A Gentle Introduction to Semantic Subtyping

Giuseppe Castagna   Alain Frisch

CNRS
Département d'Informatique
École Normale Supérieure de Paris

INRIA
Roquencourt

ICALP/PPDP – Lisboa, July 2005

Outline

1. Motivations and goals.
2. Semantic subtyping.
3. Subtyping Algorithms.
4. Application to a language.
5. Extensions.

Outline

1. Motivations and goals.
2. Semantic subtyping.
3. Subtyping Algorithms.
4. Application to a language.
5. Extensions.

Outline

1. Motivations and goals.
2. Semantic subtyping.
3. Subtyping Algorithms.
4. Application to a language.
5. Extensions.

Outline

1. Motivations and goals.
2. Semantic subtyping.
3. Subtyping Algorithms.
4. Application to a language.
5. Extensions.

Outline

1. Motivations and goals.
2. Semantic subtyping.
3. Subtyping Algorithms.
4. Application to a language.
5. Extensions.

Goal

The goal is to show how to take your favourite type constructors

$$\times,\ \rightarrow,\ \{\ldots\},\ \texttt{chan}(),\ \ldots$$

and add boolean combinators:

$$\vee,\quad \wedge,\quad \neg$$

so that they behave set-theoretically w.r.t. $\leq$

WHY?

Short answer: they are convenient and you need them to program XML in a typed language with **pattern matching**.

Goal

The goal is to show how to take your favourite type constructors

$$\times,\ \rightarrow,\ \{\ldots\},\ \texttt{chan}(),\ \ldots$$

and add boolean combinators:

$$\vee,\ \wedge,\ \neg$$

so that they behave set-theoretically w.r.t. $\leq$

WHY?

Short answer: they are convenient and you need them to program XML in a typed language with **pattern matching**.

Goal

The goal is to show how to take your favourite type constructors

$$\times,\ \rightarrow,\ \{\ldots\},\ \mathtt{chan}(),\ \ldots$$

and add boolean combinators:

$$\vee,\quad \wedge,\quad \neg$$

so that they behave set-theoretically w.r.t. $\leq$

WHY?

Short answer: they are convenient and you need them to program XML in a typed language with **pattern matching**.

Goal

The goal is to show how to take your favourite type constructors

$$\times,\ \to,\ \{\ldots\},\ \texttt{chan}(),\ \ldots$$

and add boolean combinators:

$$\vee,\quad \wedge,\quad \neg$$

so that they behave set-theoretically w.r.t. $\leq$

WHY?

Short answer: they are convenient and you need them to program XML in a typed language with **pattern matching**.

LONGER ANSWER: Patterns and ∨, ∧, ¬ types

Let:
- $t = \{v \mid v$ value of type $t\}$
- $\wr p \int = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge \wr p_1 \int$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \wr p_2 \int \wedge \neg \wr p_1 \int$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   `map` *catalog* `with`
   `x : (Car ∧ (¬Used ∨ Guarantee)) -> x`

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), . . .

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let:
- $t = \{v \mid v$ value of type $t\}$
- $\llparenthesis p \rrparenthesis = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge \llparenthesis p_1 \rrparenthesis$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \llparenthesis p_2 \rrparenthesis \wedge \neg \llparenthesis p_1 \rrparenthesis$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   `map` *catalog* `with`
   `x : (Car` $\wedge$ `(`$\neg$`Used` $\vee$ `Guarantee)) -> x`

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), ...

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let:
- $t = \{v \mid v$ value of type $t\}$
- $\llparenthesis p \rrparenthesis = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge \llparenthesis p_1 \rrparenthesis$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \llparenthesis p_2 \rrparenthesis \wedge \neg \llparenthesis p_1 \rrparenthesis$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   `map` *catalog* `with`

   `x :: (Car` $\wedge$ `(`$\neg$`Used` $\vee$ `Guarantee)) -> x`

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), ...

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let:
- $t = \{v \mid v$ value of type $t\}$
- $\{\!|p|\!\} = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

```
match e with p1 -> e1 | p2 -> e2
```

   - To infer the type t_1 of e_1 we need $t \wedge \{\!| p_1 |\!\}$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \{\!| p_2 |\!\} \wedge \neg \{\!| p_1 |\!\}$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

```
map catalog with
    x : (Car ∧ (¬Used ∨ Guarantee)) -> x
```

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), ...

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let: - $t = \{v \mid v \text{ value of type } t\}$
- $\llparenthesis p \rrparenthesis = \{v \mid v \text{ matches pattern } p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge \llparenthesis p_1 \rrparenthesis$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \llparenthesis p_2 \rrparenthesis \wedge \neg \llparenthesis p_1 \rrparenthesis$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   ```
   map catalog with
       x : (Car ∧ (¬Used ∨ Guarantee)) -> x
   ```

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), . . .

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let: - $⟦t⟧ = \{v \mid v$ value of type $t\}$
- $⦃p⦄ = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

```
match e with p1 -> e1 | p2 -> e2
```

- To infer the type t_1 of e_1 we need $t \wedge ⦃p_1⦄$ (where $e : t$);
- To infer the type t_2 of e_2 we need $t \wedge ⦃p_2⦄ \wedge \neg ⦃p_1⦄$;
- The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

```
map catalog with
    x : (Car ∧ (¬Used ∨ Guarantee)) -> x
```

Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), . . .

LONGER ANSWER: Patterns and ∨, ∧, ¬ types

Let: - $t = \{v \mid v$ value of type $t\}$
- $(\!|p|\!) = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge (\!| p_1 |\!)$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge (\!| p_2 |\!) \wedge \neg (\!| p_1 |\!)$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   `map` *catalog* `with`
   `x : (Car ∧ (¬Used ∨ Guarantee)) -> x`

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), . . .

LONGER ANSWER: Patterns and ∨, ∧, ¬ types

Let: - $t = \{v \mid v$ value of type $t\}$
- $\lbrace\!\!\lbrace p \rbrace\!\!\rbrace = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge \lbrace\!\!\lbrace p_1 \rbrace\!\!\rbrace$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge \lbrace\!\!\lbrace p_2 \rbrace\!\!\rbrace \wedge \neg \lbrace\!\!\lbrace p_1 \rbrace\!\!\rbrace$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   ```
   map catalog with
       x : (Car ∧ (¬Used ∨ Guarantee)) -> x
   ```

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), ...

LONGER ANSWER: Patterns and $\vee$, $\wedge$, $\neg$ types

Let: - $t = \{v \mid v$ value of type $t\}$
- $(\!|p|\!) = \{v \mid v$ matches pattern $p\}$

1. **Useful for typing:**

   `match` e `with` p_1 `->` e_1 `|` p_2 `->` e_2

   - To infer the type t_1 of e_1 we need $t \wedge (\!| p_1 |\!)$ (where $e : t$);
   - To infer the type t_2 of e_2 we need $t \wedge (\!| p_2 |\!) \wedge \neg (\!| p_1 |\!)$;
   - The type of the match is $t_1 \vee t_2$.

2. **Useful for programming:**

   ```
   map catalog with
       x : (Car ∧ (¬Used ∨ Guarantee)) -> x
   ```

   Select in *catalog* all the cars that if they are used then have a guarantee.

3. **Useful for other paradigms:**

   a general technique to add subtyping to different paradigms: functional (e.g. ML), concurrent (e.g. π-calculus), . . .

In details

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g.:

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1 \vee s_2) \rightarrow t \ \gtrless \ (s_1 \rightarrow t) \wedge (s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbb{0} \ | \ \mathbb{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1 \vee s_2) \rightarrow t \ \gtrless \ (s_1 \rightarrow t) \wedge (s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1 \vee s_2) \rightarrow t \quad \gtreqless \quad (s_1 \rightarrow t) \wedge (s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t \ ::= \ B \mid t{\times}t \mid t{\rightarrow}t \mid t{\vee}t \mid t{\wedge}t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1{\vee}s_2) \rightarrow t \quad \gtrless \quad (s_1 \rightarrow t){\wedge}(s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t ::= B \mid t{\times}t \mid t{\rightarrow}t \mid t{\vee}t \mid t{\wedge}t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1{\vee}s_2) \rightarrow t \quad \gtreqless \quad (s_1 \rightarrow t){\wedge}(s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \rightarrow t_1 \leq s_2 \rightarrow t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1{\vee}s_2) \rightarrow t \quad \gtreqless \quad (s_1 \rightarrow t){\wedge}(s_2 \rightarrow t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\to}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

- Handling subtyping without combinators is easy: constructors do not mix, e.g. :

$$\frac{s_2 \leq s_1 \qquad t_1 \leq t_2}{s_1 \to t_1 \leq s_2 \to t_2}$$

- With combinators is much harder: combinators distribute over constructors, e.g.

$$(s_1{\vee}s_2) \to t \quad \gtreqless \quad (s_1 \to t){\wedge}(s_2 \to t)$$

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t ::= B \mid t{\times}t \mid t{\rightarrow}t \mid t{\vee}t \mid t{\wedge}t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$$

- **Not a particularly new idea**. Many attempts (e.g. Aiken&Wimmers, Damm,. . . , Hosoya&Pierce).
- **None fully satisfactory.** (no negation, or no function types, or restrictions on unions and intersections, . . .)
- **Starting point of what follows: the approach of Hosoya&Pierce.**

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$t ::= B \mid t\times t \mid t\rightarrow t \mid t\vee t \mid t\wedge t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$

- **Not a particularly new idea**. Many attempts (e.g. Aiken&Wimmers, Damm,. . . , Hosoya&Pierce).
- **None fully satisfactory.** (no negation, or no function types, or restrictions on unions and intersections, . . .)
- **Starting point of what follows: the approach of Hosoya&Pierce.**

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

In details

$$t ::= B \mid t{\times}t \mid t{\to}t \mid t{\vee}t \mid t{\wedge}t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$$

- **Not a particularly new idea**. Many attempts (e.g. Aiken&Wimmers, Damm,. . . , Hosoya&Pierce).
- **None fully satisfactory.** (no negation, or no function types, or restrictions on unions and intersections, . . .)
- **Starting point of what follows: the approach of Hosoya&Pierce.**

MAIN IDEA

Instead of defining the subtyping relation so that it conforms to the semantic of types, define the semantics of types and derive the subtyping relation.

