1. Implicit Programming

- Programming paradigm in which some function arguments are inferred by their types
- Proven to be practical by many programming languages, such as Haskell, Scala and C++

2. Problems of state-of-the-art Implicit Programming

- Current practices are either well-formalized but restricted (e.g. Haskell) or not well-\formalized (e.g. Scala and C++)
- Coherence of programs, especially with overlapping instances, is not well-studied:

  module A

  let f : \forall b \rightarrow b =
  implicit { \lambda x. x : \forall a \rightarrow a, \lambda x. x + 1 : \text{int} \rightarrow \text{int} } in

  ?(b \rightarrow b)

  - the program is coherent if \text{B.a = 2} and \text{B.b = 'x'}
  - no system in the literature can make this program coherent
  - No existing system supports all useful features in one system

3. Solution: Implicit Calculus $\lambda_p$

- Core calculus for implicit programming
- Encompasses all the current practices
- Type-safe program = coherent program

\[
f = \Lambda X. (\text{rule}[\forall b. \{ \forall a. a \rightarrow \text{int} \rightarrow \text{int} \} \Rightarrow b \rightarrow b](?b \rightarrow b))[X]
\]

\[
\text{let } a = \text{f int 1}
\]

\[
\text{let } b = \text{f char 'x'}
\]

- this program is coherent according to our operational semantics

4. Formalism

- Syntax

\[
e \ ::= ?\rho \mid \text{rule } \rho \ e \mid e[\tau] \mid e \ e_p \pi \tau
\]

- Type System

\[
(\text{OpRule}) \quad \Gamma \vdash e : \tau \\
(\text{OpInst}) \quad \Gamma \vdash e[\tau] : \theta \Rightarrow \theta' \Rightarrow \theta \\
(\text{OpApp}) \quad \Gamma \vdash e \ e_p : \tau \\
(\text{StaRes}) \quad \Gamma \vdash \theta : \theta_p \eta
\]

- Operational Semantics

<table>
<thead>
<tr>
<th></th>
<th>Haskell</th>
<th>Scala</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>local instances</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>overlapping instances</td>
<td>▲</td>
<td>X</td>
<td>▲</td>
</tr>
<tr>
<td>higher-order rules</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>