Term Evaluation Systems with Refinements for Contextual Improvement by Critical Pair Analysis

Makoto Hamana
Kyushu Institute of Technology

Koko Muroya
Ochanomizu University

IWC, 1st September, 2025
(based on the paper in FLOPS 2024)

Vefication Problem in FP

Functional program

```
[]       ++ ys ⟶ ys
(x:xs) ++ ys ⟶ x : (xs ++ ys)
```

Verify equality:

```
(xs ++ ys) ++ zs = xs ++ (ys ++ zs)
```

Vefication Problem in FP

Functional program

```
[]     ++ ys ⟶ ys
(x:xs) ++ ys ⟶ x : (xs ++ ys)
```

Verify equality:

```
(xs ++ ys) ++ zs = xs ++ (ys ++ zs)
```

▷ Inductive theorem?

Vefication Problem in FP

Functional program

```
[]     ++ ys ⟶ ys
(x:xs) ++ ys ⟶ x : (xs ++ ys)
```

Verify equality:

```
(xs ++ ys) ++ zs = xs ++ (ys ++ zs)
```

- ▷ Inductive theorem?
- ▷ Real programming languages use **deterministic evaluation** mecahnisms
  - Call-by-value: OCaml
  - Call-by-name, call-by-need: Haskell
- ▷ HO functions, polymorphsim, e.g.

```
map f (map g xs) = map (f o g) xs
```

Vefication Problem in FP

Functional program

```
[]     ++ ys ⟶ ys
(x:xs) ++ ys ⟶ x : (xs ++ ys)
```

Verify **improvement**:

```
(xs ++ ys) ++ zs ⇛ xs ++ (ys ++ zs)
```

- Inductive theorem?
- Real programming languages use **deterministic evaluation** mecahnisms
  - Call-by-value: OCaml
  - Call-by-name, call-by-need: Haskell
- HO functions, polymorphsim, e.g.

```
map f (map g xs) = map (f ∘ g) xs
```

Term Evaluation and Refinement System (TERS)

Signature Σ — $[\,]: 0,\ (:): 2,\ (+\!\!+): 2$

Values Val — $V ::= [\,] \mid V : V$

Evaluation contexts $Ectx$ — $E ::= \square \mid E +\!\!+ t \mid V +\!\!+ E \mid E : t \mid V : E$

Evaluation rules $\mathcal{E}$

$$[\,] +\!\!+ ys \to ys$$

$$(x : xs) +\!\!+ ys \to x : (xs +\!\!+ ys)$$

Refinement rule $\mathcal{R}$

$$(xs +\!\!+ ys) +\!\!+ zs \Rightarrow xs +\!\!+ (ys +\!\!+ zs)$$

Q. Is $\mathcal{R}$ contextual improvement?

Term Evaluation System (TERS)

Evaluation

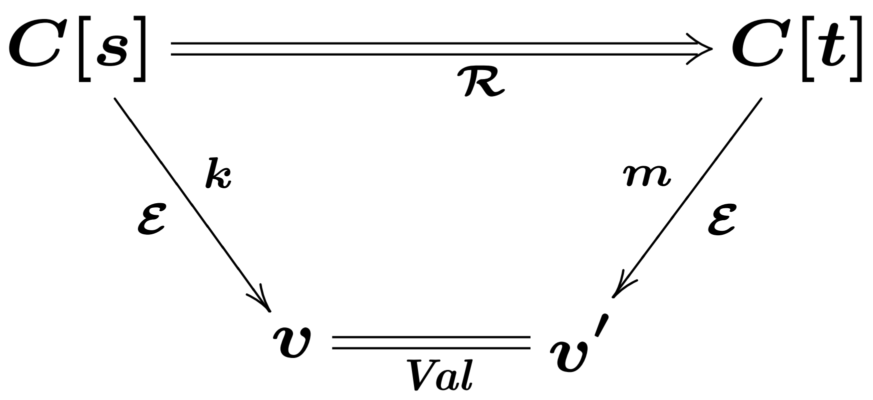
$$\frac{(l \longrightarrow r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \longrightarrow_{\mathcal{E}} E[r\theta]}$$

