



Error Localization for Sequential Effect Systems

Colin S. Gordon and Chaewon Yun, Drexel University
SAS 2023

Effect Systems

Extended type systems that track what effect a program may have as it executes

E.g., throwing certain checked exceptions, accessing memory regions / lock-protected data

Commonly given by join semilattices

$$\text{T-FUN} \frac{\Gamma, x : \tau \vdash e : \tau' \mid \chi}{\Gamma \vdash (\lambda x. e) : \tau \xrightarrow{\chi} \tau' \mid \perp}$$

$$\text{T-CALL} \frac{\Gamma \vdash e_1 : \tau \xrightarrow{\chi} \tau' \mid \chi_1 \quad \Gamma \vdash e_2 : \tau \mid \chi_2}{\Gamma \vdash e_1 e_2 : \tau' \mid \chi_1 \sqcup \chi_2 \sqcup \chi}$$



Effect Systems Theory != Implementation

Theory of effect systems: Global

Take least-upper-bound of every action in the function body, check $\sqsubseteq$ bound

E.g., if method body has effect X , and is annotated with effect (e.g., throws) Y , check $X \sqsubseteq Y$

This yields useless error messages!

```
void f() throws Bar { ...<300 LOC>... throw Foo(); ... <400 LOC> }
```

Error message becomes roughly: “LUB of the whole method is not less than Bar”

No information about *where* the error is!



Effect Systems Theory != Implementation

Implementation of effect systems: Local

Individually check every action in the function body is $\sqsubseteq$ bound

E.g., for `void foo() throws A, B, C { a(); b(); d(); c() }`
check `a()`'s effect is $\sqsubseteq \{A,B,C\}$, then `b()`'s, ...

Good error message!

If effect of `d()` is not $\sqsubseteq \{A,B,C\}$, error is issued at the call to `d()`!

Works because of a property of join semilattices: if S is the set of all method body subexpression effects,

$$(\sqcup S) \sqsubseteq Y \Leftrightarrow \forall s \in S, (s \sqsubseteq Y)$$



Sequential Effect Systems?

a.k.a. *flow-sensitive* effect systems

Java's checked exceptions are *not* flow-sensitive:

```
if (a) { throw ExcA(); } if (b) { throw ExcB(); } // exceptions are {ExcA, ExcB}
```

```
if (b) { throw ExcB(); } if (a) { throw ExcA(); } // exceptions still {ExcA, ExcB}
```

What about tracking lock acquisition & release order? Order of security-sensitive operations?

Sets of finite traces?

Sequential Effect Systems

Many varied characterizations; see Gordon TOPLAS 2021 for a survey

We'll use *effect quantales* $Q = (E, \triangleright, \sqcup, I)$ where

$(E, \sqcup)$ is a *partial* join semilattice — for subeffecting

$(E, \triangleright, I)$ is a partial monoid — for ordering operations

Plus some conditions on partiality, distributivity laws

Can generally derive an iteration operation $*$ for loops, others for non-local control

Also captures traditional effect systems, if $\triangleright := \sqcup, I := \perp$

$$\begin{array}{c} \text{T-APP} \\ \frac{\Gamma \vdash e_1 : \tau \xrightarrow{\chi} \sigma \mid \chi_1 \quad \Gamma \vdash e_2 : \tau \mid \chi_2}{\Gamma \vdash e_1 e_2 : \sigma \mid \chi_1 \triangleright \chi_2 \triangleright \chi} \\ \\ \text{T-LAMBDA} \\ \frac{\Gamma, x : \tau \vdash e : \sigma \mid \chi}{\Gamma \vdash (\lambda x. e) : \tau \xrightarrow{\chi} \sigma \mid I} \end{array}$$



Implementing Sequential Effect Systems?

