ON DIFFERENTIAL PROGRAM SEMANTICS

Ugo Dal Lago
Department of Computer Science and Engineering
University of Bologna
PROGRAM SEMANTICS
$A^* \in \tau$

$C^* : \tau \rightarrow \rho$
$A^* \in \tau$

$C^*: \tau \to \rho$

$C^*(A^*)$
A Recipe

1. Given a programming language, define a type system.

\[
\tau ::= \text{NAT} | \tau \rightarrow \rho
\]

\[\vdash M : \tau\]

2. Define a family of relations between programs, indexed by types, and following their structure.

\[M \equiv R \tau \rightarrow \rho N \iff (ML) \equiv R \rho ((NP)) \text{ whenever } L \equiv R \tau P.\]

3. Prove the Fundamental Theorem

\[\vdash M : \tau \Rightarrow M \equiv R M\]

Works extremely well for typed programming languages.

Along the years, adapted to language with non-inductive type structures, concurrent programming languages, etc.

Often proved fully abstract with respect to contextual equivalence:

\[\equiv \iff \forall C. \text{Obs}(C[M]) = \text{Obs}(C[N]).\]

Due to Plotkin, but later studied by many others.
A Recipe

1. Given a programming language, define a **type system**.

\[ \tau ::= \text{NAT} \mid \tau \to \rho \]

Along the years, adapted to language with non-inductive type structures, concurrent programming languages, etc.

Often proved fully abstract with respect to contextual equivalence:

\[ M \equiv N \iff \forall C. \text{Obs}(C[M]) = \text{Obs}(C[N]) \]

Due to Plotkin, but later studied by many others.
A Recipe

1. Given a programming language, define a type system.

\[ \tau ::= \text{NAT} \mid \tau \to \rho \quad \vdash M : \tau \]
A Recipe

1. Given a programming language, define a type system.
   \[ \tau ::= \text{NAT} \mid \tau \to \rho \quad \vdash M : \tau \]

2. Define a family of relations between programs, indexed by types, and following their structure.
   \[ M \mathcal{R}_{\tau \to \rho} N \quad \text{iff} \quad (ML) \mathcal{R}_\rho (NP) \quad \text{whenever} \quad L \mathcal{R}_\tau P. \]
A Recipe

1. Given a programming language, define a type system.

\[ \tau ::= \text{NAT} \mid \tau \to \rho \quad \vdash M : \tau \]

2. Define a family of relations between programs, indexed by types, and following their structure.

\[ M \mathcal{R}_{\tau \to \rho} N \quad \text{iff} \quad (ML) \mathcal{R}_\rho (NP) \quad \text{whenever} \quad L \mathcal{R}_\tau P. \]

3. Prove the Fundamental Theorem

\[ \vdash M : \tau \implies M \mathcal{R}_\tau M \]
A Recipe

1. Given a programming language, define a type system.
   \[ \tau ::= \text{NAT} | \tau \rightarrow \rho \]
   \[ \vdash M : \tau \]
   
   Works extremely well for typed programming languages.
   Along the years, adapted to language with non-inductive type structures, concurrent programming languages, etc.

2. Define a family of relations between programs, indexed by types, and following their structure.
   \[ M \mathcal{R}_\tau \rho \iff (\text{ML}) \mathcal{R}_\rho (\text{NP}) \text{ whenever } L \mathcal{R}_\tau P. \]
   
   Often proved fully abstract with respect to contextual equivalence:
   \[ M \equiv N \iff \forall C. \text{Obs}(C[M]) = \text{Obs}(C[N]). \]

3. Prove the Fundamental Theorem.
   \[ \vdash M : \tau \implies M \mathcal{R}_\tau M \]
   
   Due to Plotkin, but later studied by many others.
A Recipe

1. See program (configurations) as states of a **labelled transition system**.

![Diagram](image_url)

- **input**($x$)
- $y \leftarrow x \times x$
- **output**($y$)

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
</tr>
</tbody>
</table>

$\Rightarrow i(3)$

- **output**($y$)

<table>
<thead>
<tr>
<th>$x$</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
</tr>
</tbody>
</table>

$\Rightarrow o(9)$

<table>
<thead>
<tr>
<th>$x$</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>9</td>
</tr>
</tbody>
</table>
A Recipe

1. See program (configurations) as states of a labelled transition system.

2. Define a notion of bisimulation relation.

\[(M \mathcal{R} N) \land (M \xrightarrow{\ell} L) \implies \exists P. (N \xrightarrow{\ell} P) \land (L \mathcal{R} P),\]

\[(N \mathcal{R} M) \land (N \xrightarrow{\ell} P) \implies \exists L. (M \xrightarrow{\ell} L) \land (L \mathcal{R} P).\]

Bisimilarity, indicated as \(\sim\), is the largest bisimulation relation.

- Very successful in the absence of types, and in a concurrent scenario.
- Becomes extremely useful in practice when coupled with so-called up-to techniques.
- Incepted into program semantics by Milner.
A Recipe

1. See program (configurations) as states of a labelled transition system.

2. Define a notion of bisimulation relation.

\[(M \mathcal{R} N) \land (M \xrightarrow{\ell} L) \implies \exists P. (N \xrightarrow{\ell} P) \land (L \mathcal{R} P),\]

\[(N \mathcal{R} M) \land (N \xrightarrow{\ell} P) \implies \exists L. (M \xrightarrow{\ell} L) \land (L \mathcal{R} P).\]

Bisimilarity, indicated as \(\sim\), is the largest bisimulation relation.