Refinement

$$\frac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$$

Contextual Improvement

A set $\mathcal{R}$ of refinement rules is **contextual improvement** w.r.t. a set $\mathcal{E}$ evaluation rules if for all contexts C,

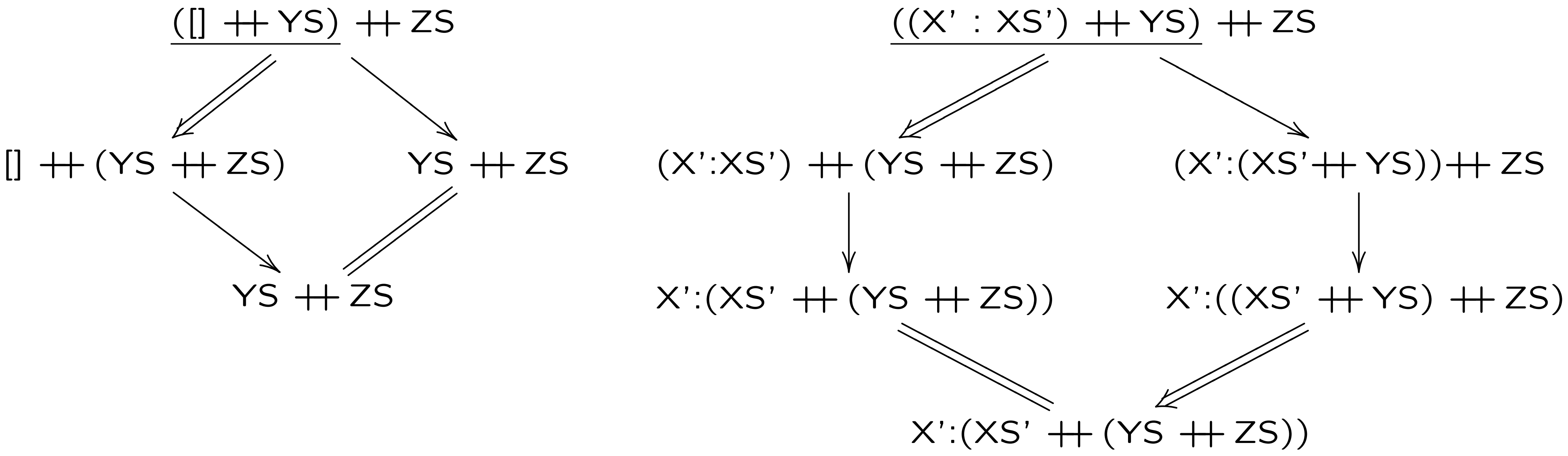


with $k \geq m$

- Difficult to prove because of universal quantification of contexts

Is $\mathcal{R}$ contextual improvement?