No existing solution for localized error reports, prior work either:

- Computes the body effect with $\triangleright$ and $\sqcup$ and compares the result to a bound

- Bad error messages

- Uses type inference with constraints using $\triangleright$ and $\sqcup$ (and $\sqsubseteq$)

- Local errors, but non-deterministic locations, maybe unrelated to mistake location



Localizing Sequential Effect Systems

Key challenge: lack of way to relate local and global constraints

Need equivalent of $(\sqcup S) \sqsubseteq Y \Leftrightarrow \forall s \in S, (s \sqsubseteq Y)$ on JSLs, but for effect quantales

We give a way to generate local constraints *relative to global constraints*, for more precise error messages



Residuated Semigroups

$Y \setminus Z$ is “what’s left to do in Z after doing Y ”

Today: we’ll ignore a couple subtleties, and present slightly simpler solution than the paper

Paper presentation builds on what we cover today

Solution seems to exist for most effect quantales, but doesn’t follow from EQ laws

A *partial right-residuated semigroup* $(S, \triangleright, \sqcup, \setminus, I)$ is a partial ordered semigroup $(S, \triangleright, \sqcup, I)$ where

$\setminus$ is partial

$$x \sqsubseteq y \setminus z \Leftrightarrow (y \triangleright x) \sqsubseteq z$$

With a couple extra laws, this is actually applicable to effect quantales!



Local Implementation of Sequential Effects

For $e_1; e_2$, with effects X_1 and X_2 : $X_1 \triangleright X_2 \sqsubseteq X$ iff $X_2 \sqsubseteq X_1 \setminus X$, i.e.,

$X_1 \setminus X$ is defined — ensuring it's possible that *some* continuation after effect X_1 can still be $\sqsubseteq X$

Fails, for example, if $X \sqsubseteq X_1$

X_2 is less than that possible continuation

If e_2 is complex, $X_1 \setminus X$ becomes new bound when recursively checking e_2

So for a program fragment $b(); c();$ with finite trace effect BC and target bound AC :

Implementation can *flag $b()$ as an error* because $B \setminus AC$ is undefined!



More in the Paper

Develops this with more nuance for a typed lambda calculus

Describes the checking algorithm in detail

Proves the checking algorithm for a typed lambda calculus gives the earliest possible type checking failure according to program execution order

Describes an implementation for Java, using the Checker Framework

Shows error localization for a couple sequential effect systems

Like the atomicity example on the next slide

Local Errors for Atomicity Effects

Flanagan and Qadeer (TLDI'03, PLDI'03) give effect systems for *atomicity*

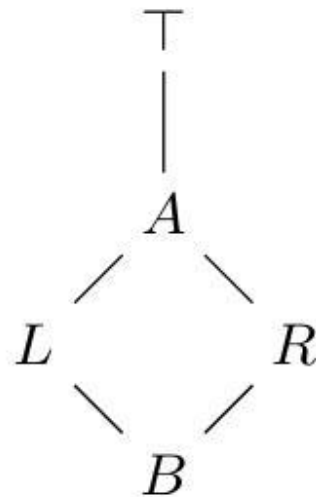
Early version is a very small effect quantale

Atomic actions have effect A , and any sequence of effects $(R|B)^*A?(L|B)^* \sqsubseteq A$

Implemented for Java in 2003, found atomicity bugs in JDK 1.4!

No information about error reporting locations

Ran our tool on JDK 1.4 source, and we get precise error locations



▷	B	L	R	A	$\top$
B	B	L	R	A	$\top$
R	R	A	R	A	$\top$
L	L	L	$\top$	$\top$	$\top$
A	A	A	$\top$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$



Atomicity

```
@Atomic
public boolean contentEquals(StringBuffer sb) {
    if (count != sb.length()) // length() is atomic
        return false;
    char v1[] = value;
    char v2[] = sb.getValue(); // <-- Error: second atomic operation
    int i = offset;
    int j = 0;
    int n = count;
    while (n-- != 0) {
        if (v1[i++] != v2[j++])
            return false;
    }
    return true;
}
```

} Effect of first 4 lines is top
(**@NonAtomic**)

Error reported on line 4 because
@NonAtomic\@Atomic is undefined

Fig. 6: Excerpt from JDK1.4 String implementation.



Wrapping Up

Identifying how traditional effect system implementations deviate from theory in a controlled way

Extending this idea to sequential effect systems using residuated semigroup theory

Practical implementation for Java

<https://github.com/swlogiclab/checker-framework-effects>

Artifact linked from paper

Validation on known & new sequential effect systems