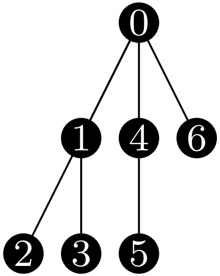
$$I\{5/a\} = 0$$

Forest Cardinalities



$$I\{6/a\} = 0$$

Forest Cardinalities



$$I\{6/a\} = 0$$

I = 7