ReCheck: Automated Contextual Improvement Verification for Functional Calculi across User-Defined Operational Semantics

Makoto Hamana | Kento Emoto
Kyushu Institute of Technology, Japan

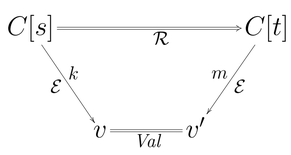
Problem and Goal

- Program optimizations and rewrite rules should preserve **efficiency**, not only extensional equivalence.
- Sands' **contextual improvement** requires that replacing s by t never increases evaluation cost in any surrounding program context.
- Existing proofs are often manual and tied to one specific operational semantics.
- We target **user-defined operational semantics**: syntax classes, evaluation contexts, evaluation rules, and refinement rules.

Goal: verify contextual improvement *automatically* by critical-pair analysis.

Contextual Improvement

A set of refinement rules $\mathcal{R}$ is **a contextual improvement** with respect to evaluation rules $\mathcal{E}$ if $s \Rightarrow_{\mathcal{R}} t$ and



closed terms

with $k \geq m$.

same observable value, with no extra evaluation cost

Key difficulty. The definition quantifies over *all contexts* C.

TERS: Term Evaluation and Refinement Systems

A TERS specifies:

- a signature,
- syntax classes (values, computations, answers, handlers, ...),
- evaluation contexts $ECtx$,
- evaluation rules $\mathcal{E}$,
- refinement rules $\mathcal{R}$.

Evaluation and refinement relations

$$\frac{(l \to r) \in \mathcal{E} \quad E \in ECtx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]} \qquad \frac{(l \Rightarrow r) \in \mathcal{R} \quad C \in \mathcal{C}}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$$

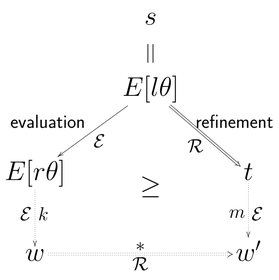
This cleanly separates **deterministic evaluation** from **global refinement**.

Main Theorem (Muroya, Hamana, FLOPS 2024)

If
1. $\mathcal{E}$ is deterministic,
2. $\mathcal{R}$ is value-invariant,
3. the TERS $(\mathcal{E}, \mathcal{R})$ is **locally coherent**,

then $\mathcal{R}$ is a **contextual improvement** with respect to $\mathcal{E}$.
This reduces a universal semantic property to a **local overlap condition** between evaluation and refinement.

Local Coherence Criterion



with $1 + k \geq m$.

Lemma. For a **well-behaved** TERS, local coherence holds iff every critical pair between $\mathcal{E}$ and $\mathcal{R}$ is joinable.

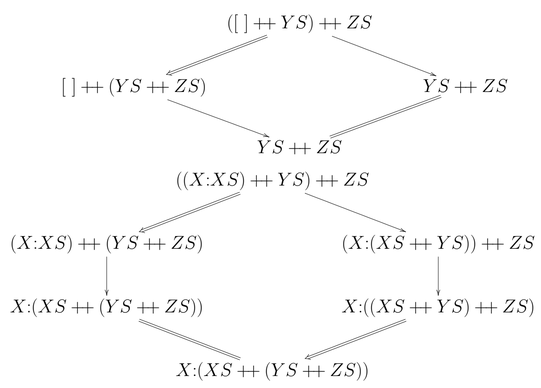
Consequence: contextual improvement is checked by finite overlap analysis.

Example 1: Append Fusion

Values $V ::= [\,] \mid V : V$
Evaluation ctxts $E ::= \Box \mid E +\!\!+ t \mid V +\!\!+ E \mid E : t \mid V : E$
Evaluation rules $\mathcal{E}$ $[\,] +\!\!+ ys \to ys$
$(x : xs) +\!\!+ ys \to x : (xs +\!\!+ ys)$
Refinement rules $\mathcal{R}$ $(xs +\!\!+ ys) +\!\!+ zs \Rightarrow xs +\!\!+ (ys +\!\!+ zs)$

Associativity of append is treated as an *efficiency-improving* rewrite under deterministic evaluation.

Critical Pairs for Append Fusion



Both critical pairs are joinable, hence the associativity refinement rule is **a contextual improvement**.

Tool Support: ReCheck

- ReCheck extends SOL and automates critical-pair analysis for contextual improvement.
- Input: syntax classes, evaluation contexts, evaluation rules, and refinement rules.
- Output: overlaps, joinability checks, and non-joinable counterexamples.

