Term Evaluation Systems with Refinements

Koko Muroya
(RIMS, Kyoto University)

Makoto Hamana
(Kyusyu Institute of Technology)

FLOPS 2024 (Kumamoto), 15 May 2024

Two kinds of rewriting in functional programming

evaluation	refinement
run-time rewriting	compile-time rewriting
beta-reduction $E[(\lambda x\,.\,t)\ v] \to E[t\{v/x\}]$	beta-law $C[(\lambda x\,.\,t)\ v] \Rightarrow C[t\{v/x\}]$ eta-law $C[\lambda x\,.\,v\ x] \Rightarrow C[v]$

Two kinds of rewriting in functional programming

- with different roles:

evaluation	refinement
run-time rewriting	compile-time rewriting
beta-reduction $E[(\lambda x\,.\,t)\ v] \to E[t\{v/x\}]$	beta-law $C[(\lambda x\,.\,t)\ v] \Rightarrow C[t\{v/x\}]$ eta-law $C[\lambda x\,.\,v\ x] \Rightarrow C[v]$
specification of *operational semantics*	model of *optimisation* to be **validated** wrt. evaluation

Validation of refinement $\Rightarrow$ wrt. evaluation $\rightarrow$

- by proving *contextual improvement* [Sands '96]
  - "two terms t, u have the same result, and u requires *fewer steps* than t"
  - $t \precsim u \overset{\Delta}{\iff} \forall C, v, v'.$ if $C[t] \rightarrow^k v$ then $C[u] \rightarrow^m v' \wedge v =_{Val} v' \wedge k \geq m$

$\lambda x.t =_{Val} \lambda x'.t'$

$\underline{n} =_{Val} \underline{n}$

- namely, by proving: refinement $t \Rightarrow u$ implies contextual improvement $t \precsim u$
  - our goal: develop a proof methodology

Our goal

- develop a proof methodology
  - to validate refinement $\Rightarrow$ wrt. evaluation $\rightarrow$
  - to establish: refinement $\Rightarrow$ implies contextual improvement $\precsim$
- by a *rewriting-theoretic* approach

contribution 1

  - exploiting critical pair analysis
    - in the hope of (partial) automation
  - by *rewriting-theoretic* modelling of both refinement $\Rightarrow$ and evaluation $\rightarrow$

contribution 2

Our goal

- develop a proof methodology
  - to validate refinement $\Rightarrow$ wrt. evaluation $\rightarrow$
  - to establish: refinement $\Rightarrow$ implies contextual improvement $\precsim$
- by a *rewriting-theoretic* approach — contribution 1
  - exploiting critical pair analysis
    - in the hope of (partial) automation
  - by *rewriting-theoretic* modelling of both refinement $\Rightarrow$ and evaluation $\rightarrow$ — contribution 2

Modelling evaluation → and refinement ⇒

- being aware of different rewriting-theoretic properties!

evaluation	refinement
run-time rewriting	compile-time rewriting
beta-reduction $E[(\lambda x\,.\,t)\ v] \to E[t\{v/x\}]$	beta-law $C[(\lambda x\,.\,t)\ v] \Rightarrow C[t\{v/x\}]$ eta-law $C[\lambda x\,.\,v\ x] \Rightarrow C[v]$
rule-based rewriting	rule-based rewriting
restricted by *evaluation contexts* [Felleisen '88]	unrestricted

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒

- being aware of different rewriting-theoretic properties!

evaluation	refinement
run-time rewriting	compile-time rewriting
beta-reduction $E[(\lambda x\,.\,t)\ v] \to E[t\{v/x\}]$	beta-law $C[(\lambda x\,.\,t)\ v] \Rightarrow C[t\{v/x\}]$ eta-law $C[\lambda x\,.\,v\ x] \Rightarrow C[v]$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x\,.\,t \mid t @ t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \square \mid E @ t \mid v @ E$ $\mathscr{E} = \{(\lambda x\,.\,t) @ v \to t\{v/x\}\}$	$C \in Ctx ::= \square \mid \lambda x\,.\,C \mid E @ t \mid t @ E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t) @ v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v @ x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x.t \mid t@t$ $v \in Val ::= \lambda x.t$ $E \in Ectx ::= \square \mid E@t \mid v@E$ $\mathscr{E} = \{(\lambda x.t)@v \to t\{v/x\}\}$ $\Sigma = \{\lambda, @\}$	$C \in Ctx ::= \square \mid \lambda x.C \mid E@t \mid t@E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x.t)@v \Rightarrow t\{v/x\}, \\ \lambda x.v@x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$
new kind of rewriting	general, standard, rewriting