Yes, by checking critical pairs between $\mathcal{E}$ and $\mathcal{R}$



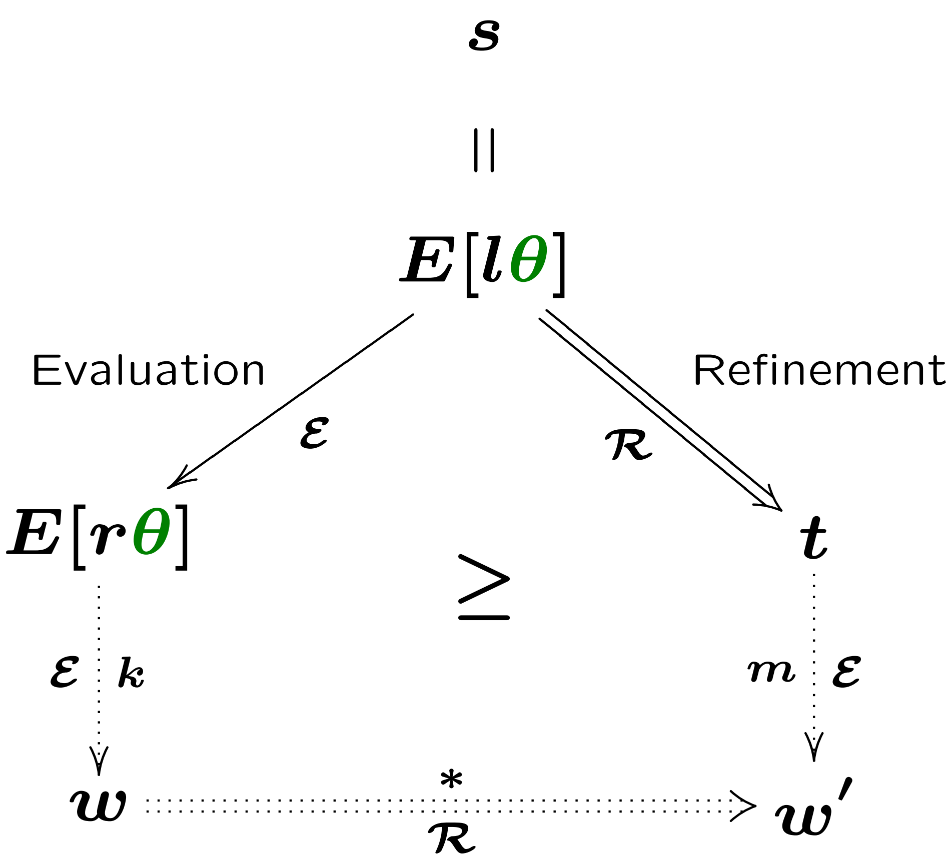
Main Theorem [FLOPS'24, Muroya&H.]

If

1. $\mathcal{E}$ is deterministic (ensured by evaluation contexts)
2. $\mathcal{R}$ is value-invariant (a refinement of value is a value)
3. TERS $(\mathcal{E}, \mathcal{R})$ is **locally coherent**

then $\mathcal{R}$ is **contextual improvement** wrt $\mathcal{E}$.

Local Coherence



with $\mathbf{1} + \boldsymbol{k} \geq \boldsymbol{m}$

- How to check?

Lemma [FLOPS'24, Muroya&H.]

A **well-behaved** TERS is locally coherent iff all its critical pairs are joinable.

Lemma [FLOPS'24, Muroya&H.]

A **well-behaved** TERS is locally coherent iff all its critical pairs are joinable.

Well-behaved TERS $(\mathcal{E}, \mathcal{R})$

- Evaluation contexs are defined inductively.
- Refinement respects evaluation contexts: $\mathit{Ectx} \ni E \Rightarrow_{\mathcal{R}} E' \in \mathit{Ectx}$
- $\mathcal{R}$ linear (wrt non-value metavariable)
- $\mathcal{E}$ left-linear (wrt non-value metavariable)
- For every non-value metavariable M,
  if $l[M \mapsto \overline{x}.\square] \in \mathit{Ectx}$ then $r[M \mapsto \overline{x}.\square] \in \mathit{Ectx}$.