3. Prove that bisimilarity is a congruence.

\[M \sim N \implies C[M] \sim C[N].\]
A Recipe

1. See program (configurations) as states of a **labelled transition system**.

2. Define a notion of **bisimulation relation**.

3. Prove that **bisimilarity** is a congruence.

- Very successful in the absence of types, and in a concurrent scenario.
- Becomes extremely useful in practice when coupled with so-called up-to techniques.
- Incepted into program semantics by Milner.

\[(N \mathcal{R} M) \land (N \xrightarrow{\ell} P) \implies \exists L. (M \xrightarrow{\ell} L) \land (L \mathcal{R} P)\].

Bisimilarity, indicated as \(\sim\), is the largest bisimulation relation.

4. Prove that **bisimilarity is a congruence**.

\[M \sim N \implies C[M] \sim C[N].\]
A Recipe
A Recipe

1. Types are interpreted as games, namely sets of sequences of moves.

\[ GAME(\text{NAT}) = \{ \varepsilon, q, q \cdot 0, q \cdot 1, q \cdot 2, \cdots \} \]

\[ GAME(\text{NAT} \rightarrow \text{BOOL}) = \{ \varepsilon, q_{\text{BOOL}}, q_{\text{BOOL}} \cdot \text{True}, q_{\text{BOOL}} \cdot \text{False}, q_{\text{BOOL}} \cdot q_{\text{NAT}}, \cdots \} \]
A Recipe

1. Types are interpreted as games, namely sets of sequences of moves.

\[ \text{GAME}(\text{NAT}) = \{ \varepsilon, q, q \cdot 0, q \cdot 1, q \cdot 2, \cdots \} \]

\[ \text{GAME}(\text{NAT} \rightarrow \text{BOOL}) = \{ \varepsilon, q_{\text{BOOL}}, q_{\text{BOOL}} \cdot \text{True}, q_{\text{BOOL}} \cdot \text{False}, q_{\text{BOOL}} \cdot q_{\text{NAT}}, \cdots \} \]

2. Programs are interpreted as strategies, namely functions which returns the next move from the history.

\[ f = \text{STRATEGY}(\lambda x. \text{if } (x > 0) \text{ then True else False}) \]

\[ f(q_{\text{BOOL}}) = q_{\text{NAT}}; \quad f(q_{\text{BOOL}} \cdot q_{\text{NAT}} \cdot 3) = \text{True} \]
A Recipe

1. Types are interpreted as games, namely sets of sequences of moves.

   \[
   \text{GAME}(\text{NAT}) = \{\varepsilon, q, q \cdot 0, q \cdot 1, q \cdot 2, \cdots \}
   \]

   \[
   \text{GAME}(\text{NAT} \rightarrow \text{BOOL}) = \{\varepsilon, q_{\text{BOOL}}, q_{\text{BOOL}} \cdot \text{True}, q_{\text{BOOL}} \cdot \text{False}, q_{\text{BOOL}} \cdot q_{\text{NAT}}, \cdots \}
   \]

2. Programs are interpreted as strategies, namely functions which returns the next move from the history.

   \[
   f = \text{STRATEGY}(\lambda x. \text{if } (x > 0) \text{ then True else False})
   \]

   \[
   f(q_{\text{BOOL}}) = q_{\text{NAT}}; \quad f(q_{\text{BOOL}} \cdot q_{\text{NAT}} \cdot 3) = \text{True}
   \]

3. Computation is seen as the interaction between a program, modeled as a strategy, and its environment, itself seen as a (possibly nondeterministic) strategy, called the observer.
A Recipe

1. Types are interpreted as **games**, namely sets of sequences of **moves**.
   - The equivalence induced by interpreting programs as strategies has been proved to *coincide* with contextual equivalence in many cases.
   - It has been proved to be adaptable to many different kinds of programming languages, and to concurrent languages in particular.
   - There is a symmetry between the program and the environment, which can be seen itself as a strategy.

2. Programs are interpreted as **strategies**, namely functions which return the next move from the history.
   - \[ \text{GAME}(\text{NAT}) = \{ \varepsilon, q_0, q_1, q_2, \ldots \} \]
   - \[ \text{GAME}(\text{NAT} \rightarrow \text{BOOL}) = \{ \varepsilon, q_{\text{BOOL}}, q_{\text{BOOL} \cdot \text{True}}, q_{\text{BOOL} \cdot \text{False}}, q_{\text{NAT}}, \ldots \} \]

3. Computation is seen as the interaction between a program, modeled as a strategy, and its environment, itself seen as a (possibly nondeterministic) strategy, called the **observer**.
A Story of Successes...
A Story of Successes... 
...is This The End?
A Story of Successes . . .

. . . is This The End?

Definitely No!
Can we turn logical relations into a metric?

\[ R_\tau \subseteq \Lambda_\tau \times \Lambda_\tau \quad \iff \quad \Delta_\tau : \Lambda_\tau \times \Lambda_\tau \rightarrow \mathbb{R}^\infty_{\geq 0} \]
Can we turn logical relations into a metric?

\[ \mathcal{R}_\tau \subseteq \Lambda_\tau \times \Lambda_\tau \quad \Longmapsto \quad \Delta_\tau : \Lambda_\tau \times \Lambda_\tau \rightarrow \mathbb{R}^\infty_{\geq 0} \]

We can answer this question affirmatively [RP2010]! Define a family of metrics, indexed by types, as follows:

\[ \Delta_{\tau \rightarrow \rho}(M, N) \leq k \quad \text{iff} \quad \Delta_\rho(ML, NP) \leq k + \Delta_\tau(L, P) \] whenever \( L, P : \tau \)

The Fundamental Theorem can be extended naturally: \( \Delta_\tau(M, M) = 0 \) whenever \( M : \tau \).

This only works in an affine setting, so not in the case of the ST\( \lambda \)C.

Functions on the reals we start from must be Lipschitz.