terms

Modelling evaluation $\to$ and refinement $\Rightarrow$: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus	
$t ::= x \mid \lambda x\,.\,t \mid t @ t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \Box \mid E @ t \mid v @ E$ $\mathscr{E} = \{(\lambda x\,.\,t) @ v \to t\{v/x\}\}$	$C \in Ctx ::= \Box \mid \lambda x\,.\,C \mid E @ t \mid t @ E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t) @ v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v @ x \Rightarrow v \end{array} \right\}$
$Val \subseteq \mathrm{NF}(\to_{\mathscr{E}})$	
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

values

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x.t \mid t@t$ $v \in \mathit{Val} ::= \lambda x.t$ $E \in \mathit{Ectx} ::= \square \mid E@t \mid v@E$ $\mathscr{E} = \{(\lambda x.t)@v \to t\{v/x\}\}$ $\lambda x.x \in \mathit{Val}, \quad x@y \notin \mathit{Val}$	$C \in \mathit{Ctx} ::= \square \mid \lambda x.C \mid E@t \mid t@E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x.t)@v \Rightarrow t\{v/x\}, \\ \lambda x.v@x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in \mathit{Ectx}}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in \mathit{Ctx}}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

values

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x\,.\,t \mid t@t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \square \mid E@t \mid v@E$ $\mathscr{E} = \{(\lambda x\,.\,t)@v \to t\{v/x\}\}$ $\square@x \in Ectx, \quad (\lambda x\,.\,x)@\square \in Ectx$	$C \in Ctx ::= \square \mid \lambda x\,.\,C \mid E@t \mid t@E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t)@v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v@x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

evaluation contexts

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x.t \mid t@t$ $v \in Val ::= \lambda x.t$ $E \in Ectx ::= \square \mid E@t \mid v@E$ $\mathscr{E} = \{(\lambda x.t)@v \to t\{v/x\}\}$ $\square@x \in Ectx, \quad ((\lambda x.x)@y)@\square \notin Ectx$	$C \in Ctx ::= \square \mid \lambda x.C \mid E@t \mid t@E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x.t)@v \Rightarrow t\{v/x\}, \\ \lambda x.v@x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

evaluation contexts

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x.t \mid t@t$ $v \in Val ::= \lambda x.t$ $E \in Ectx ::= \square \mid E@t \mid v@E$ $\mathscr{E} = \{(\lambda x.t)@v \to t\{v/x\}\}$ $(\lambda x.x)@y \to_{\mathscr{E}} y$	$C \in Ctx ::= \square \mid \lambda x.C \mid E@t \mid t@E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x.t)@v \Rightarrow t\{v/x\}, \\ \lambda x.v@x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

evaluation rules

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x\,.\,t \mid t @ t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \square \mid E @ t \mid v @ E$ $\mathscr{E} = \{(\lambda x\,.\,t) @ v \to t\{v/x\}\}$ $(\lambda x\,.\,x) @ (y @ z) \not\to_{\mathscr{E}}$	$C \in Ctx ::= \square \mid \lambda x\,.\,C \mid E @ t \mid t @ E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t) @ v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v @ x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$
new kind of rewriting	general, standard, rewriting

evaluation rules

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x\,.\,t \mid t @ t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \square \mid E @ t \mid v @ E$ $\mathscr{E} = \{(\lambda x\,.\,t) @ v \to t\{v/x\}\}$	$C \in Ctx ::= \square \mid \lambda x\,.\,C \mid E @ t \mid t @ E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t) @ v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v @ x \Rightarrow v \end{array} \right\}$ $\lambda x\,.\,\square \in Ctx, \quad ((\lambda x\,.\,x) @ y) @ \square \in Ctx$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

(arbitrary) contexts

new kind of rewriting

general, standard, rewriting

Modelling evaluation → and refinement ⇒: an example

- being aware of different rewriting-theoretic properties!