Highlights

- Developed this critical pair analysis method of **contextual improvement** for FO and SO TERS
- Developed tool ReCheck as an extension of SOL
- Checked various examples
  - Haskell: map/map, lazy lists, diverge, $\cdots$
  - Calculi: left-to-right call-by-value $\boldsymbol{\lambda}$-calculus, call-by-need $\boldsymbol{\lambda}$-calculus, computaional $\boldsymbol{\lambda}$-caluclus, effect handers

Example: Map/Map Fusion

```
map :: (a -> b) -> [a] -> [b]
map f []     = []
map f (x:xs) = f x : map f xs

(.) :: (b -> c) -> (a -> b) -> (a -> c)
(.) f g x = f (g x)

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

Demo of ReCheck

Lazy program [Kikuchi, Sasano, Aoto, PPDP19]

```
(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 using the infinite list `ones`

```
(conj)  take(N, ones) => replicate(N)
```

- N elements from `ones` is refined to N copies of `s(z)`.
- Inductive theorem proving without SN
- ReCheck reports 2 CPs and 1 non-joinable
- Adding

```
take(N,s(z):ones) => replicate(N)
```

all joinable, then contextual improvement

Simulating Context-Sensitive Rewriting

Prop. Let $(\Sigma, \mathcal{R}, \mu)$ be a context-sensitive TRS [Lucus'92,...] and $(\Sigma, \mathcal{E}, Ectx, Val)$ be the correponding nondeterministic TES. We have:

$$t \hookrightarrow_{\mathcal{R},\mu} u \quad \Longleftrightarrow \quad t \rightarrow_{\mathcal{E}} u$$

Idea: replacement maps μ are simulated by evaluation contexts

Example

- ▷ Context-sensitive $\Rightarrow$ Term Evalation System
- ▷ $\mu(\text{if}) = \{1\}$ $\Leftrightarrow$ $E ::= \Box \mid \text{if}(E, t_1, t_2) \mid \cdots$
- ▷ $\mu(+) = \{1, 2\}$ $\Leftrightarrow$ $E ::= \Box \mid E + t \mid t + E \mid \cdots$ (non-determistic TES)
- ▷ $\mu(+) = \{1, 2\}$ $\Leftarrow$ $E ::= \Box \mid E + t \mid v + E \mid \cdots$ (determistic TES) with starategy

Simulating Context-Sensitive Rewriting

Prop. Let $(\boldsymbol{\Sigma}, \boldsymbol{\mathcal{R}}, \boldsymbol{\mu})$ be a context-sensitive TRS [Lucus'92,...] and $(\boldsymbol{\Sigma}, \boldsymbol{\mathcal{E}}, \boldsymbol{Ectx}, \boldsymbol{Val})$ be the correponding nondeterministic TES. We have:

$$t \hookrightarrow_{\mathcal{R},\mu} u \quad \Longleftrightarrow \quad t \rightarrow_{\mathcal{E}} u$$

Idea: replacement maps $\boldsymbol{\mu}$ are simulated by evaluation contexts

Example

- ▷ Context-sensitive $\Rightarrow$ Term Evalation System
- ▷ $\mu(\text{if}) = \{1\}$ $\Leftrightarrow$ $E ::= \square \mid \text{if}(E, t_1, t_2) \mid \cdots$
- ▷ $\mu(+) = \{1, 2\}$ $\Leftrightarrow$ $E ::= \square \mid E + t \mid t + E \mid \cdots$ (non-determistic TES)
- ▷ $\mu(+) = \{1, 2\}$ $\Leftarrow$ $E ::= \square \mid E + t \mid v + E \mid \cdots$ (determistic TES) with starategy

▶ Based on this simulation, SOL 2025 (TERS evaluation, local coherence) has also the feature of checking CS confluence ▶ Prticipate CoCo 2025, CSR category

Future Work

- Alternative to (higher-order polymorphic) inductive theorem proving
- Rewriting properties on TERS: confluence, termination, etc.
- **Robust** Haskell's rewrite rule verifier
- Detailed comparison with FP verifier based on SMT solver
  Mochi, RCaml [Sato,Unno, Kobayashi 12,...],
  Timbuk [Haudebourg, Genet, Jenen 20]
  - These tools are good on problem involving interger constraints, but weak on ones involving algebraic datatypes
  - E.g. These could not verify `copy(copy(N) = N)` under `copy(0) = 0; copy(s(N)) = s(copy(N))`