Categorically, this corresponds to the fact that metric spaces with Lipschitz maps form a SMCC, but not a CCC.
Can we turn logical relations into a metric?

\[ \mathcal{R}_\tau \subseteq \Lambda_\tau \times \Lambda_\tau \quad \mapsto \quad \Delta_\tau : \Lambda_\tau \times \Lambda_\tau \to \mathbb{R}_{\geq 0} \]

We can answer this question affirmatively [RP2010]! Define a family of metrics, indexed by types, as follows:

\[ \Delta_{\tau\rightarrow \rho}(M, N) \leq k \quad \text{iff} \quad \Delta_\rho(ML, NP) \leq k + \Delta_\tau(L, P) \quad \text{whenever} \quad L, P : \tau \]

The Fundamental Theorem can be extended naturally:

\[ \Delta_\tau(M, M) = 0 \quad \text{whenever} \quad M : \tau \]
Can we turn logical relations into a metric?

\[ \tau \subseteq \Lambda \tau \times \Lambda \tau \rightarrow \mathbb{R}^{\infty}_{\geq 0} \]

We can answer this question affirmatively [RP2010]! Define a family of metrics, indexed by types, as follows:

\[ \Delta_{\tau} \rightarrow \rho(M, N) \leq k \iff \Delta_{\rho}(ML, NP) \leq k + \Delta_{\tau}(L, P) \]

whenever \( L, P : \tau \)

The Fundamental Theorem can be extended naturally:

\[ \Delta_{\tau}(M, M) = 0 \] whenever \( M : \tau \)
def A(x):
    return sin(x)

def B(x):
    return x
def A(x):
    return sin(x)

def B(x):
    return x

Not Equivalent At Infinite Distance
But Very Similar if x is Close to 0

Approximate Program Transformation
def A(x):
    return sin(x)

def B(x):
    return x

Not Equivalent

At Infinite Distance
But Very Similar
if \(x\) is Close to 0

Approximate Program Transformation
def A(x):
    return sin(x)
def B(x):
    return x

Not Equivalent At Infinite Distance
\texttt{def A(x):}  
\hspace{1em} \texttt{return sin(x)}

def B(x):
\hspace{1em} \texttt{return x}

\textbf{Not Equivalent}  
\textbf{At Infinite Distance}  
\textbf{But Very Similar if } x \textbf{ is Close to 0}
def A(x):
    return sin(x)
def B(x):
    return x

Not Equivalent
At Infinite Distance
But Very Similar if x is Close to 0
def A(x):
    return 1/sqrt(x)

def B(x):
    return 2.08-1.119*x
def A(x):
    return 1/sqrt(x)

def B(x):
    return 2.08-1.119*x
FURTHER EXAMPLES

Loop perforation

```python
def sum_to_n(n):
    sum = 0
    for (i=0, i<n, i++)
        sum += i
    return sum
```

Fast inverse square root

```c
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threethirds = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = *( long *) &y;
    i = 0x5f3759df - ( i >> 1 );
    y = *( float *) &i;
    y = y * ( threethirds - ( x2 * y * y ) );
    return y;
}
```
The **Difference** is the Meaning.
The **Difference** is the Meaning.
The Context **Matters**.
The **Difference** is the Meaning. The Context **Matters**.
The **Difference** is the Meaning. The Context **Matters**.
```
def A(x):
    return sin(x)
```

```
def B(x):
    return x
```

\[ \varepsilon \approx 0 \]
\[ \delta(y) = y - \sin y \]
\[ \delta \circ \varepsilon = \delta(\varepsilon) \approx 0 \]

**Not Equivalent**

At **Infinite** Distance

But **Very Similar** if \( x \) is Close to 0
FUNDAMENTAL TENSION

Effective

Not Expressive

Expressive

Not Effective

Relations

Program Pair

Expressiveness

Relations

Program Pair

Expressiveness

13
$\delta \in \mathbb{R}^+$
$\varepsilon \in \mathbb{R}^+$

Effective
Not Expressive
Effective
Not Expressive

}\delta \in \mathbb{R}^+
\varepsilon \in \mathbb{R}^+

Expressive
Not Effective

}\delta = (A, B)
\varepsilon = C

FUNDAMENTAL TENSION

Program Pair
Expressiveness

Program Pair

Relations

Relations

13
\[ \delta \in \mathbb{R}^+ \quad \varepsilon \in \mathbb{R}^+ \]

\[ \delta = (A, B) \quad \varepsilon = C \]

Effective Not Expressive

Expressive Not Effective

Expressiveness

Effectiveness

Relations

Metrics

Program Pair Distances
FUNDAMENTAL TENSION

Effective
Not Expressive

\[ \delta \in \mathbb{R}^+ \]
\[ \varepsilon \in \mathbb{R}^+ \]

Expressive
Not Effective

\[ \delta = (A, B) \]
\[ \varepsilon = C \]

Expressiveness
Effectiveness

Relations
Metrics
DIAPASoN
Program Pair Distances
Differential Logical Relations
Types

\[ \tau, \rho ::= REAL \mid \tau \rightarrow \rho \mid \tau \times \rho \]
Types
\[ \tau, \rho ::= \text{REAL} \mid \tau \to \rho \mid \tau \times \rho \]