evaluation	refinement
the left-to-right call-by-value λ-calculus $t ::= x \mid \lambda x\,.\,t \mid t @ t$ $v \in Val ::= \lambda x\,.\,t$ $E \in Ectx ::= \square \mid E @ t \mid v @ E$ $\mathscr{E} = \{(\lambda x\,.\,t) @ v \to t\{v/x\}\}$	$C \in Ctx ::= \square \mid \lambda x\,.\,C \mid E @ t \mid t @ E$ $\mathscr{R} = \left\{ \begin{array}{l} (\lambda x\,.\,t) @ v \Rightarrow t\{v/x\}, \\ \lambda x\,.\,v @ x \Rightarrow v \end{array} \right\}$
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

refinement rules

new kind of rewriting

general, standard, rewriting

Term Evaluation and Refinement Systems (TERS)

Def. A TERS $(\Sigma, \mathscr{E}, \mathscr{R}, Ectx, SClass)$ is given by:

- a signature Σ
- a set $\mathscr{E}$ of evaluation rules $l \to r$
- a set $\mathscr{R}$ of refinement rules $l \Rightarrow r$
- a set $Ectx \subseteq Ctx$ of evaluation contexts
- a set $SClass$ of syntax classes including $Val \subseteq \mathrm{NF}(\to_{\mathscr{E}})$

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

Term Evaluation and Refinement Systems (TERS)

Ex. A TERS $\mathbf{CBV}\lambda$ for the left-to-right call-by-value λ-calculus

- a signature $\Sigma = \{\lambda, @\}$
- a set $\mathscr{E} = \{(\lambda x\,.\,t)\,@\,v \rightarrow t\{v/x\}\}$ of an evaluation rule
- a set $\mathscr{R} = \{(\lambda x\,.\,t)\,@\,v \Rightarrow t\{v/x\},\ \ \lambda x\,.\,v\,@\,x \Rightarrow v\}$ of refinement rules
- evaluation contexts $E \in Ectx ::= \square \mid E\,@\,t \mid v\,@\,E$
- a set $SClass$ of syntax classes including $Val = \{\lambda x\,.\,t\} \subseteq \mathrm{NF}(\rightarrow_{\mathscr{E}})$

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

Term Evaluation and Refinement Systems (TERS)

<u>Ex.</u> A TERS $\mathbf{Hndl}$ for a λ-calculus with effect handlers [Pretnar '15]

Syntax class *Sclass*

functions $F ::= x \mid \mathtt{fun}(x.P)$

values $V ::= \mathtt{true} \mid \mathtt{false} \mid F \mid H$

handlers $H ::= \mathtt{handler}_1(x.P, x.k.P_1) \mid \mathtt{handler}_0(x.P)$

computations $P, P_1, P_2 ::= \mathtt{return}(V) \mid \mathtt{op}(V, y.P) \mid \mathtt{do}(P_1, x.P_2)$
$\mid \mathtt{if}(V, P_1, P_2) \mid F\ V \mid \mathtt{with\_handle}(H, P)$

Evaluation contexts *Ectx* $E ::= \square \mid \mathtt{do}(E, x.P) \mid \mathtt{with\_handle}(H, E)$

Evaluation rules $\mathcal{E}$ where $i \in [2]$

$\mathtt{do}(\mathtt{return}(V), x.P[x]) \to P[V]$ (1)

$\mathtt{do}(\mathtt{op}_i(V, y.P_1[y]), x.P_2[x]) \to \mathtt{op}_i(V, y.\mathtt{do}(P_1[y], x.P_2[x]))$ (2)

$\mathtt{if}(\mathtt{true}, P_1, P_2) \to P_1$ (3)

$\mathtt{if}(\mathtt{false}, P_1, P_2) \to P_2$ (4)

$\mathtt{fun}(x.P[x])\ V \to P[V]$ (5)

In the following three rules, $h_1 \equiv \mathtt{handler}_1(x.P[x], x.k.P_1[x, k])$.

$\mathtt{with\_handle}(h_1, \mathtt{return}(V)) \to P[V]$ (6)

$\mathtt{with\_handle}(h_1, \mathtt{op}_1(V, y.P'[y])) \to P_1[V, \mathtt{fun}(y.P'[y])]$ (7)

$\mathtt{with\_handle}(h_1, \mathtt{op}_1(V, y.P'[y])) \to P_1[V, \mathtt{fun}(y.\mathtt{with\_handle}(h_1, P'[y]))]$ (7')

$\mathtt{with\_handle}(h_1, \mathtt{op}_2(V, y.P'[y])) \to \mathtt{op}_2(V, y.\mathtt{with\_handle}(h_1, P'[y]))$ (8)

In the following two rules, $h_0 \equiv \mathtt{handler}_0(x.P[x])$.