Semantic subtyping

Semantic subtyping

1. Define a set-theoretic semantics of the types:

$$[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$$

2. Define the subtyping relation as follows:

$$s \leq t \quad \stackrel{def}{\Longleftrightarrow} \quad [\![s]\!] \subseteq [\![t]\!]$$

KEY OBSERVATION 1:

The *model of types* may be independent from a *model of terms*

Hosoya and Pierce use the model of values:

$$[\![t]\!]_{\mathcal{V}} = \{v \mid \vdash v : t\}$$

Ok because the only values of XDuce are XML documents

Semantic subtyping

1. Define a set-theoretic semantics of the types:

$$[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$$

2. Define the subtyping relation as follows:

$$s \leq t \stackrel{def}{\Longleftrightarrow} [\![s]\!] \subseteq [\![t]\!]$$

KEY OBSERVATION 1:

The *model of types* may be independent from a *model of terms*

Hosoya and Pierce use the model of values:

$$[\![t]\!]_{\mathscr{V}} = \{v \mid \ \vdash v : t\}$$

Ok because the only values of XDuce are XML documents

Semantic subtyping

1. Define a set-theoretic semantics of the types:

$$[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$$

2. Define the subtyping relation as follows:

$$s \leq t \quad \stackrel{def}{\Longleftrightarrow} \quad [\![s]\!] \subseteq [\![t]\!]$$

KEY OBSERVATION 1:

The *model of types* may be independent from a *model of terms*

Hosoya and Pierce use the model of values:

$$[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$$

Ok because the only values of XDuce are XML documents

Semantic subtyping

1. Define a set-theoretic semantics of the types:

$$[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$$

2. Define the subtyping relation as follows:

$$s \leq t \quad \stackrel{def}{\Longleftrightarrow} \quad [\![s]\!] \subseteq [\![t]\!]$$

KEY OBSERVATION 1:

The *model of types* may be independent from a *model of terms*

Hosoya and Pierce use the model of values:

$$[\![t]\!]_{\mathscr{V}} = \{v \mid \ \vdash v : t\}$$

Ok because the only values of XDuce are XML documents

Semantic subtyping

1. Define a set-theoretic semantics of the types:

$$[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$$

2. Define the subtyping relation as follows:

$$s \leq t \quad \stackrel{def}{\Longleftrightarrow} \quad [\![s]\!] \subseteq [\![t]\!]$$

KEY OBSERVATION 1:

The *model of types* may be independent from a *model of terms*

Hosoya and Pierce use the model of values:

$$[\![t]\!]_{\mathscr{V}} = \{v \mid \ \vdash v : t\}$$

Ok because the only values of XDuce are XML documents

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

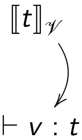
$[\![t]\!]_{\mathscr{V}}$

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

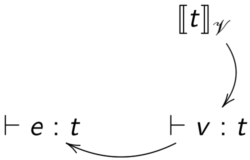


Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

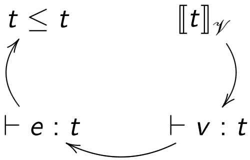


Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid\ \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

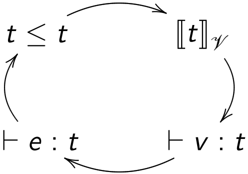


Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

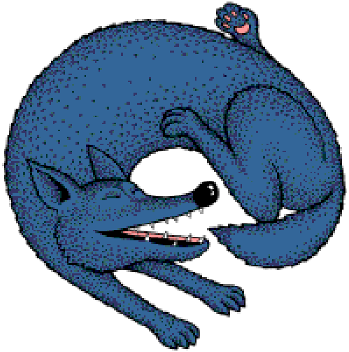


Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid\ \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

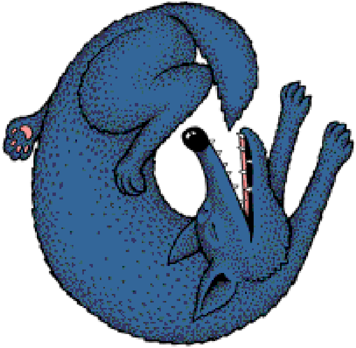
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}} \quad \text{where} \quad [\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$$

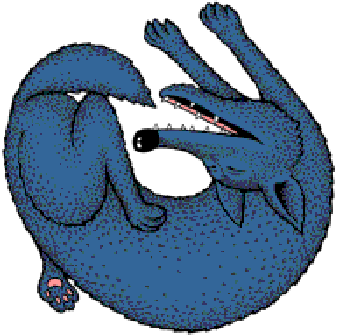
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

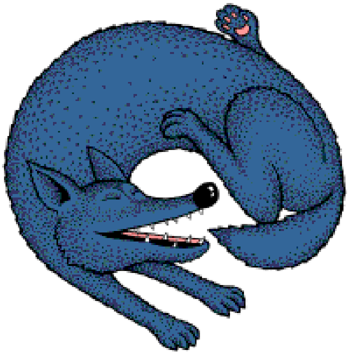
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

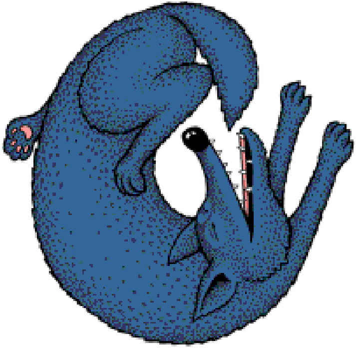
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

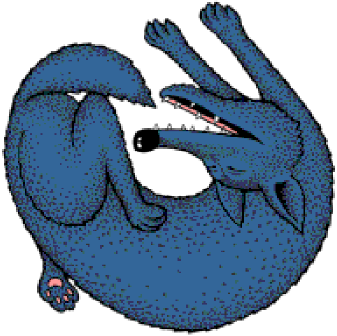
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

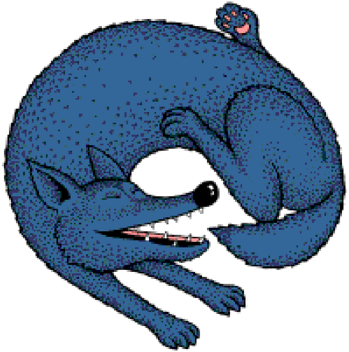
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Circularity

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

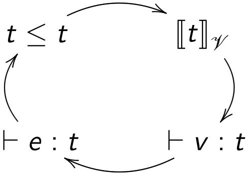
No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



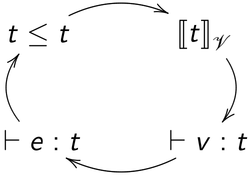
Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

$[\![t]\!]_{\mathscr{D}}$



Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

$[\![t]\!]_{\mathscr{D}}$

$t \leq t$ $\quad\quad$ $[\![t]\!]_{\mathscr{V}}$

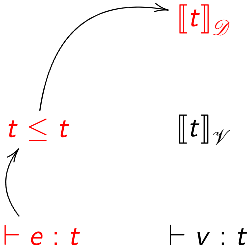
$\vdash e : t$ $\quad\quad$ $\vdash v : t$

Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

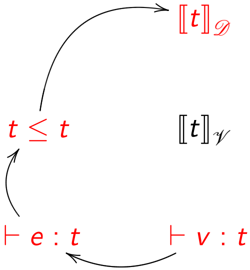


Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid\ \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

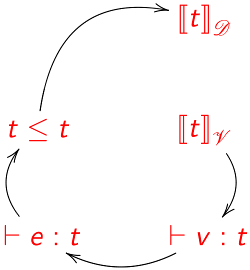


Bootstrap

Model of values

$$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}} \quad \text{where} \quad [\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

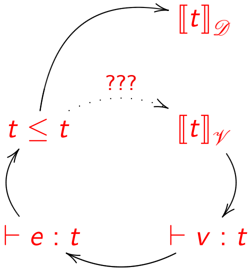


Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined

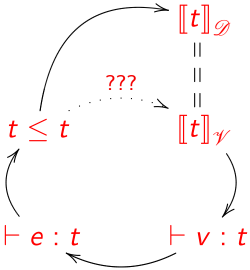


Bootstrap

Model of values

$t \leq s \iff [\![t]\!]_{\mathscr{V}} \subseteq [\![s]\!]_{\mathscr{V}}$ where $[\![t]\!]_{\mathscr{V}} = \{v \mid \vdash v : t\}$

No longer works with arrow types: values are λ-abstractions and need (sub)typing to be defined



Step 1 : Model

Define when $[\![\ ]\!] : \mathbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$
\begin{aligned}
[\![t_1 \vee t_2]\!] &= [\![t_1]\!] \cup [\![t_2]\!] \\
[\![t_1 \wedge t_2]\!] &= [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\neg t]\!] &= \mathscr{D} \setminus [\![t]\!] \\
[\![\mathbf{0}]\!] &= \emptyset \\
[\![\mathbf{1}]\!] &= \mathscr{D}
\end{aligned}
$$

- Hard for constructors:

$$
\begin{aligned}
[\![t_1 \times t_2]\!] &= [\![t_1]\!] \times [\![t_2]\!] \\
[\![t_1 \rightarrow t_2]\!] &= ???
\end{aligned}
$$

Think semantically!

Step 1 : Model

Define when $[\![\;]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$
\begin{array}{lcl}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] \\
[\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\neg t]\!] & = & \mathscr{D} \setminus [\![t]\!] \\
[\![\mathbf{0}]\!] & = & \emptyset \\
[\![\mathbf{1}]\!] & = & \mathscr{D}
\end{array}
$$

- Hard for constructors:

$$
\begin{array}{lcl}
[\![t_1 \times t_2]\!] & = & [\![t_1]\!] \times [\![t_2]\!] \\
[\![t_1 \rightarrow t_2]\!] & = & ???
\end{array}
$$

Think semantically!

Step 1 : Model

Define when $[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$
\begin{array}{lcl}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] \\
[\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\neg t]\!] & = & \mathscr{D} \setminus [\![t]\!] \\
[\![\mathbf{0}]\!] & = & \emptyset \\
[\![\mathbf{1}]\!] & = & \mathscr{D}
\end{array}
$$

- Hard for constructors:

$$
\begin{array}{lcl}
[\![t_1 \times t_2]\!] & = & [\![t_1]\!] \times [\![t_2]\!] \\
[\![t_1 \rightarrow t_2]\!] & = & ???
\end{array}
$$

Think semantically!

Step 1 : Model

Define when $[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$\begin{array}{lcl} [\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] \\ [\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\ [\![\neg t]\!] & = & \mathscr{D} \setminus [\![t]\!] \\ [\![\mathbf{0}]\!] & = & \emptyset \\ [\![\mathbf{1}]\!] & = & \mathscr{D} \end{array}$$

- Hard for constructors:

$$\begin{array}{lcl} [\![t_1 \times t_2]\!] & = & [\![t_1]\!] \times [\![t_2]\!] \\ [\![t_1 \rightarrow t_2]\!] & = & ??? \end{array}$$

Think semantically!

Step 1 : Model

Define when $[\![\ ]\!] : \textbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$
\begin{array}{lcl}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] \\
[\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\neg t]\!] & = & \mathscr{D} \backslash [\![t]\!] \\
[\![\mathbf{0}]\!] & = & \emptyset \\
[\![\mathbf{1}]\!] & = & \mathscr{D}
\end{array}
$$

- Hard for constructors:

$$
\begin{array}{lcl}
[\![t_1 \times t_2]\!] & = & [\![t_1]\!] \times [\![t_2]\!] \\
[\![t_1 \rightarrow t_2]\!] & = & ???
\end{array}
$$

Think semantically!

Step 1 : Model

Define when $[\![\ ]\!] : \mathbf{Types} \longrightarrow \mathscr{P}(\mathscr{D})$ yields a *set-theoretic* model.

- Easy for the combinators:

$$
\begin{array}{lcl}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] \\
[\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\neg t]\!] & = & \mathscr{D} \setminus [\![t]\!] \\
[\![\mathbf{0}]\!] & = & \emptyset \\
[\![\mathbf{1}]\!] & = & \mathscr{D}
\end{array}
$$

- Hard for constructors:

$$
\begin{array}{lcl}
[\![t_1 \times t_2]\!] & = & [\![t_1]\!] \times [\![t_2]\!] \\
[\![t_1 \rightarrow t_2]\!] & = & ???
\end{array}
$$

Think semantically!

