Programming Language Theory

Type and Effect Systems
Type-and-effect systems

New topic: An elegant framework to extend type systems to track “things that may happen” (effects) during evaluation.

Plain-old type systems have judgments like $\Gamma \vdash e : \tau$ to mean:

- $e$ won’t get stuck
- If $e$ produces a value, then that value has type $\tau$

Adding effects reuses the “plumbing” of typing rules to compute something about “how $e$ executes”.

- There are many things we may want to conservatively approximate.
  - Example: What exceptions might get thrown.
- All effect systems are very similar, especially treatment of functions.
  - Example: All values have no effect since their “computation” does nothing.
Start with a type system

\[ \tau ::= \tau \rightarrow \tau \mid \tau \ast \tau \mid B \]

\[ e ::= x \mid \lambda x.\ e \mid e\ e \mid (e,e) \mid e.1 \mid e.2 \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e \text{ else } e \mid \text{raise } s \mid \text{try } e \text{ handle } s \]

\[ \Gamma ::= \cdot \mid \Gamma, x : \tau \]

\[ \Gamma \vdash e : \tau \]

\[ \Gamma @ x \rightsquigarrow \tau \quad \Gamma, x : \tau_a \vdash e_b : \tau_r \quad \Gamma \vdash e_f : \tau_a \rightarrow \tau_r \quad \Gamma \vdash e_a : \tau_a \]

\[ \Gamma \vdash \lambda x.\ e_b : \tau_a \rightarrow \tau_r \]

\[ \Gamma \vdash (e_1, e_2) : \tau_1 \ast \tau_2 \quad \Gamma \vdash e_p : \tau_1 \ast \tau_2 \quad \Gamma \vdash e_p.1 : \tau_1 \quad \Gamma \vdash e_p.2 : \tau_2 \]

\[ \Gamma \vdash \text{true} : B \quad \Gamma \vdash \text{false} : B \]

\[ \Gamma \vdash \text{if } e_b \text{ then } e_t \text{ else } e_f : \tau \]

\[ \Gamma \vdash \text{if } e_b \text{ then } e_t \text{ else } e_f : \tau \]

\[ \Gamma \vdash \text{raise } s : \tau \quad \Gamma \vdash \text{try } e_b \text{ handle } s \text{ e}_h : \tau \]

(In this example, exceptions raise constant strings \textbf{s}.)

Matthew Fluet
Programming Language Theory
Lecture 17 3
Add effects

\[\begin{align*}
\epsilon &::= \{s_1, \ldots, s_n\} \\
\tau &::= \tau \stackrel{\epsilon}{\rightarrow} \tau \mid \tau \star \tau \mid B \\
e &::= x \mid \lambda x. \ e \mid e \ e \mid (e, e) \mid e.1 \mid e.2 \mid \\
&\text{true} \mid \text{false} \mid \text{if } e \text{ then } e \text{ else } e \mid \text{raise } s \mid \text{try } e \text{ handle } s \ e \\
\Gamma &::= \cdot \mid \Gamma, x : \tau
\end{align*}\]
Key facts

Soundness:
If \( \Gamma \vdash e : \tau; \epsilon \) and \( e \) raises uncaught exception \( s \), then \( s \in \epsilon \).

- Corollary to Preservation and Progress
  (once you define the operational semantics for exceptions)

All effect systems work this way:
- Values effectless
- Functions have latent effects
- Conservative due to control-flow (if and try/handle)
- Often some way to mask effects (catch an exception)

Only a couple rules special to this effect system
- Also, not always sets and \( \cup \)
More general rules

Every effect system also substantially more expressive via appropriate subsumption:

- Typing rule for subeffecting (also useful for Preservation)
- Subtyping of function types is covariant in latent effects

\[
\Gamma \vdash e : \tau; \epsilon' \quad \epsilon' \subseteq \epsilon \\
\Gamma \vdash e : \tau; \epsilon \\
\tau_a \leq \tau'_a \quad \tau'_r \leq \tau_r \\
\epsilon' \subseteq \epsilon \\
\tau'_a \xrightarrow{\epsilon'} \tau'_r \leq \tau_a \xrightarrow{\epsilon} \tau_r
\]

Not shown:

- Also want effect polymorphism (type variables ranging over effects) for higher-order functions like map
Other examples

▶ Definitely terminates (true) or possibly diverges (false)
  ▶ Give `fix e` effect `false`
  ▶ Give values effect `true`
  ▶ Treat `∪` as `and`
  ▶ No change to rules for functions, pairs, conditionals, etc.

▶ What type casts might occur
▶ Are certain variables always accessed in critical sections
▶ Does code obey a locking protocol
▶ Does code only access memory regions that haven’t been deallocated
▶ ...

Really a general way to lift static analysis to higher-order functions.

▶ Key is recognizing “from a mile away” when an effect system is the right tool