$\mathtt{with\_handle}(h_0, \mathtt{return}(V)) \to P[V]$ (9)

$\mathtt{with\_handle}(h_0, \mathtt{op}_i(V, y.P'[y])) \to \mathtt{op}_i(V, y.\mathtt{with\_handle}(h_0, P'[y]))$ (10)

Refinement rules $\mathcal{R}$

$\mathtt{do}(P, x.\mathtt{return}(x)) \Rightarrow P$ (r3)

$\mathtt{do}(\mathtt{do}(P_1, x_1.P_2[x_1]), x_2.P_3[x_2]) \Rightarrow \mathtt{do}(P_1, x_1.\mathtt{do}(P_2[x_1], x_2.P_3[x_2]))$ (r4)

$\mathtt{if}(V, P[\mathtt{true}], P[\mathtt{false}]) \Rightarrow P[V]$ (r7)

$\mathtt{fun}(x.F\ x) \Rightarrow F$ (r9)

$\mathtt{with\_handle}(\mathtt{handler}_0(x.P[x]), P') \Rightarrow \mathtt{do}(P', x.P[x])$ (r13)

Our goal

- develop a proof methodology
  - to validate refinement $\Rightarrow$ wrt. evaluation $\rightarrow$
  - to establish refinement $\Rightarrow$ implies contextual improvement $\preceq$

- by a *rewriting-theoretic* approach
  contribution 1
  - exploiting critical pair analysis
  - in the hope of (partial) automation
  - by *rewriting-theoretic* modelling of both refinement $\Rightarrow$ and evaluation $\rightarrow$
    as Term Evaluation and Refinement Systems (TERS)
    contribution 2

Our goal

- develop a proof methodology
    - to validate refinement $\Rightarrow$ wrt. evaluation $\rightarrow$
    - to establish refinement $\Rightarrow$ implies contextual improvement $\preceq$
- by a *rewriting-theoretic* approach

contribution 1

- exploiting critical pair analysis
- in the hope of (partial) automation
- by *rewriting-theoretic* modelling of both refinement $\Rightarrow$ and evaluation $\rightarrow$ as Term Evaluation and Refinement Systems (TERS)

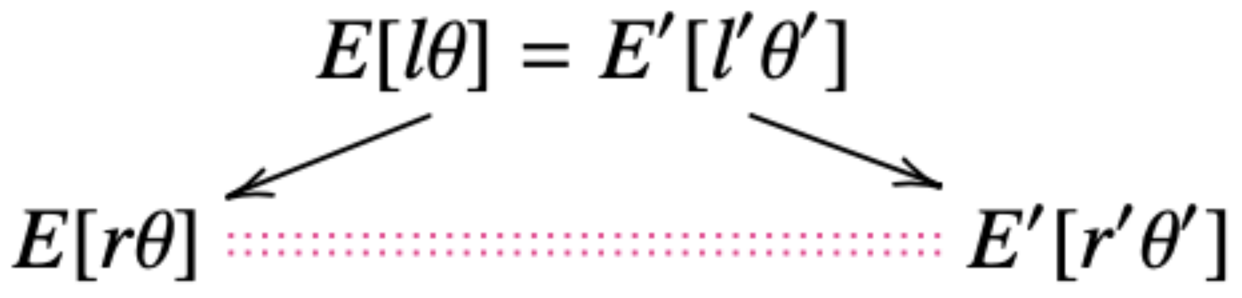
contribution 2

Sufficient conditions for contextual improvement $\preceq$

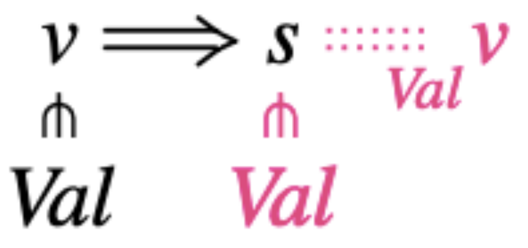
Thm. (sufficient conditions for improvement)