Intuition

$$[\![t \longrightarrow s]\!] = ???$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state how types are related rather than what the types are

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ as if equation $(*)$ held, namely

$$[\![t_1 \longrightarrow s_1]\!] \subseteq [\![t_2 \longrightarrow s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \to s]\!] = \{\text{functions from } [\![t]\!] \text{ to } [\![s]\!]\}$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state how types are related rather than what the types are

Accept every $[\![\;]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1 \to s_1]\!] \subseteq [\![t_2 \to s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \to s]\!] = \{f \subseteq \mathscr{D}^2 \mid \forall (d_1, d_2) \in f.\ d_1 \in [\![t]\!] \Rightarrow d_2 \in [\![s]\!]\}$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state how types are related rather than what the types are

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ as if equation $(*)$ held, namely

$$[\![t_1 \to s_1]\!] \subseteq [\![t_2 \to s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \longrightarrow s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (\ \overline{X} \stackrel{\text{def}}{=} \text{complement of } X \ )$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ as if equation $(*)$ held, namely

$$[\![t_1 \longrightarrow s_1]\!] \subseteq [\![t_2 \longrightarrow s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \to s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (*)$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\;]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1 \to s_1]\!] \subseteq [\![t_2 \to s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \longrightarrow s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (*)$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\;]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1 \longrightarrow s_1]\!] \subseteq [\![t_2 \longrightarrow s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t{\longrightarrow}s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (*)$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1{\longrightarrow}s_1]\!] \subseteq [\![t_2{\longrightarrow}s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \to s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (*)$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1 \to s_1]\!] \subseteq [\![t_2 \to s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Intuition

$$[\![t \rightarrow s]\!] = \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \qquad (*)$$

Impossible since it requires $\mathscr{P}(\mathscr{D}^2) \subseteq \mathscr{D}$

KEY OBSERVATION 2:

We need the model to state **how types are related** rather than **what the types are**

Accept every $[\![\ ]\!]$ that behaves w.r.t. $\subseteq$ **as if** equation $(*)$ held, namely

$$[\![t_1 \rightarrow s_1]\!] \subseteq [\![t_2 \rightarrow s_2]\!] \iff \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathscr{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}})$$

and similarly for any boolean combination of arrow types.

Technically . . .

1. **Take** $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ such that

$$[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!] \qquad [\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$$
$$[\![\mathbf{0}]\!] = \emptyset \qquad [\![\mathbf{1}]\!] = \mathscr{D}$$
$$[\![\neg t]\!] = \mathscr{D} \backslash [\![t]\!]$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$\mathbb{E}[\![t_1 \times t_2]\!] \stackrel{\text{def}}{=} [\![t_1]\!] \times [\![t_2]\!] \subseteq \mathscr{D}^2$$
$$\mathbb{E}[\![t_1 \to t_2]\!] \stackrel{\text{def}}{=} \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) \subseteq \mathscr{P}(\mathscr{D}^2)$$
$$\mathbb{E}[\![t_1 \vee t_2]\!] \stackrel{\text{def}}{=} \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!]$$

[constructor semantics]

Technically . . .

1. **Take** $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ such that

$$
\begin{array}{lll}
[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!] & \quad & [\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\mathbf{0}]\!] = \emptyset & & [\![\mathbf{1}]\!] = \mathscr{D} \\
[\![\neg t]\!] = \mathscr{D} \setminus [\![t]\!] & &
\end{array}
$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$
\begin{array}{lcll}
\mathbb{E}[\![t_1 \times t_2]\!] & \stackrel{\text{def}}{=} & [\![t_1]\!] \times [\![t_2]\!] & \subseteq \mathscr{D}^2 \\
\mathbb{E}[\![t_1 \to t_2]\!] & \stackrel{\text{def}}{=} & \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) & \subseteq \mathscr{P}(\mathscr{D}^2) \\
\mathbb{E}[\![t_1 \vee t_2]\!] & \stackrel{\text{def}}{=} & \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!] & \\
 & \vdots & &
\end{array}
$$

[constructor semantics]

3. **Model:** Instead of requiring $[\![t]\!] = \mathbb{E}[\![t]\!]$, accept $[\![\_]\!]$ if

$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

(which is equivalent to $[\![s]\!] \subseteq [\![t]\!] \iff \mathbb{E}[\![s]\!] \subseteq \mathbb{E}[\![t]\!]$)

Technically . . .

1. **Take** $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ such that

$$
\begin{array}{lcl@{\qquad}lcl}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] & [\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\mathbf{0}]\!] & = & \emptyset & [\![\mathbf{1}]\!] & = & \mathscr{D} \\
[\![\neg t]\!] & = & \mathscr{D} \backslash [\![t]\!] & & &
\end{array}
$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$
\begin{array}{lcl@{\qquad}ll}
\mathbb{E}[\![t_1 \times t_2]\!] & \stackrel{\text{def}}{=} & [\![t_1]\!] \times [\![t_2]\!] & \subseteq & \mathscr{D}^2 \\
\mathbb{E}[\![t_1 \to t_2]\!] & \stackrel{\text{def}}{=} & \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) & \subseteq & \mathscr{P}(\mathscr{D}^2) \\
\mathbb{E}[\![t_1 \vee t_2]\!] & \stackrel{\text{def}}{=} & \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!] & & \\
 & \vdots & & &
\end{array}
$$

[constructor semantics]

3. **Model:** Instead of requiring $[\![t]\!] = \mathbb{E}[\![t]\!]$, accept $[\![\_]\!]$ if

$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

(which is equivalent to $[\![s]\!] \subseteq [\![t]\!] \iff \mathbb{E}[\![s]\!] \subseteq \mathbb{E}[\![t]\!]$)

Technically . . .

1. **Take** $[\![\_]\!] : \mathbf{Types} \to \mathscr{P}(\mathscr{D})$ such that

$$[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!] \qquad [\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$$
$$[\![\mathbf{0}]\!] = \emptyset \qquad [\![\mathbf{1}]\!] = \mathscr{D}$$
$$[\![\neg t]\!] = \mathscr{D} \setminus [\![t]\!]$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!] : \mathbf{Types} \to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$\mathbb{E}[\![t_1 \times t_2]\!] \stackrel{\text{def}}{=} [\![t_1]\!] \times [\![t_2]\!] \subseteq \mathscr{D}^2$$
$$\mathbb{E}[\![t_1 \to t_2]\!] \stackrel{\text{def}}{=} \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) \subseteq \mathscr{P}(\mathscr{D}^2)$$
$$\mathbb{E}[\![t_1 \vee t_2]\!] \stackrel{\text{def}}{=} \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!]$$
$$\vdots$$

[constructor semantics]

3. **Model:** Instead of requiring $[\![t]\!] = \mathbb{E}[\![t]\!]$, accept $[\![\ ]\!]$ if

$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

(which is equivalent to $[\![s]\!] \subseteq [\![t]\!] \iff \mathbb{E}[\![s]\!] \subseteq \mathbb{E}[\![t]\!]$)

Technically . . .

1. **Take** $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ such that

$$
\begin{array}{lll@{\qquad}lll}
[\![t_1 \vee t_2]\!] & = & [\![t_1]\!] \cup [\![t_2]\!] & [\![t_1 \wedge t_2]\!] & = & [\![t_1]\!] \cap [\![t_2]\!] \\
[\![\mathbb{0}]\!] & = & \emptyset & [\![\mathbb{1}]\!] & = & \mathscr{D} \\
[\![\neg t]\!] & = & \mathscr{D} \backslash [\![t]\!] & & &
\end{array}
$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$
\begin{array}{lcll}
\mathbb{E}[\![t_1 \times t_2]\!] & \stackrel{\text{def}}{=} & [\![t_1]\!] \times [\![t_2]\!] & \subseteq \mathscr{D}^2 \\
\mathbb{E}[\![t_1 \to t_2]\!] & \stackrel{\text{def}}{=} & \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) & \subseteq \mathscr{P}(\mathscr{D}^2) \\
\mathbb{E}[\![t_1 \vee t_2]\!] & \stackrel{\text{def}}{=} & \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!] & \\
 & \vdots & &
\end{array}
$$

[constructor semantics]

3. **Model:** Instead of requiring $[\![t]\!] = \mathbb{E}[\![t]\!]$, accept $[\![\ ]\!]$ if

$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

(which is equivalent to $[\![s]\!] \subseteq [\![t]\!] \iff \mathbb{E}[\![s]\!] \subseteq \mathbb{E}[\![t]\!]$)

Technically . . .

1. **Take** $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ such that

$$[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!] \qquad [\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$$
$$[\![\mathbf{0}]\!] = \emptyset \qquad [\![\mathbf{1}]\!] = \mathscr{D}$$
$$[\![\neg t]\!] = \mathscr{D} \backslash [\![t]\!]$$

[combinator semantics]

2. **Define** $\mathbb{E}[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D}^2 + \mathscr{P}(\mathscr{D}^2))$ as follows

$$\mathbb{E}[\![t_1 \times t_2]\!] \stackrel{\text{def}}{=} [\![t_1]\!] \times [\![t_2]\!] \subseteq \mathscr{D}^2$$
$$\mathbb{E}[\![t_1 \to t_2]\!] \stackrel{\text{def}}{=} \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}}) \subseteq \mathscr{P}(\mathscr{D}^2)$$
$$\mathbb{E}[\![t_1 \vee t_2]\!] \stackrel{\text{def}}{=} \mathbb{E}[\![t_1]\!] \cup \mathbb{E}[\![t_2]\!]$$
$$\vdots$$

[constructor semantics]

3. **Model:** Instead of requiring $[\![t]\!] = \mathbb{E}[\![t]\!]$, accept $[\![\ ]\!]$ if

$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

(which is equivalent to $[\![s]\!] \subseteq [\![t]\!] \iff \mathbb{E}[\![s]\!] \subseteq \mathbb{E}[\![t]\!]$)

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but . . .

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!]$: **Types** $\rightarrow \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![]\!]$ such that $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$?

YES: an example within two slides

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but ...

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![\,]\!]$ such that $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$?

YES: an example within two slides

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!]{=}\emptyset \Leftrightarrow \mathbb{E}[\![t]\!]{=}\emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but ...

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!] :$ **Types** $\to \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![]\!]$ such that $[\![t]\!]{=}\emptyset \Leftrightarrow \mathbb{E}[\![t]\!]{=}\emptyset$?

YES: an example within two slides

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but . . .

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!]$: **Types** $\rightarrow \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![]\!]$ such that $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$?

YES: an example within two slides

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but ...

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!]$: **Types** $\rightarrow \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![]\!]$ such that $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$?

YES: an example within two slides

The main intuition

To characterize $\leq$ all is needed is the test of emptyness

Indeed: $s \leq t \Leftrightarrow [\![s]\!] \subseteq [\![t]\!] \Leftrightarrow [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!] = \varnothing$

Instead of $[\![t]\!] = \mathbb{E}[\![t]\!]$, the weaker $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$ suffices for $\leq$.

$[\![\ ]\!]$ and $\mathbb{E}[\![\ ]\!]$ must have the same zeros

We relaxed our requirement but . . .

DOES A MODEL EXIST?

Is it possible to define $[\![\_]\!]$: **Types** $\to \mathscr{P}(\mathscr{D})$ that satisfies the model conditions, in particular a $[\![]\!]$ such that $[\![t]\!] = \emptyset \Leftrightarrow \mathbb{E}[\![t]\!] = \emptyset$?