Typing Rules
\[
\begin{align*}
\Gamma &\vdash x : \tau \quad & \text{x : \tau \in \Gamma} \\
\Gamma &\vdash r : \text{REAL} \quad & \text{\Gamma \vdash r : \text{REAL}} \\
\Gamma &\vdash f_n : \text{REAL}^n \to \text{REAL} \quad & \text{\Gamma \vdash f_n : \text{REAL}^n \to \text{REAL}} \\
\Gamma, x : \tau &\vdash M : \rho \quad & \text{\Gamma, x : \tau \vdash M : \rho} \\
\Gamma &\vdash M : \tau \to \rho \quad \Gamma &\vdash N : \tau \quad & \text{\Gamma \vdash M : \tau \to \rho \quad \Gamma \vdash N : \tau} \\
\Gamma \vdash \langle M, N \rangle : \tau \times \rho \quad & \text{\Gamma \vdash \langle M, N \rangle : \tau \times \rho} \\
\Gamma &\vdash \pi_1 : \tau \times \rho \to \tau \quad & \text{\Gamma \vdash \pi_1 : \tau \times \rho \to \tau} \\
\Gamma &\vdash \pi_2 : \tau \times \rho \to \rho \quad & \text{\Gamma \vdash \pi_2 : \tau \times \rho \to \rho} \\
\Gamma &\vdash M : \tau \quad \Gamma &\vdash N : \tau \quad & \text{\Gamma \vdash M : \tau} \quad \Gamma \vdash N : \tau \\
\Gamma &\vdash \text{if} \text{lz } M \text{ else } N : \text{REAL} \to \tau \quad & \text{\Gamma \vdash \text{if} \text{lz } M \text{ else } N : \text{REAL} \to \tau} \\
\Gamma &\vdash \text{iter } M \text{ base } N : \text{REAL} \to \tau \quad & \text{\Gamma \vdash \text{iter } M \text{ base } N : \text{REAL} \to \tau}
\end{align*}
\]
Types

$$\tau, \rho ::= \text{REAL} \mid \tau \to \rho \mid \tau \times \rho$$

Typing Rules

\[
\begin{align*}
\Gamma \vdash x : \tau & \quad \quad & \Gamma \vdash r : \text{REAL} & \quad \quad & \Gamma \vdash f_n : \text{REAL}^n \to \text{REAL} & \quad \quad & \Gamma, x : \tau \vdash M : \rho \\
\Gamma \vdash M : \tau \to \rho & \quad \quad & \Gamma \vdash N : \tau & \quad \quad & \Gamma \vdash M : \tau & \quad \quad & \Gamma \vdash N : \rho & \quad \quad & \Gamma \vdash MN : \rho \\
\Gamma \vdash M : \tau & \quad \quad & \Gamma \vdash N : \tau & \quad \quad & \Gamma \vdash \langle M, N \rangle : \tau \times \rho & \quad \quad & \Gamma \vdash \pi_1 : \tau \times \rho \to \tau & \quad \quad & \Gamma \vdash \pi_2 : \tau \times \rho \to \rho \\
\Gamma \vdash if lz \ M \ \text{else} \ N : \text{REAL} \to \tau & \quad \quad & \Gamma \vdash \text{iter} M \ \text{base} N : \text{REAL} \to \tau
\end{align*}
\]

Denotational Semantics

\[
[\text{REAL}] = \mathbb{R}; \quad [\tau \to \rho] = [\tau] \to [\rho]; \quad [\tau \times \rho] = [\tau] \times [\rho].
\]
Distance Spaces

\[(\text{REAL}) = \mathbb{R}_0^{\infty};\]
\[(\tau \rightarrow \rho) = [\tau] \times (\tau) \rightarrow (\rho);\]
\[(\tau \times \rho) = (\tau) \times (\rho);\]
Distance Spaces

\[(\text{REAL}) = \mathbb{R}_{\geq 0}^\infty;\]  
\[(\tau \rightarrow \rho) = \llbracket\tau\rrbracket \times (\tau) \rightarrow (\rho);\]  
\[(\tau \times \rho) = (\tau) \times (\rho);\]

DLRs as Ternary Relations

\[\delta_{\text{REAL}}(M, r, N) \Leftrightarrow |NF(M) - NF(N)| \leq r;\]
\[\delta_{\tau \times \rho}(M, (d_1, d_2), N) \Leftrightarrow \delta_\tau(\pi_1 M, d_1, \pi_1 N) \land \delta_\rho(\pi_2 M, d_2, \pi_2 N)\]
\[\delta_{\tau \rightarrow \rho}(M, d, N) \Leftrightarrow (\forall V \in CV(\tau). \forall x \in (\llbracket\tau\rrbracket). \forall W \in CV(\tau).\]
\[\delta_\tau(V, x, W) \Rightarrow \delta_\rho(MV, d(\llbracket V \rrbracket, x), NW)\]
\[\land \delta_\rho(MW, d(\llbracket V \rrbracket, x), NV)).\]
Distance Spaces

\(|REAL| = \mathbb{R}_{\geq 0}; (\tau \rightarrow \rho) = [\tau] \times (\tau) \rightarrow (\rho); (\tau \times \rho) = (\tau) \times (\rho);\) \\

**DLRs as Ternary Relations**

\begin{align*}
\delta_{REAL}(M, r, N) &\Leftrightarrow |NF(M) - NF(N)| \leq r; \\
\delta_{\tau \times \rho}(M, (d_1, d_2), N) &\Leftrightarrow \delta_{\tau}(\pi_1 M, d_1, \pi_1 N) \land \delta_{\rho}(\pi_2 M, d_2, \pi_2 N) \\
\delta_{\tau \rightarrow \rho}(M, d, N) &\Leftrightarrow (\forall V \in CV(\tau). \forall x \in (\tau). \forall W \in CV(\tau).
\delta_{\tau}(V, x, W) \Rightarrow \delta_{\rho}(MV, d([V], x), NW)
\land \delta_{\rho}(MW, d([V], x), NV)).
\end{align*}

**Theorem (Fundamental Lemma, Version I)**

*For every \(\vdash M : \tau\), there is \(d \in (|\tau|)\) such that \(\delta_{\tau}(M, d, M)\)*.
Claim

\[ \delta_{\text{REAL} \rightarrow \text{REAL}}(M_{ID}, \lambda \langle x, y \rangle. y + |x - \sin x|, M_{SIN}) \]
Claim

\[ \delta_{\text{REAL} \to \text{REAL}}(M_{\text{ID}}, \lambda \langle x, y \rangle. y + |x - \sin x|, M_{\text{SIN}}) \]