If a TERS is

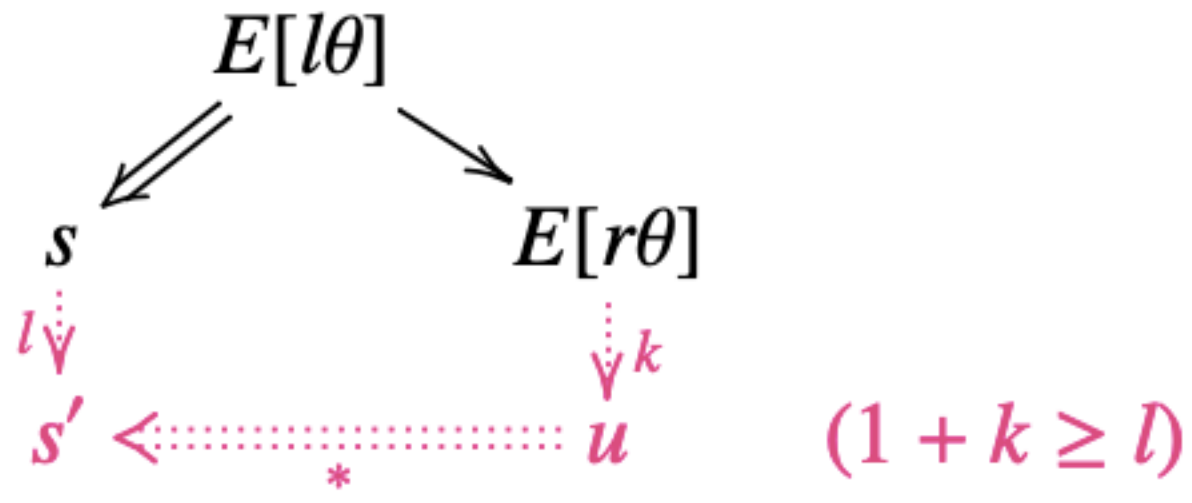
- *deterministic*



- *value-invariant*



- *locally coherent*,



akin to *local coherence* for equational rewriting [Huet '80] [Aoto & Toyama '12]

then refinement $\mathscr{R}$ is *contextual improvement* w.r.t. evaluation $\mathscr{E}$

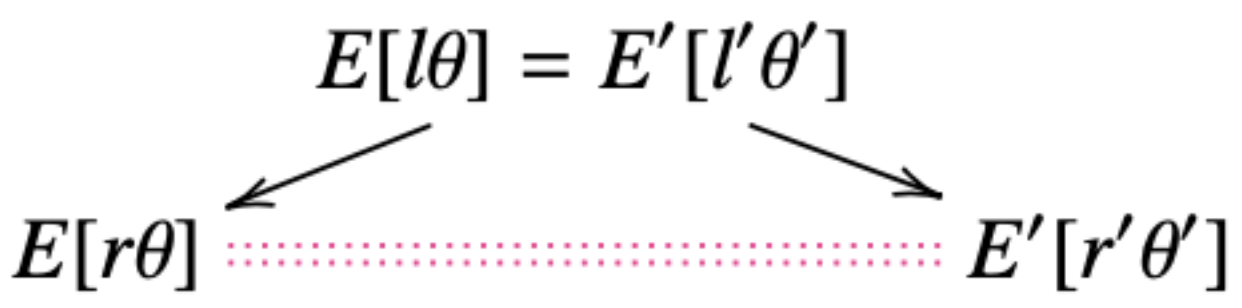
(i.e. $t \preceq u$ for each $t \Rightarrow_{\mathscr{R}} u$)

Sufficient conditions for contextual improvement $\precsim$

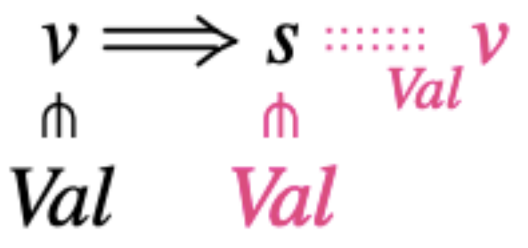
Thm. (sufficient conditions for improvement)