YES: an example within two slides

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1 \rightarrow t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic:*
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating:*
   a function in $[\![t \rightarrow s]\!]$ may be not total on $[\![t]\!]$. E.g.
   $$[\![t \rightarrow \mathbb{0}]\!] = \text{functions diverging on } t$$
3. *Overloaded:*
   $$[\![(t_1 \vee t_2) \rightarrow (s_1 \wedge s_2)]\!] \subsetneq [\![(t_1 \rightarrow s_1) \wedge (t_2 \rightarrow s_2)]\!]$$

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1 \longrightarrow t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic*:
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating*:
   a function in $[\![t \longrightarrow s]\!]$ may be not total on $[\![t]\!]$. E.g.
   $$[\![t \longrightarrow \mathbb{0}]\!] = \text{functions diverging on } t$$
3. *Overloaded*:
   $$[\![(t_1 \vee t_2) \longrightarrow (s_1 \wedge s_2)]\!] \subsetneq [\![(t_1 \longrightarrow s_1) \wedge (t_2 \longrightarrow s_2)]\!]$$

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1 \longrightarrow t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic*:
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating*:
   a function in $[\![t \longrightarrow s]\!]$ may be not total on $[\![t]\!]$. E.g.
   $$[\![t \longrightarrow \mathbf{0}]\!] = \text{functions diverging on } t$$
3. *Overloaded*:
   $$[\![(t_1 \vee t_2) \longrightarrow (s_1 \wedge s_2)]\!] \subsetneq [\![(t_1 \longrightarrow s_1) \wedge (t_2 \longrightarrow s_2)]\!]$$

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1{\longrightarrow}t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic*:
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating*:
   a function in $[\![t{\longrightarrow}s]\!]$ may be not total on $[\![t]\!]$. E.g.

   $$[\![t{\longrightarrow}\mathbb{0}]\!] = \text{functions diverging on } t$$

3. *Overloaded*:

   $$[\![(t_1 \vee t_2){\longrightarrow}(s_1 \wedge s_2)]\!] \subsetneq [\![(t_1{\longrightarrow}s_1) \wedge (t_2{\longrightarrow}s_2)]\!]$$

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1 \longrightarrow t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic*:
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating*:
   a function in $[\![t \longrightarrow s]\!]$ may be not total on $[\![t]\!]$. E.g.
   $$[\![t \longrightarrow \mathbf{0}]\!] = \text{functions diverging on } t$$
3. *Overloaded*:
   $$[\![(t_1 \vee t_2) \longrightarrow (s_1 \wedge s_2)]\!] \subsetneq [\![(t_1 \longrightarrow s_1) \wedge (t_2 \longrightarrow s_2)]\!]$$

The role of $\mathbb{E}[\![\,]\!]$

$\mathbb{E}[\![\,]\!]$ characterizes the behavior of types (for what it concerns $\leq$ one can consider $[\![t]\!] = \mathbb{E}[\![t]\!]$): it depends on the language the types are intended for.

Variations are possible. Our choice

$$\mathbb{E}[\![t_1 \to t_2]\!] = \mathscr{P}(\overline{[\![t_1]\!] \times \overline{[\![t_2]\!]}})$$

accounts for languages that are:

1. *Non-deterministic*:
   Admits functions in which (d, d_1) and (d, d_2) with $d_1 \neq d_2$.
2. *Non-terminating*:
   a function in $[\![t \to s]\!]$ may be not total on $[\![t]\!]$. E.g.
   $$[\![t \to \mathbf{0}]\!] = \text{functions diverging on } t$$
3. *Overloaded*:
   $$[\![(t_1 \vee t_2) \to (s_1 \wedge s_2)]\!] \subsetneq [\![(t_1 \to s_1) \wedge (t_2 \to s_2)]\!]$$

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define
$$s \leq_{\mathscr{B}} t \quad \iff \quad [\![s]\!]_{\mathscr{B}} \subseteq [\![t]\!]_{\mathscr{B}}$$
3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it
$$\Gamma \vdash_{\mathscr{B}} e : t$$
4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \vdash_{\mathscr{B}} v : t\}$ and
$$s \leq_{\mathscr{V}} t \quad \iff \quad [\![s]\!]_{\mathscr{V}} \subseteq [\![t]\!]_{\mathscr{V}}$$
5. If $\mathscr{L}$ is "appropriate" ($\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t$) then $[\![]\!]_{\mathscr{V}}$ is a model and
$$s \leq_{\mathscr{B}} t \quad \iff \quad s \leq_{\mathscr{V}} t$$

The circle is closed

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define

$$s \leq_{\mathscr{B}} t \quad \iff \quad [\![s]\!]_{\mathscr{B}} \subseteq [\![t]\!]_{\mathscr{B}}$$

3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it

$$\Gamma \vdash_{\mathscr{B}} e : t$$

4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \vdash_{\mathscr{B}} v : t\}$ and

$$s \leq_{\mathscr{V}} t \quad \iff \quad [\![s]\!]_{\mathscr{V}} \subseteq [\![t]\!]_{\mathscr{V}}$$

5. If $\mathscr{L}$ is "appropriate" ($\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t$) then $[\![]\!]_{\mathscr{V}}$ is a model and

$$s \leq_{\mathscr{B}} t \quad \iff \quad s \leq_{\mathscr{V}} t$$

The circle is closed

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define

$$s \leq_{\mathscr{B}} t \quad\iff\quad [\![s]\!]_{\mathscr{B}} \subseteq [\![t]\!]_{\mathscr{B}}$$

3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it

$$\Gamma \vdash_{\mathscr{B}} e : t$$

4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \vdash_{\mathscr{B}} v : t\}$ and

$$s \leq_{\mathscr{V}} t \quad\iff\quad [\![s]\!]_{\mathscr{V}} \subseteq [\![t]\!]_{\mathscr{V}}$$

5. If $\mathscr{L}$ is "appropriate" $(\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t)$ then $[\![]\!]_{\mathscr{V}}$ is a model and

$$s \leq_{\mathscr{B}} t \quad\iff\quad s \leq_{\mathscr{V}} t$$

The circle is closed

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define
$$s \leq_{\mathscr{B}} t \quad\iff\quad [\![s]\!]_{\mathscr{B}} \subseteq [\![t]\!]_{\mathscr{B}}$$
3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it
$$\Gamma \vdash_{\mathscr{B}} e : t$$
4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \vdash_{\mathscr{B}} v : t\}$ and
$$s \leq_{\mathscr{V}} t \quad\iff\quad [\![s]\!]_{\mathscr{V}} \subseteq [\![t]\!]_{\mathscr{V}}$$
5. If $\mathscr{L}$ is "appropriate" $(\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t)$ then $[\![]\!]_{\mathscr{V}}$ is a model and
$$s \leq_{\mathscr{B}} t \quad\iff\quad s \leq_{\mathscr{V}} t$$

The circle is closed

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define
$$s \leq_{\mathscr{B}} t \qquad \iff \qquad [\![s]\!]_{\mathscr{B}} \subseteq [\![t]\!]_{\mathscr{B}}$$
3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it
$$\Gamma \vdash_{\mathscr{B}} e : t$$
4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \ \vdash_{\mathscr{B}} v : t\}$ and
$$s \leq_{\mathscr{V}} t \qquad \iff \qquad [\![s]\!]_{\mathscr{V}} \subseteq [\![t]\!]_{\mathscr{V}}$$
5. If $\mathscr{L}$ is "appropriate" $(\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t)$ then $[\![]\!]_{\mathscr{V}}$ is a model and
$$s \leq_{\mathscr{B}} t \qquad \iff \qquad s \leq_{\mathscr{V}} t$$

The circle is closed

Closing the circle

1. Take any model $(\mathscr{B}, [\![]\!]_{\mathscr{B}})$ to bootstrap the definition.
2. Define
$$s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad [\![s]\!]_{\mathscr{B}} \ \subseteq \ [\![t]\!]_{\mathscr{B}}$$
3. Take any "appropriate" language $\mathscr{L}$ and use $\leq_{\mathscr{B}}$ to type it
$$\Gamma \vdash_{\mathscr{B}} e : t$$
4. Define a new interpretation $[\![t]\!]_{\mathscr{V}} = \{v \in \mathscr{V} \mid \ \vdash_{\mathscr{B}} v : t\}$ and
$$s \leq_{\mathscr{V}} t \qquad \Longleftrightarrow \qquad [\![s]\!]_{\mathscr{V}} \ \subseteq \ [\![t]\!]_{\mathscr{V}}$$
5. If $\mathscr{L}$ is "appropriate" ($\vdash_{\mathscr{B}} v : t \iff \not\vdash_{\mathscr{B}} v : \neg t$) then $[\![]\!]_{\mathscr{V}}$ is a model and
$$s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad s \leq_{\mathscr{V}} t$$

The circle is closed

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $\llbracket\ \rrbracket$ such that $\llbracket t \rrbracket = \emptyset \iff \mathbb{E}\llbracket t \rrbracket = \emptyset$)

YES: take $(\mathscr{U}, \llbracket\ \rrbracket_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $\llbracket\ \rrbracket_{\mathscr{U}}$ is defined as:

$$\llbracket \mathbb{0} \rrbracket_{\mathscr{U}} = \emptyset \qquad \llbracket \mathbb{1} \rrbracket_{\mathscr{U}} = \mathscr{U} \qquad \llbracket \neg t \rrbracket_{\mathscr{U}} = \mathscr{U} \setminus \llbracket t \rrbracket_{\mathscr{U}}$$

$$\llbracket s \vee t \rrbracket_{\mathscr{U}} = \llbracket s \rrbracket_{\mathscr{U}} \cup \llbracket t \rrbracket_{\mathscr{U}} \qquad \llbracket s \wedge t \rrbracket_{\mathscr{U}} = \llbracket s \rrbracket_{\mathscr{U}} \cap \llbracket t \rrbracket_{\mathscr{U}}$$

It is a model: $\mathscr{P}_f(\overline{\llbracket t \rrbracket_{\mathscr{U}} \times \overline{\llbracket s \rrbracket_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{\llbracket t \rrbracket_{\mathscr{U}} \times \overline{\llbracket s \rrbracket_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $\llbracket\ \rrbracket_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbb{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbb{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \Rightarrow t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$

2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \Rightarrow t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$

2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$
2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Exhibit a model

Does a model exists? (i.e. a $[\![\ ]\!]$ such that $[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$)

YES: take $(\mathscr{U}, [\![\ ]\!]_{\mathscr{U}})$ where

1. $\mathscr{U}$ least solution of $X = X^2 + \mathscr{P}_f(X^2)$

2. $[\![\ ]\!]_{\mathscr{U}}$ is defined as:

$$[\![\mathbf{0}]\!]_{\mathscr{U}} = \emptyset \qquad [\![\mathbf{1}]\!]_{\mathscr{U}} = \mathscr{U} \qquad [\![\neg t]\!]_{\mathscr{U}} = \mathscr{U} \setminus [\![t]\!]_{\mathscr{U}}$$

$$[\![s \vee t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cup [\![t]\!]_{\mathscr{U}} \qquad [\![s \wedge t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \cap [\![t]\!]_{\mathscr{U}}$$

$$[\![s \times t]\!]_{\mathscr{U}} = [\![s]\!]_{\mathscr{U}} \times [\![t]\!]_{\mathscr{U}} \qquad [\![t \to s]\!]_{\mathscr{U}} = \mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}})$$

It is a model: $\mathscr{P}_f(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing \iff \mathscr{P}(\overline{[\![t]\!]_{\mathscr{U}} \times \overline{[\![s]\!]_{\mathscr{U}}}}) = \varnothing$

It is the **best** model: for any other model $[\![\ ]\!]_{\mathscr{D}}$

$$t_1 \leq_{\mathscr{D}} t_2 \quad \Rightarrow \quad t_1 \leq_{\mathscr{U}} t_2$$

Subtyping Algorithms.