Specification → critical pairs → joinability → certificate

Example 2: Maybe Monad

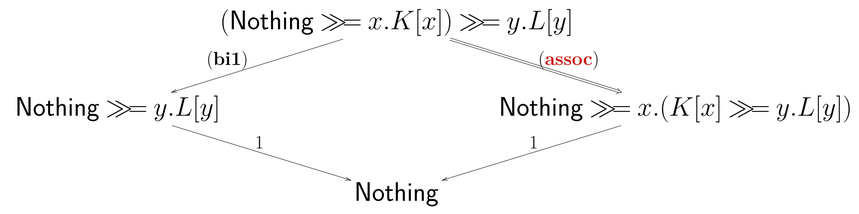
The same method also handles user-defined calculi beyond first-order list programs.
Consider the Maybe monad and the monad laws.
Evaluation rules $\mathcal{E}$

$$\begin{aligned}
\lambda(x.M[x])@V &\Rightarrow M[V] && (\beta)\\
\texttt{return}(M) &\Rightarrow \texttt{Just}(M) && (\text{ret})\\
\texttt{Nothing} \gg\!= x.F[x] &\Rightarrow \texttt{Nothing} && (\text{bi1})\\
\texttt{Just}(M) \gg\!= x.F[x] &\Rightarrow F[M] && (\text{bi2})\\
& && (1)
\end{aligned}$$

Refinement rules $\mathcal{R}$

$$\begin{aligned}
\mathsf{return}(x) \gg\!= K &\Rightarrow K[x] && (\text{idL})\\
M \gg\!= x.\,\mathsf{return}(x) &\Rightarrow M && (\text{idR})\\
(M \gg\!= K) \gg\!= L &\Rightarrow M \gg\!= (x.\,K[x] \gg\!= L) && (\text{assoc})
\end{aligned}$$

A representative critical pair is:



ReCheck enumerates all critical-pair patterns for this calculus and verifies jinability, yielding contextual improvement for the refinement rules $\mathcal{R}$.

Example 3: Untyped call-by-value λ-calculus

Values $V ::= x \mid \lambda x.M$
Evaluation ctxts $E ::= \Box \mid E\,M \mid V\,E$
Evaluation rule $(\lambda x.M)\,V \to M[x := V]$

The same framework also supports second-order TERS presentations and **refinement rules** such as

$$\lambda(x.M[x])@V \Rightarrow M[V], \qquad \lambda(x.V@x) \Rightarrow V.$$

Example 4: Map/map fusion in Haskell

```
map f []     = []
map f (x:xs) = f x : map f xs
(.) f g x    = f (g x)

{-# RULES
 "map/map" forall f g xs. map f (map g xs) = map (f . g) xs
#-}
```

Example 5: Lazy program optimization

Program as evaluation rules $\mathcal{E}$

```
(rep1)  replicate(z)      => []
(rep2)  replicate(s(N))   => s(z) : replicate(N)
(take1) take(z, XS)       => []
(take2) take(s(N), [])    => []
(take3) take(s(N), X:XS)  => X : take(N,XS)
(ones)  ones              => s(z) : ones
```

Refinement rule $\mathcal{R}$

$$(\text{conj}) \quad \texttt{take}(N, \texttt{ones}) \Rightarrow \texttt{replicate}(N)$$

ReCheck reports 2 critical pairs, with 1 non-joinable; after adding

$$\texttt{take}(N, s(z)\texttt{:ones}) \Rightarrow \texttt{replicate}(N),$$

all critical pairs become joinable.

Other Examples

Computational λ-calculus, Call-by-need λ, Effect handlers, map/append, **copying natural numbers**, $\cdots$

$$\mathsf{copy}(Z) \to Z; \quad \mathsf{copy}(S(N)) \to S(\mathsf{copy}(N))$$
$$\mathsf{copy}(\mathsf{copy}(N)) \Rightarrow N \qquad (\text{copy})$$

Take-Home Message

- Contextual improvement can be reduced to **joinability of critical pairs**.
- This yields a **semantics-parametric** and **automation-friendly** verification method.
- ReCheck is effective for user-defined TERS, including ADT-heavy examples where SMT-centered tools are often less suitable.

Tools	Typical strength
Mochi / RCaml	safety or refinement-type verification with SMT
Timbuk	TRS + tree-automata completion
ReCheck	contextual improvement for user-defined TERS, strong on ADTs

Keywords: contextual improvement, operational semantics, term rewriting, program optimization, critical pairs, ReCheck