If a TERS is

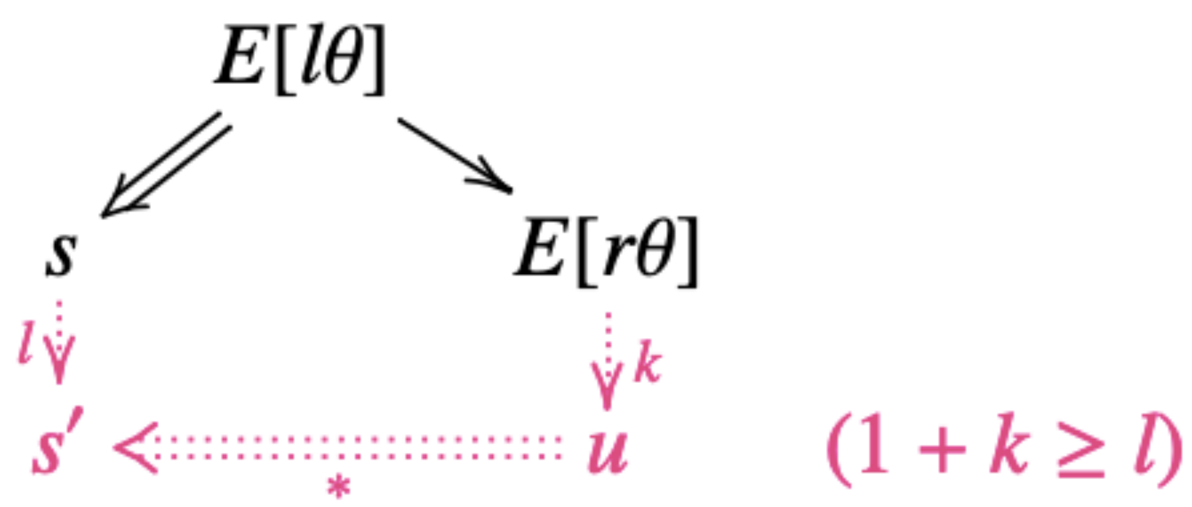
- *deterministic*



- *value-invariant*



- *locally coherent*,



feasible
pen-and-paper
proof

tedious, error-prone,
case analysis

where *critical pair analysis* can help!

then refinement $\mathscr{R}$ is *contextual improvement* w.r.t. evaluation $\mathscr{E}$

(i.e. $t \precsim u$ for each $t \Rightarrow_{\mathscr{R}} u$)

Sufficient conditions for contextual improvement $\preceq$

Thm. (critical pair theorem)

A *well-behaved* TERS

- refinement respecting evaluation contexts

$$\begin{array}{ccc} E & \Longrightarrow & C \\ \pitchfork & & \pitchfork \\ \mathit{Ectx} & & \mathit{Ectx} \end{array}$$

- linearity for $(l \Rightarrow r) \in \mathscr{R}$
- left-linearity for $(l \rightarrow r) \in \mathscr{E}$
- lhs l being Miller's HO pattern for $(l \multimap r) \in \mathscr{R} \cup \mathscr{E}$
- …

is *locally coherent* iff every critical pair is *joinable*.

feasible
pen-and-paper
proof

automated
enumeration &
joinability check

Term Evaluation and Refinement Systems (TERS)

Ex. A TERS $\mathbf{CBV}\lambda$ for the left-to-right call-by-value λ-calculus

- a signature $\Sigma = \{\lambda, @\}$
- a set $\mathscr{E} = \{(\lambda x\,.\,t)\,@\,v \to t\{v/x\}\}$ of an evaluation rule
- a set $\mathscr{R} = \{(\lambda x\,.\,t)\,@\,v \Rightarrow t\{v/x\},\ \ \lambda x\,.\,v\,@\,x \Rightarrow v\}$ of refinement rules
- evaluation contexts $E \in Ectx ::= \square \mid E\,@\,t \mid v\,@\,E$
- a set $SClass$ of syntax classes including $Val = \{\lambda x\,.\,t\} \subseteq \mathrm{NF}(\to_{\mathscr{E}})$

Prop. For the TERS $\mathbf{CBV}\lambda$, refinement $\mathscr{R}$ is contextual improvement $\preceq$.

- ✓ determinism
- ✓ value-invariance
- ✓ local coherence
  - ✓ well-behavedness
  - ✓ 2 critical pairs joinable

automatically
enumerated & checked

Term Evaluation and Refinement Systems (TERS)

Ex. A TERS **Hndl** for a λ-calculus with effect handlers [Pretnar ’15]

Syntax class *Sclass*

functions $F ::= x \mid \mathtt{fun}(x.P)$

values $V ::= \mathtt{true} \mid \mathtt{false} \mid F \mid H$

handlers $H ::= \mathtt{handler}_1(x.P, x.k.P_1) \mid \mathtt{handler}_0(x.P)$

computations $P, P_1, P_2 ::= \mathtt{return}(V) \mid \mathtt{op}(V, y.P) \mid \mathtt{do}(P_1, x.P_2)$
$\mid \mathtt{if}(V, P_1, P_2) \mid F\ V \mid \mathtt{with\_handle}(H, P)$