Canonical forms

Every (recursive) type

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

is equivalent (semantically, that is w.r.t. $\leq$) to a type of the form:

$$\bigvee_{(P,N)\in\Pi}\Big(\big(\bigwedge_{s\times t\in P} s{\times}t\big)\wedge\big(\bigwedge_{s\times t\in N}\neg(s{\times}t)\big)\Big)\ \bigvee_{(P,N)\in\Sigma}\Big(\big(\bigwedge_{s\rightarrow t\in P} s{\rightarrow}t\big)\wedge\big(\bigwedge_{s\rightarrow t\in N}\neg(s{\rightarrow}t)\big)\Big)$$

1. Put it in disjunctive normal form, e.g.

   $$(a_1\wedge a_2\wedge\neg a_3)\vee(a_4\wedge\neg a_5)\vee(\neg a_6\wedge\neg a_7)\vee(a_8\wedge a_9)$$

2. Transform to have only homogeneous intersections, e.g.

   $$((s_1\times t_1)\wedge\neg(s_2\times t_2))\ \vee\ (\neg(s_3\rightarrow t_3)\wedge\neg(s_4\rightarrow t_4))\ \vee\ (s_5\times t_5)$$

3. Group negative and positive atoms in the intersections:

Canonical forms

Every (recursive) type

$$t \ ::= \ B \ | \ t{\times}t \ | \ t{\rightarrow}t \ | \ t{\vee}t \ | \ t{\wedge}t \ | \ \neg t \ | \ \mathbb{0} \ | \ \mathbb{1}$$

is equivalent (semantically, that is w.r.t. $\leq$) to a type of the form:

$$\bigvee_{(P,N)\in\Pi}((\bigwedge_{s\times t\in P} s{\times}t)\wedge(\bigwedge_{s\times t\in N}\neg(s{\times}t)))\ \bigvee_{(P,N)\in\Sigma}((\bigwedge_{s\rightarrow t\in P} s{\rightarrow}t)\wedge(\bigwedge_{s\rightarrow t\in N}\neg(s{\rightarrow}t)))$$

1. Put it in disjunctive normal form, e.g.
   $$(a_1\wedge a_2\wedge\neg a_3)\vee(a_4\wedge\neg a_5)\vee(\neg a_6\wedge\neg a_7)\vee(a_8\wedge a_9)$$
2. Transform to have only homogeneous intersections, e.g.
   $$((s_1\times t_1)\wedge\neg(s_2\times t_2))\ \vee\ (\neg(s_3\rightarrow t_3)\wedge\neg(s_4\rightarrow t_4))\ \vee\ (s_5\times t_5)$$
3. Group negative and positive atoms in the intersections:
   $$\bigvee_{(P,N)\in S}((\bigwedge_{a\in P} a)\wedge(\bigwedge_{a\in N}\neg a))$$

Canonical forms

Every (recursive) type

$$t ::= B \mid t{\times}t \mid t{\to}t \mid t{\vee}t \mid t{\wedge}t \mid \neg t \mid \mathbf{0} \mid \mathbf{1}$$

is equivalent (semantically, that is w.r.t. $\leq$) to a type of the form:

$$\bigvee_{(P,N)\in\Pi}((\bigwedge_{s\times t\in P} s{\times}t)\wedge(\bigwedge_{s\times t\in N}\neg(s{\times}t))) \bigvee_{(P,N)\in\Sigma}((\bigwedge_{s\to t\in P} s{\to}t)\wedge(\bigwedge_{s\to t\in N}\neg(s{\to}t)))$$

1. Put it in disjunctive normal form, e.g.
   $(a_1\wedge a_2\wedge\neg a_3)\vee(a_4\wedge\neg a_5)\vee(\neg a_6\wedge\neg a_7)\vee(a_8\wedge a_9)$
2. Transform to have only homogeneous intersections, e.g.
   $((s_1\times t_1)\wedge\neg(s_2\times t_2)) \vee (\neg(s_3\to t_3)\wedge\neg(s_4\to t_4)) \vee (s_5\times t_5)$
3. Group negative and positive atoms in the intersections:

$$\bigvee_{(P,N)\in S}((\bigwedge_{a\in P} a)\wedge(\bigwedge_{a\in N}\neg a))$$

Canonical forms

Every (recursive) type

$$t \ ::= \ B \ | \ t\times t \ | \ t\rightarrow t \ | \ t\vee t \ | \ t\wedge t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

is equivalent (semantically, that is w.r.t. $\leq$) to a type of the form:

$$\bigvee_{(P,N)\in\Pi}((\bigwedge_{s\times t\in P} s\times t)\wedge(\bigwedge_{s\times t\in N}\neg(s\times t))) \ \bigvee_{(P,N)\in\Sigma}((\bigwedge_{s\rightarrow t\in P} s\rightarrow t)\wedge(\bigwedge_{s\rightarrow t\in N}\neg(s\rightarrow t)))$$

1. Put it in disjunctive normal form, e.g.
   $$(a_1\wedge a_2\wedge\neg a_3)\vee(a_4\wedge\neg a_5)\vee(\neg a_6\wedge\neg a_7)\vee(a_8\wedge a_9)$$
2. Transform to have only homogeneous intersections, e.g.
   $$((s_1\times t_1)\wedge\neg(s_2\times t_2)) \vee (\neg(s_3\rightarrow t_3)\wedge\neg(s_4\rightarrow t_4)) \vee (s_5\times t_5)$$
3. Group negative and positive atoms in the intersections:

$$\bigvee_{(P,N)\in S}((\bigwedge_{a\in P} a)\wedge(\bigwedge_{a\in N}\neg a))$$

Canonical forms

Every (recursive) type

$$t \ ::= \ B \ | \ t\times t \ | \ t\rightarrow t \ | \ t\vee t \ | \ t\wedge t \ | \ \neg t \ | \ \mathbf{0} \ | \ \mathbf{1}$$

is equivalent (semantically, that is w.r.t. $\leq$) to a type of the form:

$$\bigvee_{(P,N)\in\Pi}((\bigwedge_{s\times t\in P} s\times t)\wedge(\bigwedge_{s\times t\in N}\neg(s\times t))) \bigvee_{(P,N)\in\Sigma}((\bigwedge_{s\rightarrow t\in P} s\rightarrow t)\wedge(\bigwedge_{s\rightarrow t\in N}\neg(s\rightarrow t)))$$

1. Put it in disjunctive normal form, e.g.
   $$(a_1\wedge a_2\wedge\neg a_3)\vee(a_4\wedge\neg a_5)\vee(\neg a_6\wedge\neg a_7)\vee(a_8\wedge a_9)$$
2. Transform to have only homogeneous intersections, e.g.
   $$((s_1\times t_1)\wedge\neg(s_2\times t_2)) \vee (\neg(s_3\rightarrow t_3)\wedge\neg(s_4\rightarrow t_4)) \vee (s_5\times t_5)$$
3. Group negative and positive atoms in the intersections:

$$\bigvee_{(P,N)\in S}((\bigwedge_{a\in P} a)\wedge(\bigwedge_{a\in N}\neg a))$$

Subtyping decomposition

Some ugly formulas:

$$\bigwedge_{i\in I} t_i{\times}s_i \leq \bigvee_{i\in J} t_i{\times}s_i$$

$$\iff \forall J' \subseteq J.\ \left(\bigwedge_{i\in I} t_i \leq \bigvee_{i\in J'} t_i\right) \text{ or } \left(\bigwedge_{i\in I} s_i \leq \bigvee_{i\in J\setminus J'} s_i\right)$$

$$\bigwedge_{i\in I} t_i{\rightarrow}s_i \leq \bigvee_{i\in J} t_i{\rightarrow}s_i$$

$$\iff \exists j \in J.\forall I' \subseteq I.\ \left(t_j \leq \bigvee_{i\in I'} t_i\right) \text{ or } \left(I' \neq I \text{ et } \bigwedge_{i\in I\setminus I'} s_i \leq s_j\right)$$

Decision procedure

$$s \leq t?$$

Recall that:

$$s \leq t \iff [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \iff [\![s \wedge \neg t]\!] = \varnothing \iff s \wedge \neg t = \mathbf{0}$$

1. Consider $s \wedge \neg t$
2. Put it in canonical form

$$\bigvee_{(P,N)\in\Pi} \Big(\big(\bigwedge_{s\times t\in P} s\times t\big) \wedge \big(\bigwedge_{s\times t\in N} \neg(s\times t)\big)\Big) \quad \bigvee_{(P,N)\in\Sigma} \Big(\big(\bigwedge_{s\to t\in P} s\to t\big) \wedge \big(\bigwedge_{s\to t\in N} \neg(s\to t)\big)\Big)$$

3. Decide (coinductively) whether the two summands are both empty by applying the ugly formulas of the previous slide.

Decision procedure

$$s \leq t?$$

Recall that:

$$s \leq t \iff [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \iff [\![s \wedge \neg t]\!] = \varnothing \iff s \wedge \neg t = \mathbf{0}$$

1. Consider $s\wedge\neg t$
2. Put it in canonical form

$$\bigvee_{(P,N)\in\Pi} ((\bigwedge_{s\times t\in P} s\times t)\wedge(\bigwedge_{s\times t\in N} \neg(s\times t))) \quad \bigvee_{(P,N)\in\Sigma} ((\bigwedge_{s\to t\in P} s\to t)\wedge(\bigwedge_{s\to t\in N} \neg(s\to t)))$$

3. Decide (coinductively) whether the two summands are both empty by applying the ugly formulas of the previous slide.

Decision procedure

$$s \leq t?$$

Recall that:

$$s \leq t \iff [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \iff [\![s \wedge \neg t]\!] = \varnothing \iff s \wedge \neg t = \mathbf{0}$$

1. Consider $s \wedge \neg t$
2. Put it in canonical form

$$\bigvee_{(P,N)\in\Pi} ((\bigwedge_{s\times t\in P} s\times t) \wedge (\bigwedge_{s\times t\in N} \neg(s\times t))) \quad \bigvee_{(P,N)\in\Sigma} ((\bigwedge_{s\to t\in P} s\to t) \wedge (\bigwedge_{s\to t\in N} \neg(s\to t)))$$

3. Decide (coinductively) whether the two summands are both empty by applying the ugly formulas of the previous slide.

Decision procedure

$$s \leq t?$$

Recall that:

$$s \leq t \iff [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \iff [\![s \wedge \neg t]\!] = \varnothing \iff s \wedge \neg t = \mathbf{0}$$

1. Consider $s \wedge \neg t$
2. Put it in canonical form

$$\bigvee_{(P,N)\in\Pi}((\bigwedge_{s\times t\in P} s\times t) \wedge (\bigwedge_{s\times t\in N} \neg(s\times t))) \bigvee_{(P,N)\in\Sigma}((\bigwedge_{s\to t\in P} s\to t) \wedge (\bigwedge_{s\to t\in N} \neg(s\to t)))$$

3. Decide (coinductively) whether the two summands are both empty by applying the ugly formulas of the previous slide.

Decision procedure

$$s \leq t?$$

Recall that:

$$s \leq t \iff [\![s]\!] \cap \overline{[\![t]\!]} = \varnothing \iff [\![s \wedge \neg t]\!] = \varnothing \iff s \wedge \neg t = \mathbf{0}$$

1. Consider $s \wedge \neg t$
2. Put it in canonical form

$$\bigvee_{(P,N)\in\Pi} \Big(\big(\bigwedge_{s\times t\in P} s\times t\big) \wedge \big(\bigwedge_{s\times t\in N} \neg(s\times t)\big)\Big) \bigvee_{(P,N)\in\Sigma} \Big(\big(\bigwedge_{s\to t\in P} s\to t\big) \wedge \big(\bigwedge_{s\to t\in N} \neg(s\to t)\big)\Big)$$

3. Decide (coinductively) whether the two summands are both empty by applying the ugly formulas of the previous slide.

Application to a language.