Proof.
Consider any pairs of real numbers \( r, s \in \mathbb{R} \) such that \( |r - s| \leq \varepsilon \), where \( \varepsilon \in \mathbb{R}_{\geq 0} \). We have that:

\[
|\sin r - s| = |\sin r - r + r - s| \leq |\sin r - r| + |r - s| \\
\leq |\sin r - r| + \varepsilon = f(r, \varepsilon) \\
|\sin s - r| = |\sin s - \sin r + \sin r - r| \\
\leq |\sin s - \sin r| + |\sin r - r| \leq |s - r| + |\sin r - r| \\
\leq \varepsilon + |\sin r - r| = f(r, \varepsilon).
\]
$M_{ID} = \lambda x. x$

The function $\Delta \rho$ is indeed a (pseudo)-metric. The function $\lambda \lambda \langle x, y \rangle . y$ is the smallest self distance for $M_{ID}$. (The function induced by) $\delta \rho$, indeed, does not satisfy the reflexivity axiom, and as a consequence does not have the structure of a metric. It can however be given the structure of a generalized metric domain.
A METRIC?

\[ M_{ID} = \lambda x.x \]

\[ \Delta_{\tau \rightarrow \tau}(M_{ID}, M_{ID}) = 0 \]
$M_{ID} = \lambda x. x$

$\Delta_{\tau \to \tau}(M_{ID}, M_{ID}) = 0$

$\delta_{\tau \to \tau}(M_{ID}, \lambda \langle x, y \rangle. y, M_{ID})$
The function $\Delta_\rho$ is indeed a (pseudo)-metric.

$M_{ID} = \lambda x.x$

$\Delta_{\tau \rightarrow \tau}(M_{ID}, M_{ID}) = 0$

$\delta_{\tau \rightarrow \tau}(M_{ID}, \lambda \langle x, y \rangle.y, M_{ID})$
$M_{ID} = \lambda x.x$

$\Delta_{\tau \rightarrow \tau}(M_{ID}, M_{ID}) = 0$

$\delta_{\tau \rightarrow \tau}(M_{ID}, \lambda \langle x, y \rangle.y, M_{ID})$

- The function $\Delta_{\rho}$ is indeed a (pseudo)-metric.
- The function $\lambda \langle x, y \rangle.y$ is the smallest self distance for $M_{ID}$. 
The function $\Delta_\rho$ is indeed a (pseudo)-metric.

The function $\lambda \langle x, y \rangle . y$ is the smallest self distance for $M_{ID}$.

(The function induced by) $\delta_\rho$, indeed, does not satisfy the reflexivity axiom, and as a consequence does not have the structure of a metric.
A METRIC?

\[ M_{ID} = \lambda x.x \]

\[ \Delta_{\tau \to \tau}(M_{ID}, M_{ID}) = 0 \]

\[ \delta_{\tau \to \tau}(M_{ID}, \lambda\langle x, y \rangle.y, M_{ID}) \]

- The function \( \Delta_\rho \) is indeed a (pseudo)-metric.
- The function \( \lambda\langle x, y \rangle.y \) is the smallest self distance for \( M_{ID} \).
- (The function induced by) \( \delta_\rho \), indeed, does not satisfy the reflexivity axiom, and as a consequence does not have the structure of a metric.
- It can however be given the structure of a generalized metric domain.
Hereditarily Null Distances

\[
(\text{REAL})^0 = \{0\} \quad (\tau \times \rho)^0 = (\tau)^0 \times (\rho)^0 \\
(\tau \rightarrow \rho)^0 = \{f \mid \forall x \in \tau. \forall y \in (\tau)^0. f(x, y) \in (\rho)^0\}
\]
Hereditarily Null Distances

\[ (\text{REAL})^0 = \{0\} \quad (\tau \times \rho)^0 = (\tau)^0 \times (\rho)^0 \]
\[ (\tau \rightarrow \rho)^0 = \{ f \mid \forall x \in [\tau].\forall y \in (\tau)^0 . f(x, y) \in (\rho)^0 \} \]

Hereditarily Finite Distances

\[ (\text{REAL})^{<\infty} = \mathbb{R}_{\geq 0}; \quad (\tau \times \rho)^{<\infty} = (\tau)^{<\infty} \times (\rho)^{<\infty} \]
\[ (\tau \rightarrow \rho)^{<\infty} = \{ f \in (\tau \rightarrow \rho) \mid \forall x \in [\tau].\forall t \in (\tau)^{<\infty} . f(x, t) \in (\rho)^{<\infty} \}. \]
Hereditarily Null Distances

\[
\langle \text{REAL} \rangle^0 = \{0\} \quad (\tau \times \rho)^0 = \langle \tau \rangle^0 \times \langle \rho \rangle^0
\]

\[
(\tau \to \rho)^0 = \{ f \mid \forall x \in [\tau]. \forall y \in \langle \tau \rangle^0 . f(x, y) \in \langle \rho \rangle^0 \}
\]

Hereditarily Finite Distances

\[
\langle \text{REAL} \rangle^{<\infty} = \mathbb{R}_{\geq 0}; \quad (\tau \times \rho)^{<\infty} = \langle \tau \rangle^{<\infty} \times \langle \rho \rangle^{<\infty};
\]

\[
(\tau \to \rho)^{<\infty} = \{ f \in (\tau \to \rho) \mid \forall x \in [\tau]. \forall t \in \langle \tau \rangle^{<\infty} . f(x, t) \in \langle \rho \rangle^{<\infty} \}.
\]