Evaluation contexts *Ectx* $E ::= \square \mid \mathtt{do}(E, x.P) \mid \mathtt{with\_handle}(H, E)$

Evaluation rules $\mathcal{E}$ where $i \in [2]$

$\mathtt{do}(\mathtt{return}(V), x.P[x]) \to P[V]$ (1)

$\mathtt{do}(\mathtt{op}_i(V, y.P_1[y]), x.P_2[x]) \to \mathtt{op}_i(V, y.\mathtt{do}(P_1[y], x.P_2[x]))$ (2)

$\mathtt{if}(\mathtt{true}, P_1, P_2) \to P_1$ (3)

$\mathtt{if}(\mathtt{false}, P_1, P_2) \to P_2$ (4)

$\mathtt{fun}(x.P[x])\ V \to P[V]$ (5)

In the following three rules, $h_1 \equiv \mathtt{handler}_1(x.P[x], x.k.P_1[x,k])$.

$\mathtt{with\_handle}(h_1, \mathtt{return}(V)) \to P[V]$ (6)

$\mathtt{with\_handle}(h_1, \mathtt{op}_1(V, y.P'[y])) \to P_1[V, \mathtt{fun}(y.P'[y])]$ (7)

$\mathtt{with\_handle}(h_1, \mathtt{op}_1(V, y.P'[y])) \to P_1[V, \mathtt{fun}(y.\mathtt{with\_handle}(h_1, P'[y]))]$ (7’)

$\mathtt{with\_handle}(h_1, \mathtt{op}_2(V, y.P'[y])) \to \mathtt{op}_2(V, y.\mathtt{with\_handle}(h_1, P'[y]))$ (8)

In the following two rules, $h_0 \equiv \mathtt{handler}_0(x.P[x])$.

$\mathtt{with\_handle}(h_0, \mathtt{return}(V)) \to P[V]$ (9)

$\mathtt{with\_handle}(h_0, \mathtt{op}_i(V, y.P'[y])) \to \mathtt{op}_i(V, y.\mathtt{with\_handle}(h_0, P'[y]))$ (10)

Refinement rules $\mathcal{R}$

$\mathtt{do}(P, x.\mathtt{return}(x)) \Rightarrow P$ (r3)

$\mathtt{do}(\mathtt{do}(P_1, x_1.P_2[x_1]), x_2.P_3[x_2]) \Rightarrow \mathtt{do}(P_1, x_1.\mathtt{do}(P_2[x_1], x_2.P_3[x_2]))$ (r4)

$\mathtt{if}(V, P[\mathtt{true}], P[\mathtt{false}]) \Rightarrow P[V]$ (r7)

$\mathtt{fun}(x.F\ x) \Rightarrow F$ (r9)

$\mathtt{with\_handle}(\mathtt{handler}_0(x.P[x]), P') \Rightarrow \mathtt{do}(P', x.P[x])$ (r13)

Prop. For the TERS **Hndl**,

refinement $\mathscr{R}$ is

contextual improvement $\preceq$.

- ✓ determinism
- ✓ value-invariance
- ✓ local coherence
  - ✓ well-behavedness
  - ✓ 10 critical pairs joinable

automatically enumerated & checked

Overview

evaluation	refinement
run-time rewriting	compile-time rewriting
specification of operational semantics	model of optimisation to be **validated** wrt. evaluation
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

formalise **Term Evaluation and Refinement Systems (TERS)**

$$(\Sigma, \mathscr{E}, \mathscr{R}, Ectx, SClass)$$

Overview

evaluation	refinement
run-time rewriting	compile-time rewriting
specification of operational semantics	model of optimisation to be **validated** wrt. evaluation
$\dfrac{(l \to r) \in \mathcal{E} \quad E \in Ectx}{E[l\theta] \to_{\mathcal{E}} E[r\theta]}$	$\dfrac{(l \Rightarrow r) \in \mathcal{R} \quad C \in Ctx}{C[l\theta] \Rightarrow_{\mathcal{R}} C[r\theta]}$

- formally, *contextual improvement*
- develop a proof methodology centred around critical pair analysis