Language

$e ::=$	x	variable
$\mid$	$\mu f^{(s_1 \to t_1; \ldots; s_n \to t_n)}(x).e$	abstraction, $n \geq 1$
$\mid$	$e_1 e_2$	application
$\mid$	(e_1, e_2)	pair
$\mid$	$\pi_i(e)$	projection, $i = 1, 2$
$\mid$	$(x = e \in t)?e_1 : e_2$	binding type case

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t} \quad (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1{\to}t_1\wedge\ldots\wedge s_n{\to}t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1\to t_1;\ldots;s_n\to t_n)}(x).e : s_1{\to}t_1\wedge\ldots\wedge s_n{\to}t_n} \quad (abstr)$$

(for $s_1 \equiv s\wedge t$, $s_2 \equiv s\wedge\neg t$)

$$\frac{\Gamma \vdash e : s \qquad \Gamma, (x : s_1) \vdash e_1 : t_1 \qquad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 : e_2 : \bigvee_{\{i \mid s_i \not\simeq \mathbf{0}\}} t_i} \quad (typecase)$$

Consider:

$$\mu f^{(\mathrm{Int}\to\mathrm{Int};\mathrm{Bool}\to\mathrm{Bool})}(x).(y = x \in \mathrm{Int})?(y+1):\mathrm{not}(y)$$

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t}\ (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1{\to}t_1 \wedge \ldots \wedge s_n{\to}t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1\to t_1;\ldots;s_n\to t_n)}(x).e : s_1{\to}t_1 \wedge \ldots \wedge s_n{\to}t_n}\ (abstr)$$

(for $s_1 \equiv s \wedge t$, $s_2 \equiv s \wedge \neg t$)

$$\frac{\Gamma \vdash e : s \qquad \Gamma, (x : s_1) \vdash e_1 : t_1 \quad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 : e_2 : \bigvee_{\{i \mid s_i \not\simeq \mathbb{0}\}} t_i}\ (typecase)$$

Consider:

$$\mu f^{(\mathrm{Int}\to\mathrm{Int};\mathrm{Bool}\to\mathrm{Bool})}(x).(y = x \in \mathrm{Int})?(y+1):\mathrm{not}(y)$$

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t} \ (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1 \rightarrow t_1 \wedge \ldots \wedge s_n \rightarrow t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1 \rightarrow t_1; \ldots; s_n \rightarrow t_n)}(x).e : s_1 \rightarrow t_1 \wedge \ldots \wedge s_n \rightarrow t_n} \ (abstr)$$

(for $s_1 \equiv s \wedge t$, $s_2 \equiv s \wedge \neg t$)

$$\frac{\Gamma \vdash e : s \quad \Gamma, (x : s_1) \vdash e_1 : t_1 \quad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 : e_2 : \bigvee_{\{i \mid s_i \not\simeq \mathbf{0}\}} t_i} \ (typecase)$$

Consider:

$$\mu f^{(\text{Int}\rightarrow\text{Int};\text{Bool}\rightarrow\text{Bool})}(x).(y = x \in \text{Int})?(y+1):\text{not}(y)$$

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t}\ (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1{\to}t_1 \wedge \ldots \wedge s_n{\to}t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1\to t_1;\ldots;s_n\to t_n)}(x).e : s_1{\to}t_1 \wedge \ldots \wedge s_n{\to}t_n}\ (abstr)$$

(for $s_1 \equiv s \wedge t$, $s_2 \equiv s \wedge \neg t$)

$$\frac{\Gamma \vdash e : s \qquad \Gamma, (x : s_1) \vdash e_1 : t_1 \quad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 : e_2 : \bigvee_{\{i \mid s_i \not\simeq \mathbf{0}\}} t_i}\ (typecase)$$

Consider:

$$\mu f^{(\mathrm{Int}\to\mathrm{Int};\mathrm{Bool}\to\mathrm{Bool})}(x).(y = x \in \mathrm{Int})?(y+1):\mathrm{not}(y)$$

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t} \quad (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1 \to t_1 \wedge \ldots \wedge s_n \to t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1 \to t_1; \ldots; s_n \to t_n)}(x).e : s_1 \to t_1 \wedge \ldots \wedge s_n \to t_n} \quad (abstr)$$

$$(\text{for } s_1 \equiv s \wedge t,\ s_2 \equiv s \wedge \neg t)$$
$$\frac{\Gamma \vdash e : s \qquad \Gamma, (x : s_1) \vdash e_1 : t_1 \quad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 : e_2 : \bigvee_{\{i \mid s_i \not\simeq \mathbf{0}\}} t_i} \quad (typecase)$$

Consider:

$$\mu f^{(\text{Int} \to \text{Int}; \text{Bool} \to \text{Bool})}(x).(y = x \in \text{Int})?(y + 1) : \text{not}(y)$$

Typing

$$\frac{\Gamma \vdash e : s \leq_{\mathscr{B}} t}{\Gamma \vdash e : t} \; (subsumption)$$

$$\frac{(\forall i)\ \Gamma, (f : s_1{\rightarrow}t_1 \wedge \ldots \wedge s_n{\rightarrow}t_n), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1\rightarrow t_1;\ldots;s_n\rightarrow t_n)}(x).e : s_1{\rightarrow}t_1 \wedge \ldots \wedge s_n{\rightarrow}t_n} \; (abstr)$$

(for $s_1 \equiv s{\wedge}t$, $s_2 \equiv s{\wedge}\neg t$)

$$\frac{\Gamma \vdash e : s \qquad \Gamma, (x : s_1) \vdash e_1 : t_1 \quad \Gamma, (x : s_2) \vdash e_2 : t_2}{\Gamma \vdash (x = e \in t)?e_1 {:} e_2 : \bigvee_{\{i | s_i \not\simeq \mathbf{0}\}} t_i} \; (typecase)$$

Consider:

$$\mu \mathrm{f}^{(\mathrm{Int}\rightarrow\mathrm{Int};\mathrm{Bool}\rightarrow\mathrm{Bool})}(x).(y = x \in \mathrm{Int})?(y+1){:}\mathrm{not}(y)$$

Reduction

$$
\begin{array}{rcll}
(\mu f^{(\dots)}(x).e)v & \to & e[x/v, (\mu f^{(\dots)}(x).e)/f] & \\
(x = v \in t)?e_1 : e_2 & \to & e_1[x/v] & \text{if } v \in [\![t]\!] \\
(x = v \in t)?e_1 : e_2 & \to & e_2[x/v] & \text{if } v \notin [\![t]\!]
\end{array}
$$

where

$$v \;::=\; \mu f^{(\dots)}(x).e \;\mid\; (v,v)$$

And we have

$$s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad s \leq_{\mathscr{V}} t$$

The circle is closed

Reduction

$$
\begin{array}{rcll}
(\mu f^{(\dots)}(x).e)v & \rightarrow & e[x/v,(\mu f^{(\dots)}(x).e)/f] & \\
(x = v \in t)?e_1 : e_2 & \rightarrow & e_1[x/v] & \text{if } v \in [\![t]\!] \\
(x = v \in t)?e_1 : e_2 & \rightarrow & e_2[x/v] & \text{if } v \notin [\![t]\!]
\end{array}
$$

where

$$v \ ::= \ \mu f^{(\dots)}(x).e \ \mid \ (v,v)$$

And we have

$$s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad s \leq_{\mathscr{V}} t$$

The circle is closed

Reduction

$$
\begin{array}{rcll}
(\mu f^{(\dots)}(x).e)v & \rightarrow & e[x/v,(\mu f^{(\dots)}(x).e)/f] & \\
(x=v\in t)?e_1{:}e_2 & \rightarrow & e_1[x/v] & \text{if } v\in[\![t]\!] \\
(x=v\in t)?e_1{:}e_2 & \rightarrow & e_2[x/v] & \text{if } v\notin[\![t]\!]
\end{array}
$$

where

$$v \; ::= \; \mu f^{(\dots)}(x).e \mid (v,v)$$

And we have

$$s \leq_{\mathscr{B}} t \quad \Longleftrightarrow \quad s \leq_{\mathscr{V}} t$$

The circle is closed

Reduction

$$
\begin{array}{rcll}
(\mu f^{(\dots)}(x).e)v & \rightarrow & e[x/v, (\mu f^{(\dots)}(x).e)/f] & \\
(x = v \in t)?e_1 : e_2 & \rightarrow & e_1[x/v] & \text{if } v \in [\![t]\!] \\
(x = v \in t)?e_1 : e_2 & \rightarrow & e_2[x/v] & \text{if } v \notin [\![t]\!]
\end{array}
$$

where

$$
v \; ::= \; \mu f^{(\dots)}(x).e \;\mid\; (v,v)
$$

And we have

$$
s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad s \leq_{\mathscr{V}} t
$$

The circle is closed

Why does it work?

$$s \leq_{\mathscr{B}} t \quad\Longleftrightarrow\quad s \leq_{\mathscr{V}} t \qquad (1)$$

Equation (1) (actually, $\Rightarrow$) states that the language is quite rich, since there always exists a value to separate two distinct types; i.e. its set of values is a model of types with "enough points"

For any model $\mathscr{B}$,

$$s \not\leq_{\mathscr{B}} t \Longrightarrow \text{there exists } v \text{ such that } \vdash v : s \text{ and } \not\vdash v : t$$

In particular, thanks to multiple arrows in λ-abstractions:

$$\bigwedge_{i=1..k} s_i \to t_i \not\leq t$$

then the two types are distinguished by $\mu f^{(s_1 \to t_1; \ldots; s_k \to t_k)}(x).e$

Why does it work?

$$s \leq_{\mathscr{B}} t \qquad \Longleftrightarrow \qquad s \leq_{\mathscr{V}} t \qquad (1)$$

Equation (1) (actually, $\Rightarrow$) states that the language is quite rich, since there always exists a value to separate two distinct types; i.e. its set of values is a model of types with "enough points"

For any model $\mathscr{B}$,

$s \not\leq_{\mathscr{B}} t \Longrightarrow$ there exists v such that $\vdash v : s$ and $\not\vdash v : t$

In particular, thanks to multiple arrows in λ-abstractions:

$$\bigwedge_{i=1..k} s_i \to t_i \not\leq t$$

then the two types are distinguished by $\mu f^{(s_1 \to t_1; \ldots; s_k \to t_k)}(x).e$

Why does it work?

$$s \leq_{\mathscr{B}} t \quad \Longleftrightarrow \quad s \leq_{\mathscr{V}} t \qquad (1)$$

Equation (1) (actually, $\Rightarrow$) states that the language is quite rich, since there always exists a value to separate two distinct types; i.e. its set of values is a model of types with "enough points"

For any model $\mathscr{B}$,

$$s \not\leq_{\mathscr{B}} t \Longrightarrow \text{there exists } v \text{ such that } \vdash v : s \text{ and } \not\vdash v : t$$

In particular, thanks to multiple arrows in λ-abstractions:

$$\bigwedge_{i=1..k} s_i \rightarrow t_i \not\leq t$$

then the two types are distinguished by $\mu f^{(s_1 \rightarrow t_1; \ldots; s_k \rightarrow t_k)}(x).e$

Advantages for the programmer

The programmer does not need to know the gory details. All s/he needs to retain is

1. Types are the set of values of that type
2. Subtyping is set inclusion

Furthermore the property

$$s \not\leq t \Longrightarrow \text{there exists } v \text{ such that } \vdash v : s \text{ and } \not\vdash v : t$$

is fundamental for meaningful error messages:

Exibit the v at issue rather than pointing to the failure of some deduction rule.