Lemma

Whenever \( \vdash M, N : \tau \), \( M \) is logically related to \( N \) iff \( \delta_\tau(M, d, N) \) where \( d \in (\langle \tau \rangle^0) \).
Hereditarily Null Distances

\langle \text{REAL} \rangle^0 = \{0\} \quad \langle \tau \times \rho \rangle^0 = \langle \tau \rangle^0 \times \langle \rho \rangle^0

\langle \tau \rightarrow \rho \rangle^0 = \{ f \mid \forall x \in \llbracket \tau \rrbracket. \forall y \in \langle \tau \rangle^0. f(x, y) \in \langle \rho \rangle^0 \}\}

Hereditarily Finite Distances

\langle \text{REAL} \rangle^{<\infty} = \mathbb{R}_{\geq 0}; \quad \langle \tau \times \rho \rangle^{<\infty} = \langle \tau \rangle^{<\infty} \times \langle \rho \rangle^{<\infty};

\langle \tau \rightarrow \rho \rangle^{<\infty} = \{ f \in \langle \tau \rightarrow \rho \rangle \mid \forall x \in \llbracket \tau \rrbracket. \forall t \in \langle \tau \rangle^{<\infty}. f(x, t) \in \langle \rho \rangle^{<\infty} \}\).

Lemma

Whenever \( \vdash M, N : \tau \), \( M \) is logically related to \( N \) iff \( \delta_\tau(M, d, N) \) where \( d \in \langle \mid \tau \mid \rangle^0 \).

Theorem (Fundamental Lemma, Version II)

For every \( \vdash M : \tau \), there is \( d \in \langle \mid \tau \mid \rangle^{<\infty} \) such that \( \delta_\tau(M, d, M) \).
def sum_to_n(n):
    sum = 0
    for (i=0, i<n, i++)
        sum += i
    return sum
def sum_to_n(n):
    sum = 0
    for (i=0, i<n, i+=k)
        sum += i
    return sum
Loops, iteration, ...

Adding Recursion

\[
\Gamma, f : \tau \rightarrow \rho, x : \tau \vdash M : \rho \\
\Gamma \vdash \text{fix}(f, x).M : \tau \rightarrow \rho
\]
Step-indexed DLRs

\[ \delta_{\text{REAL}}(n, V, r, W) \iff |V - W| \leq r; \]

\[ \vdots \]

\[ \delta_{\tau \rightarrow \rho}(n, V, d, W) \iff (\forall k < n, \forall U \in CV(\tau). \forall x \in |\tau|. \forall Z \in CV(\tau). \]

\[ \delta_{\tau}(k, U, x, Z) \Rightarrow \delta_{\rho}(k, VU, d([U], x), WZ) \]

\[ \land \delta_{\rho}(k, VZ, d([U], x), WU)); \]

\[ \delta_{\tau}(n, M, d, N) \iff \forall k < n. \ (M \downarrow_k V \land N \downarrow W \Rightarrow \delta_{\tau}(n - k, V, d, W) \land \]

\[ N \downarrow_k W \land M \downarrow V \Rightarrow \delta_{\tau}(n - k, V, d, W)). \]
Step-indexed DLRs

\[ \delta_{REAL}(n, V, r, W) \iff |V - W| \leq r; \]

\[ \vdots \]

\[ \delta_{\tau \rightarrow \rho}(n, V, d, W) \iff (\forall k < n, \forall U \in CV(\tau). \forall x \in \langle \tau \rangle. \forall Z \in CV(\tau).
\delta_{\tau}(k, U, x, Z) \Rightarrow \delta_{\rho}(k, VU, d([U], x), WZ) \wedge \delta_{\rho}(k, VZ, d([U], x), WU)); \]

\[ \delta_{\tau}(n, M, d, N) \iff \forall k < n. (M \Downarrow_k V \wedge N \Downarrow W \Rightarrow \delta_{\tau}(n - k, V, d, W) \wedge
N \Downarrow_k W \wedge M \Downarrow V \Rightarrow \delta_{\tau}(n - k, V, d, W)). \]

Theorem (Fundamental Lemma, Version III)

For every \( \vdash M : \tau \) and \( n \geq 0 \), there is \( d \in (|\tau|) \) such that \( \delta_{\tau}(n, M, d, M) \).
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalves = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y;
    i = 0x5f3759df - ( i >> 1 );
    y = * ( float * ) &i;
    y = y * ( threehalves - ( x2 * y * y ) );
    return y;
}

- Imperative features
- Errors
- Randomness
- IO
Adding Effects

\[ \tau ::= \cdots \mid T(\tau) \]
Effectful Distance Spaces

\((REAL) = \mathbb{R}_{\geq 0};\quad (\tau \rightarrow \rho) = [\tau] \times [\tau] \rightarrow [\rho];\quad (\tau \times \rho) = [\tau] \times [\rho];\quad (T(\tau)) = U(\tau)\)
Effectful Distance Spaces

\[(\text{REAL}) = \mathbb{R}_{\geq 0}^\infty; \quad (\tau \to \rho) = [[\tau]] \times (\tau) \to (\rho); \quad (\tau \times \rho) = (\tau) \times (\rho); \quad (T(\tau)) = U(|\tau|)\]

- Randomness.  \(T = U = D\) (distribution monad)
- Output.
  - \(T = \Sigma^* \times \ldots\) (output monad);
  - \(U = \mathbb{N} \times \ldots\) (difference between output strings).
Effectful Distance Spaces

\( (REAL) = \mathbb{R}_{\geq 0}^\infty; \quad (\tau \to \rho) = [\tau] \times [\tau] \to (\rho); \quad (\tau \times \rho) = (\tau) \times (\rho); \quad (T(\tau)) = U(\tau) \)

- Randomness. \( T = U = D \) (distribution monad)
- Output.
  - \( T = \Sigma^* \times - \) (output monad);
  - \( U = \mathbb{N} \times - \) (difference between output strings).

**DLRs as Ternary Dependent Relations**

\[ \delta_A \in \prod_{a \in A} 2^{\Delta(a) \times A} \]

\( \subseteq (\langle A \rangle) \): Allowed distances for \( a \in A \)
Denotational Semantics
Can we replace DLRs with metrics?

\[ d : \sigma \times \sigma \rightarrow |\sigma| \]
Can we replace DLRs with **metrics**?

\[ d : \llbracket \sigma \rrbracket \times \llbracket \sigma \rrbracket \rightarrow (|\sigma|) \]

In usual metric semantics each type \( \sigma \) is associated with a metric \( d : \llbracket \sigma \rrbracket \times \llbracket \sigma \rrbracket \rightarrow \mathbb{R}_\geq^\infty \) with a **fixed** distance space, \( \mathbb{R}_\geq^\infty \).
Can we replace DLRs with **metrics**?

\[ d : [\sigma] \times [\sigma] \rightarrow (|\sigma|) \]

In usual metric semantics each type \( \sigma \) is associated with a metric \( d : [\sigma] \times [\sigma] \rightarrow \mathbb{R}^\infty_{\geq 0} \) with a **fixed** distance space, \( \mathbb{R}^\infty_{\geq 0} \).

However, these semantics cannot account for ST\(\lambda\)C (they are not **cartesian closed**)

By letting the distance spaces \( (|\sigma|) \) **depend on** \( \sigma \) the picture changes
Suppose that

\[(\text{REAL} \rightarrow \text{REAL}) = (\mathbb{R}^\infty)^{\mathcal{I}(\mathbb{R})}\]

where \(\mathcal{I}(\mathbb{R}) = \{\text{compact intervals of } \mathbb{R}\} \).
Suppose that
\[ \langle \text{REAL} \to \text{REAL} \rangle = (\mathbb{R}_{\geq 0})^{\mathcal{I}(\mathbb{R})} \]
where \( \mathcal{I}(\mathbb{R}) = \{\text{compact intervals of } \mathbb{R}\} \).

Then \( p : \llbracket \text{REAL} \to \text{REAL} \rrbracket \times \llbracket \text{REAL} \to \text{REAL} \rrbracket \to (\mathbb{R}_{\geq 0})^{\mathcal{I}(\mathbb{R})} \), where
\[ p(f, g)(I) = \text{diam}(f(I) \cup g(I)) \].
Suppose that
\[
(\text{REAL} \to \text{REAL}) = (\mathbb{R}_\infty^{\geq 0})^\mathcal{I}(\mathbb{R})
\]
where \(\mathcal{I}(\mathbb{R}) = \{\text{compact intervals of } \mathbb{R}\}\).

Then \(p : [\text{REAL} \to \text{REAL}] \times [\text{REAL} \to \text{REAL}] \to (\mathbb{R}_\infty^{\geq 0})^\mathcal{I}(\mathbb{R})\), where \(p(f, g)(I) = \text{diam}(f(I) \cup g(I))\).

This way we get a partial metric space:
\[
\begin{align*}
\quad p(f, f) &\leq p(f, g), p(g, f) \\
p(f, g) & = p(g, f) \\
p(f, g) & = p(f, f) = p(g, g) \Rightarrow f = g \\
p(f, g) &\leq p(f, h) + p(h, g) - p(h, h)
\end{align*}
\]

And out of it, a proper metric space:
\[
p^* (f, g) = 2p(f, g) - p(f, f) - p(g, g)
\]

This way \(\mathbb{R}\) is extended to all simple types.
Suppose that
\[ (\text{REAL} \rightarrow \text{REAL}) = (\mathbb{R}^{\infty}_{\geq 0})^\mathcal{I}(\mathbb{R}) \]
where \( \mathcal{I}(\mathbb{R}) = \{\text{compact intervals of } \mathbb{R}\} \).

Then \( p : \left[ \text{REAL} \rightarrow \text{REAL} \right] \times \left[ \text{REAL} \rightarrow \text{REAL} \right] \rightarrow (\mathbb{R}^{\infty}_{\geq 0})^\mathcal{I}(\mathbb{R}) \), where
\[ p(f, g)(I) = \text{diam}(f(I) \cup g(I)) \]
This way we get a \textbf{partial metric space}:
\[
\begin{align*}
p(f, f) &\leq p(f, g), p(g, f) \\
p(f, g) &= p(g, f) \\
p(f, g) &= p(f, f) = p(g, g) \Rightarrow f = g \\
p(f, g) &\leq p(f, h) + p(h, g) - p(h, h)
\end{align*}
\]

And out of it, a proper \textbf{metric space}:
\[ p^*(f, g) = 2p(f, g) - p(f, f) - p(g, g) \]
▶ Suppose that 
\[(\text{REAL} \to \text{REAL}) = (\mathbb{R}_\geq 0)^\mathcal{I}(\mathbb{R})\]
where \(\mathcal{I}(\mathbb{R}) = \{\text{compact intervals of } \mathbb{R}\}.\)
▶ Then \(p : [\text{REAL} \to \text{REAL}] \times [\text{REAL} \to \text{REAL}] \to (\mathbb{R}_\geq 0)^\mathcal{I}(\mathbb{R}),\) where \(p(f, g)(I) = \text{diam}(f(I) \cup g(I)).\)
▶ This way we get a **partial metric space**:

\[
\begin{align*}
    p(f, f) &\leq p(f, g), p(g, f) \\
    p(f, g) &= p(g, f) \\
    p(f, g) &= p(f, f) = p(g, g) \Rightarrow f = g \\
    p(f, g) &\leq p(f, h) + p(h, g) - p(h, h)
\end{align*}
\]
▶ And out of it, a proper **metric space**:

\[
p^*(f, g) = 2p(f, g) - p(f, f) - p(g, g)
\]
▶ This way \(\mathbb{R}\) is extended to all simple types.
Coinduction and Game Semantics
Given an LTS \((A, \mathcal{LbL}, \rightarrow)\), we say that \(\delta : A \times A \rightarrow \mathbb{R}_\geq 0^\infty\) is a behavioural metric iff

\[
\delta(M, N) \geq \text{Obs}(M, N)
\]

\[
\delta(M, N) \geq \delta(L, P)\ \text{whenever } M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P
\]
Given an LTS \((A, \mathcal{L}B\mathcal{L}, \rightarrow)\), we say that \(\delta : A \times A \rightarrow \mathbb{R}_{\geq 0}^\infty\) is a **behavioural metric** iff

\[
\delta(M, N) \geq \text{Obs}(M, N)
\]

\[
\delta(M, N) \geq \delta(L, P) \text{ whenever } M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P
\]

The (pointwise) smallest behavioural metric is called **bisimilarity metric**.
Given an LTS \((A, \mathcal{LB}, \rightarrow)\), we say that \(\delta : A \times A \rightarrow \mathbb{R}_{\geq 0}^\infty\) is a **behavioural metric** iff

\[
\delta(M, N) \geq \text{Obs}(M, N)
\]

\[
\delta(M, N) \geq \delta(L, P) \text{ whenever } M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P
\]

The (pointwise) smallest behavioural metric is called **bisimilarity metric**.

Variations along this theme for LTSs in various monadic flavours are very well known and studied.
Given an LTS \((A, \mathcal{L}\mathcal{B}\mathcal{L}, \rightarrow)\), we say that \(\delta : A \times A \rightarrow \mathbb{R}_{\geq 0}^{\infty}\) is a behavioural metric iff

\[
\delta(M, N) \geq \text{Obs}(M, N)
\]

\[
\delta(M, N) \geq \delta(L, P) \quad \text{whenever} \quad M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P
\]

The (pointwise) smallest behavioural metric is called bisimilarity metric.

Variations along this theme for LTSs in various monadic flavours are very well known and studied.

When applied to LTSs coming from higher-order languages, these suffer from the same problems as MLRs:

- Only \(\lambda\)-calculi with bounded replication can be modeled [Gavazzo2018].
- The distance does not depend on the context.
Given an LTS \((A, \mathcal{L}, \rightarrow)\), we say that \(\delta : A \times A \rightarrow \mathbb{R}^\infty_{\geq 0}\) is a **behavioural metric** iff

\[
\delta(M, N) \geq \text{Obs}(M, N)
\]

\[
\delta(M, N) \geq \delta(L, P) \text{ whenever } M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P
\]

The (pointwise) smallest behavioural metric is called **bisimilarity metric**.

Variations along this theme for LTSs in various monadic flavours are very well known and studied.

When applied to LTSs coming from higher-order languages, these suffer from the same problems:

- Only \(\lambda\)-calculi with bounded replication can be modeled [Gavazzo2018].
- The label \(\ell\) is arbitrary.
- \(\delta(M, N)\) can be much bigger than \(\delta(L, P)\) for some \(\ell\).
Rather than taking $\mathbb{R}_{\geq 0}^\infty$ as the codomain of $\delta$, one could take a set $Q$ such that, e.g.

$$Q \sim \mathbb{R}_{\geq 0}^\infty \times (\mathcal{L}\mathcal{B}\mathcal{L} \rightarrow Q)$$

We can capture contextual bisimilarity [Larsen85]. The properties of the induced notion of distance are still being scrutinized.
Rather than taking $\mathbb{R}_{\geq 0}^\infty$ as the codomain of $\delta$, one could take a set $Q$ such that, e.g.

$$Q \simeq \mathbb{R}_{\geq 0}^\infty \times (\mathcal{L}\mathcal{B}\mathcal{L} \to Q)$$

Then, the bisimulation game could be made contextual:

$$\pi_1(\delta(M, N)) \geq \text{Obs}(M, N)$$

$$\pi_2(\delta(M, N), \ell) \geq \delta(L, P)$$ whenever $M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P$
Rather than taking $\mathbb{R}_{\geq 0}^\infty$ as the codomain of $\delta$, one could take a set $Q$ such that, e.g.

$$Q \simeq \mathbb{R}_{\geq 0}^\infty \times (\mathcal{L}\mathcal{B}\mathcal{L} \to Q)$$

Then, the bisimulation game could be made **contextual**:

$$\pi_1(\delta(M, N)) \geq \text{Obs}(M, N)$$

$$\pi_2(\delta(M, N), \ell) \geq \delta(L, P)$$

whenever $M \xrightarrow{\ell} L \land N \xrightarrow{\ell} P$.

We can capture contextual bisimilarity [Larsen85].

The properties of the induced notion of distance are still being scrutinized.
Strategies can be seen as sets of sequences of moves rather than as functions.

The distance $\delta_{\tau}(f, g)$ between two such strategies can then be taken as $f \cup g$. 

Quite surprisingly $\delta(f \circ h, g \circ k) = \delta(f, g) \circ \delta(h, k)$.

Very close to program pair distances.

How about playing on abstract moves?
Strategies can be seen as **sets of sequences of moves** rather than as functions.

The distance $\delta_\tau(f, g)$ between two such strategies can then be taken as $f \cup g$.

$$\delta_{\text{REAL}}(\lambda x.x + 2, \lambda x.x + 3) =$$

Quite surprisingly $\delta(f \circ h, g \circ k) = \delta(f, g) \circ \delta(h, k)$.

- Very close to program pair distances.
- How about playing on **abstract moves**?
Questions?