Advantages for the programmer

The programmer does not need to know the gory details. All s/he needs to retain is

1. Types are the set of values of that type
2. Subtyping is set inclusion

Furthermore the property

$$s \not\leq t \Longrightarrow \text{there exists } v \text{ such that} \vdash v : s \text{ and } \not\vdash v : t$$

is fundamental for meaningful error messages:

Exibit the v at issue rather than pointing to the failure of some deduction rule.

Advantages for the programmer

The programmer does not need to know the gory details. All s/he needs to retain is

1. Types are the set of values of that type
2. Subtyping is set inclusion

Furthermore the property

$$s \not\leq t \Longrightarrow \text{there exists } v \text{ such that } \vdash v : s \text{ and } \not\vdash v : t$$

is fundamental for meaningful error messages:

Exibit the v at issue rather than pointing to the failure of some deduction rule.

Advantages for the programmer

The programmer does not need to know the gory details. All s/he needs to retain is

1. Types are the set of values of that type
2. Subtyping is set inclusion

Furthermore the property

$$s \not\leq t \implies \text{there exists } v \text{ such that } \vdash v : s \text{ and } \not\vdash v : t$$

is fundamental for meaningful error messages:

Exibit the v at issue rather than pointing to the failure of some deduction rule.

Advantages for the programmer

The programmer does not need to know the gory details. All s/he needs to retain is

1. Types are the set of values of that type
2. Subtyping is set inclusion

Furthermore the property

$$s \not\leq t \implies \text{there exists } v \text{ such that} \vdash v : s \text{ and } \not\vdash v : t$$

is fundamental for meaningful error messages:

Exibit the v at issue rather than pointing to the failure of some deduction rule.

Extensions.

ref types

$$[\![\texttt{ref}\ t]\!] = \{\texttt{ref}\ v \mid v \in [\![t]\!]\}$$

In practice equivalent to

$$[\![\texttt{ref}\ t]\!] = \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \qquad (2)$$

Deduce the subtyping relation

$$\Big(\bigwedge_{\texttt{ref}\ s \in P} \texttt{ref}\ s\Big) \leq \Big(\bigvee_{\texttt{ref}\ t \in N} \texttt{ref}\ t\Big) \iff \begin{array}{l} \exists \texttt{ref}\ s \in P,\ \ s \simeq 0, \text{or} \\ \exists \texttt{ref}\ s_1 \in P,\ \exists \texttt{ref}\ s_2 \in P, s_1 \not\simeq s_2, \text{or} \\ \exists \texttt{ref}\ s \in P,\ \exists \texttt{ref}\ t \in N,\ \ s \simeq t \end{array}$$

ref types

$$[\![\texttt{ref}\ t]\!] = \{\texttt{ref}\ v \mid v \in [\![t]\!]\}$$

In practice equivalent to

$$[\![\texttt{ref}\ t]\!] = \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \qquad (2)$$

Deduce the subtyping relation

$$\Big(\bigwedge_{\texttt{ref}\ s \in P} \texttt{ref}\ s\Big) \leq \Big(\bigvee_{\texttt{ref}\ t \in N} \texttt{ref}\ t\Big) \iff \begin{array}{l} \exists \texttt{ref}\ s \in P,\ \ s \simeq 0, \text{or} \\ \exists \texttt{ref}\ s_1 \in P,\ \exists \texttt{ref}\ s_2 \in P, s_1 \not\simeq s_2, \text{or} \\ \exists \texttt{ref}\ s \in P,\ \exists \texttt{ref}\ t \in N,\ \ s \simeq t \end{array}$$

ref types

$$[\![\texttt{ref}\ t]\!] = \{\texttt{ref}\ v \mid v \in [\![t]\!]\}$$

In practice equivalent to

$$\mathbb{E}[\![\texttt{ref}\ t]\!] = \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \tag{2}$$

Deduce the subtyping relation

$$\Big(\bigwedge_{\texttt{ref}\ s \in P} \texttt{ref}\ s\Big) \leq \Big(\bigvee_{\texttt{ref}\ t \in N} \texttt{ref}\ t\Big) \iff \begin{array}{l} \exists \texttt{ref}\ s \in P,\ \ s \simeq 0, \text{or} \\ \exists \texttt{ref}\ s_1 \in P,\ \exists \texttt{ref}\ s_2 \in P, s_1 \not\simeq s_2, \text{or} \\ \exists \texttt{ref}\ s \in P,\ \exists \texttt{ref}\ t \in N,\ \ s \simeq t \end{array}$$

ref types

$$[\![\texttt{ref}\ t]\!] = \{\texttt{ref}\ v \mid v \in [\![t]\!]\}$$

In practice equivalent to

$$\mathbb{E}[\![\texttt{ref}\ t]\!] = \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \qquad (2)$$

Deduce the subtyping relation

$$\Big(\bigwedge_{\texttt{ref}\ s \in P} \texttt{ref}\ s\Big) \leq \Big(\bigvee_{\texttt{ref}\ t \in N} \texttt{ref}\ t\Big) \iff \begin{array}{l} \exists \texttt{ref}\ s \in P,\ \ s \simeq 0, \text{or} \\ \exists \texttt{ref}\ s_1 \in P,\ \exists \texttt{ref}\ s_2 \in P, s_1 \not\simeq s_2, \text{or} \\ \exists \texttt{ref}\ s \in P,\ \exists \texttt{ref}\ t \in N,\ \ s \simeq t \end{array}$$

lazy types

If we define $t = \mathsf{Int}\times t$ then $t \simeq \mathbf{0}$.

Use $s = \mathsf{Int}\times \mathtt{lazy}\, s$.

$$(\mu f^{(\mathsf{Int}\to s)}(x).(x, \mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\textstyle\bigwedge_{\mathtt{lazy}\, s\in P} \mathtt{lazy}\, s) \leq (\textstyle\bigvee_{\mathtt{lazy}\, t\in N} \mathtt{lazy}\, t) \iff$$
$$\exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P (\textstyle\bigwedge_{\mathtt{lazy}\, s\in P} s) \leq t$$

lazy types

If we define $t = \mathsf{Int}\times t$ then $t \simeq \mathbf{0}$.

Use $s = \mathsf{Int}\times \mathtt{lazy}\, s$.

$$(\mu f^{(\mathsf{Int}\to s)}(x).(x, \mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation
$(\bigwedge_{\mathtt{lazy}\, s\in P} \mathtt{lazy}\, s) \leq (\bigvee_{\mathtt{lazy}\, t\in N} \mathtt{lazy}\, t) \iff$
$\exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P (\bigwedge_{\mathtt{lazy}\, s\in P'} s) \leq t$

lazy types

If we define $t = \mathrm{Int}\times t$ then $t \simeq \mathbf{0}$.

Use $s = \mathrm{Int}\times \mathtt{lazy}\, s$.

$$(\mu f^{(\mathrm{Int}\to s)}(x).(x,\mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\textstyle\bigwedge_{\mathtt{lazy}\, s\in P} \mathtt{lazy}\, s) \leq (\bigvee_{\mathtt{lazy}\, t\in N} \mathtt{lazy}\, t) \iff$$
$$\exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P (\textstyle\bigwedge_{\mathtt{lazy}\, s\in P'} s) \leq t$$

lazy types

If we define $t = \mathsf{Int}\times t$ then $t \simeq \mathbf{0}$.

Use $s = \mathsf{Int}\times \mathtt{lazy}\, s$.

$$(\mu f^{(\mathsf{Int}\to s)}(x).(x,\mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\textstyle\bigwedge_{\mathtt{lazy}\, s\in P} \mathtt{lazy}\, s) \leq (\bigvee_{\mathtt{lazy}\, t\in N} \mathtt{lazy}\, t) \iff \exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P (\bigwedge_{\mathtt{lazy}\, s\in P'} s) \leq t$$

lazy types

If we define $t = \mathrm{Int}{\times}t$ then $t \simeq \mathbf{0}$.

Use $s = \mathrm{Int}{\times}\mathtt{lazy}\, s$.

$$(\boldsymbol{\mu} f^{(\mathrm{Int}\to s)}\mathbf{(}x\mathbf{).}(x,\mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\textstyle\bigwedge_{\mathtt{lazy}\, s\in P} \mathtt{lazy}\, s) \leq (\bigvee_{\mathtt{lazy}\, t\in N} \mathtt{lazy}\, t) \iff$$
$$\exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P(\textstyle\bigwedge_{\mathtt{lazy}\, s\in P'} s) \leq t$$

lazy types

If we define $t = \mathsf{Int} \times t$ then $t \simeq \mathbf{0}$.

Use $s = \mathsf{Int} \times \mathtt{lazy}\, s$.

$$(\boldsymbol{\mu} f^{(\mathsf{Int} \to s)}(x).(x, \mathtt{lazy}\,(f(x+1))))0$$

$$[\![\mathtt{lazy}\, t]\!] = \{\mathtt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\mathtt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$\mathbb{E}[\![\mathtt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\bigwedge_{\mathtt{lazy}\, s \in P} \mathtt{lazy}\, s) \leq (\bigvee_{\mathtt{lazy}\, t \in N} \mathtt{lazy}\, t) \iff$$
$$\exists \mathtt{lazy}\, t \in N : \forall P' \subseteq P (\bigwedge_{\mathtt{lazy}\, s \in P} s) \leq t$$

lazy types

If we define $t = \text{Int}\times t$ then $t \simeq \mathbf{0}$.

Use $s = \text{Int}\times \texttt{lazy}\, s$.

$$(\mu f^{(\text{Int}\to s)}(x).(x,\texttt{lazy}\,(f(x+1))))0$$

$$[\![\texttt{lazy}\, t]\!] = \{\texttt{lazy}\, e \mid e \text{ is closed and } e : t\}$$

But each $\texttt{lazy}\, e$ is identified by all the possible results it can return, namely $\{v \mid e \to^* v\}$, from which we deduce:

$$\mathbb{E}[\![\texttt{lazy}\, t]\!] = \mathscr{P}([\![t]\!])$$

Deduce the subtyping relation

$$(\textstyle\bigwedge_{\texttt{lazy}\, s\in P} \texttt{lazy}\, s) \leq (\bigvee_{\texttt{lazy}\, t\in N} \texttt{lazy}\, t) \iff$$
$$\exists \texttt{lazy}\, t \in N : \forall P' \subseteq P (\textstyle\bigwedge_{\texttt{lazy}\, s\in P} s) \leq t$$

channel types

Really, no time to show it but . . .

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to **π-calculus**. There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(t_1) \vee ch^{+}(t_2) \lneqq ch^{+}(t_1 \vee t_2)$$

[the "not equal" is interesting]

`channel` types

Really, no time to show it but . . .

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to π-**calculus**.
There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(t_1) \vee ch^{+}(t_2) \lneq ch^{+}(t_1 \vee t_2)$$

[the "not equal" is interesting]

channel types

Really, no time to show it but . . .

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to π-**calculus**. There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(t_1) \vee ch^{+}(t_2) \lneq ch^{+}(t_1 \vee t_2)$$

[the "not equal" is interesting]

channel types

Really, no time to show it but . . .

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to **π-calculus**. There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(t_1) \vee ch^{+}(t_2) \lneq ch^{+}(t_1 \vee t_2)$$

[the "not equal" is interesting]

channel types

Really, no time to show it but ...

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to **π-calculus**. There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(t_1) \vee ch^{+}(t_2) \lneq ch^{+}(t_1 \vee t_2)$$

[the "not equal" is interesting]

channel types

Really, no time to show it but . . .

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to π-**calculus**.
There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(\mathtt{t}_1) \vee ch^{+}(\mathtt{t}_2) \lneqq ch^{+}(\mathtt{t}_1 \vee \mathtt{t}_2)$$

[the "not equal" is interesting]

channel types

Really, no time to show it but ...

This theory applies to other paradigms, too.

For instance in a paper in LICS '05 it is applied to **π-calculus**. There you have nice things such as:

$$ch(t) \stackrel{\text{def}}{=} ch^{-}(t) \wedge ch^{+}(t)$$

[save a constructor]

$$ch^{+}(\mathtt{t}_1) \vee ch^{+}(\mathtt{t}_2) \lneqq ch^{+}(\mathtt{t}_1 \vee \mathtt{t}_2)$$

[the "not equal" is interesting]

Summarizing

$[\![\mathbf{0}]\!] = \emptyset$; $[\![\mathbf{1}]\!] = \mathscr{D}$;
$[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!]$;
$[\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$;
$[\![\neg t]\!] = \mathscr{D} \backslash [\![t]\!]$;
$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$

where the extensional interpretation associated to $[\![\;]\!]$ is defined as:

$$
\begin{array}{lcl}
\mathbb{E}[\![t \rightarrow s]\!] & = & \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \\
\mathbb{E}[\![t \times s]\!] & = & [\![t]\!] \times [\![s]\!] \\
\mathbb{E}[\![\texttt{lazy}\ t]\!] & = & \mathscr{P}([\![t]\!]) \\
\mathbb{E}[\![\texttt{ref}\ t]\!] & = & \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases}
\end{array}
$$

...

Summarizing

$$[\![\mathbf{0}]\!] = \emptyset;\ [\![\mathbf{1}]\!] = \mathscr{D};$$
$$[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!];$$
$$[\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!];$$
$$[\![\neg t]\!] = \mathscr{D} \setminus [\![t]\!];$$
$$[\![t]\!] = \emptyset \iff \mathbb{E}[\![t]\!] = \emptyset$$

where the extensional interpretation associated to $[\![\ ]\!]$ is defined as:

$$\begin{array}{lcl} \mathbb{E}[\![t \rightarrow s]\!] & = & \mathscr{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}}) \\ \mathbb{E}[\![t \times s]\!] & = & [\![t]\!] \times [\![s]\!] \\ \mathbb{E}[\![\mathtt{lazy}\ t]\!] & = & \mathscr{P}([\![t]\!]) \\ \mathbb{E}[\![\mathtt{ref}\ t]\!] & = & \begin{cases} \{[\![t]\!]\} & \text{if } [\![t]\!] \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \end{array}$$

...

Conclusion

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. Find a model (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise tweak it.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence derive a subtyping algorithm.
5. Enjoy.

La morale de l'histoire est ...

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. Find a model (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise tweak it.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence derive a subtyping algorithm.
5. Enjoy.

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. Find a model (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise tweak it.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence derive a subtyping algorithm.
5. Enjoy.

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. Find a model (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise tweak it.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence derive a subtyping algorithm.
5. Enjoy.

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. Find a model (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise tweak it.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence derive a subtyping algorithm.
5. Enjoy.

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. **Find a model** (any model). [may be not easy/possible]
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise **tweak it**. [may be not easy/possible]
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence **derive a subtyping algorithm**. [may be not easy/possible]
5. Enjoy.

La morale de l'histoire est . . .

If you have a strong semantic intuition of your favorite language and you want to add set-theoretic $\vee$, $\wedge$, $\neg$ types then:

1. Define $\mathbb{E}[\![\ ]\!]$ for your type constructors so that it matches your semantic intuition
2. **Find a model** (any model).
3. Use the subtyping relation induced by the model to type your language: if the intuition was right then the set of values is also a model, otherwise **tweak it**.
4. Use the set-theoretic properties of the model (actually of $\mathbb{E}[\![\ ]\!]$) to decompose the emptyness test for your type constructors, and hence **derive a subtyping algorithm**.
5. **Enjoy.**

Addendum 1: a model may not exist

$$t = \texttt{int} \vee (\texttt{ref}(\texttt{int}) \wedge \texttt{ref}(t))$$

Is t equal to $\texttt{int}$?

$$t = \texttt{int} \iff (\texttt{ref}(\texttt{int}) \wedge \texttt{ref}(t)) = \varnothing \iff t \neq \texttt{int}$$

but also

$$t \neq \texttt{int} \iff (\texttt{ref}(\texttt{int}) \wedge \texttt{ref}(t)) \neq \varnothing \iff t = \texttt{int}$$

Addendum 1: a model may not exist

$$t = \mathtt{int} \vee (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t))$$

Is t equal to $\mathtt{int}$?

$$t = \mathtt{int} \iff (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t)) = \varnothing \iff t \neq \mathtt{int}$$

but also

$$t \neq \mathtt{int} \iff (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t)) \neq \varnothing \iff t = \mathtt{int}$$

Addendum 1: a model may not exist

$$t = \mathtt{int} \vee (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t))$$

Is t equal to $\mathtt{int}$?

$$t = \mathtt{int} \iff (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t)) = \varnothing \iff t \neq \mathtt{int}$$

but also

$$t \neq \mathtt{int} \iff (\mathtt{ref}(\mathtt{int}) \wedge \mathtt{ref}(t)) \neq \varnothing \iff t = \mathtt{int}$$

Addendum 2: the real *abstr* typing rule

$$t \equiv (\bigwedge_{i=1..n} s_i \to t_i) \setminus (\bigvee_{j=1..m} s'_j \to t'_j) \not\leq \mathbf{0}$$

$$\frac{(\forall i)\ \Gamma, (f : t), (x : s_i) \vdash e : t_i}{\Gamma \vdash \mu f^{(s_1 \to t_1; \ldots; s_n \to t_n)}(x).e : t}\ (abstr)$$

Addendum 3: A different definition for $\mathbb{E}[\![\ ]\!]$

Note that according to the previous $\mathbb{E}[\![\ ]\!]$:

$$s \rightarrow t \ \leq \ \mathbf{1} \rightarrow \mathbf{1} \tag{3}$$

Every application is well typed. Add a distinguished Ω to denote a runtime type error, modify

$$\mathbb{E}[\![t{\longrightarrow}s]\!] = \{f \subseteq \mathscr{D}{\times}(\mathscr{D}{\cup}\{\Omega\}) \mid \forall(d_1, d_2) \in f.\, d_1 \in [\![t]\!] \Rightarrow d_2 \in [\![s]\!]\}$$

(3) no longer holds since the constant map $\neg s \mapsto \Omega$, is in the left hand type but not in the right one.

$$\bigwedge_{i\in I}(t_i{\longrightarrow}s_i) \ \leq \bigvee_{j\in J}(t'_j{\longrightarrow}s'_j) \iff$$
$$\exists j \in J.\begin{cases} t'_j \leq \bigvee_{i\in I} t_i \ \wedge \\ \forall I' \subseteq I.\ (t'_j \leq \bigvee_{i\in I'} t_i) \vee (\bigwedge_{i\in I\setminus I'} s_i \leq s'_j) \end{cases}$$

Addendum 3: A different definition for $\mathbb{E}[\![\ ]\!]$

Note that according to the previous $\mathbb{E}[\![\ ]\!]$:

$$s \to t \ \leq\ \mathbf{1} \to \mathbf{1} \qquad (3)$$

Every application is well typed. Add a distinguished Ω to denote a runtime type error, modify

$$\mathbb{E}[\![t{\longrightarrow}s]\!] = \{f \subseteq \mathscr{D}\times(\mathscr{D}\cup\{\Omega\}) \mid \forall(d_1,d_2)\in f.\, d_1\in[\![t]\!] \Rightarrow d_2\in[\![s]\!]\,\}$$

(3) no longer holds since the constant map $\neg s \mapsto \Omega$, is in the left hand type but not in the right one.

$$\bigwedge_{i\in I}(t_i{\to}s_i)\ \leq \bigvee_{j\in J}(t'_j{\to}s'_j)\ \iff$$

$$\exists j\in J.\begin{cases} t'_j \leq \bigvee_{i\in I} t_i\ \wedge \\ \forall I'\subseteq I.\ (t'_j \leq \bigvee_{i\in I'} t_i) \vee (\bigwedge_{i\in I\setminus I'} s_i \leq s'_j) \end{cases}$$

Addendum 3: A different definition for $\mathbb{E}[\![\ ]\!]$

Note that according to the previous $\mathbb{E}[\![\ ]\!]$:

$$s \to t \leq \mathbf{1} \to \mathbf{1} \tag{3}$$

Every application is well typed. Add a distinguished Ω to denote a runtime type error, modify

$$\mathbb{E}[\![t{\longrightarrow}s]\!] = \{f \subseteq \mathscr{D}\times(\mathscr{D}\cup\{\Omega\}) \mid \forall(d_1,d_2)\in f.\, d_1 \in [\![t]\!] \Rightarrow d_2 \in [\![s]\!]\,\}$$

(3) no longer holds since the constant map $\neg s \mapsto \Omega$, is in the left hand type but not in the right one.

$$\bigwedge_{i\in I}(t_i{\to}s_i) \leq \bigvee_{j\in J}(t'_j{\to}s'_j) \iff \exists j\in J. \begin{cases} t'_j \leq \bigvee_{i\in I} t_i \wedge \\ \forall I'\subseteq I.\ (t'_j \leq \bigvee_{i\in I'} t_i) \vee (\bigwedge_{i\in I\setminus I'} s_i \leq s'_j) \end{cases}$$

Addendum 3: A different definition for $\mathbb{E}[\![\ ]\!]$

Note that according to the previous $\mathbb{E}[\![\ ]\!]$:

$$s \to t \;\leq\; \mathbf{1} \to \mathbf{1} \tag{3}$$

Every application is well typed. Add a distinguished Ω to denote a runtime type error, modify

$$\mathbb{E}[\![t \!\to\! s]\!] = \{ f \subseteq \mathscr{D} \times (\mathscr{D} \cup \{\Omega\}) \mid \forall (d_1, d_2) \in f.\, d_1 \in [\![t]\!] \Rightarrow d_2 \in [\![s]\!] \}$$

(3) no longer holds since the constant map $\neg s \mapsto \Omega$, is in the left hand type but not in the right one.

$$\bigwedge_{i\in I}(t_i \!\to\! s_i) \;\leq \bigvee_{j\in J}(t'_j \!\to\! s'_j) \iff \exists j \in J. \begin{cases} t'_j \leq \bigvee_{i\in I} t_i \;\wedge \\ \forall I' \subseteq I.\; (t'_j \leq \bigvee_{i\in I'} t_i) \vee (\bigwedge_{i\in I\setminus I'} s_i \leq s'_j) \end{cases}$$