Symmetries and Programs

Carlos C. Martínez

Department of Mathematics and Computer Science,

Wesleyan University

cmartinez@wesleyan.edu

March, 2005

Example 1:

There are four Caml functions abstracted from $2*x+3*y$:

#Let f1 $=$ $\boxed{\text{fun}(x,y)}$ $\rightarrow 2*x+3*y;;$

f1: $\mathrm{int}*\mathrm{int} \rightarrow \mathrm{int}$ = $\langle$**fun**$\rangle$

#Let f2 $=$ $\boxed{\text{fun}(y,x)}$ $\rightarrow 2*x+3*y;;$

f2: $\mathrm{int}*\mathrm{int} \rightarrow \mathrm{int}$ = $\langle$**fun**$\rangle$

#Let f3 $=$ $\boxed{\text{fun}xy}$ $\rightarrow 2*x+3*y;;$

f3: $\mathrm{int} \rightarrow \mathrm{int} \rightarrow \mathrm{int}$ = $\langle$**fun**$\rangle$

#Let f4 $=$ $\boxed{\text{fun}yx}$ $\rightarrow 2*x+3*y;;$

f4: $\mathrm{int} \rightarrow \mathrm{int} \rightarrow \mathrm{int}$ = $\langle$**fun**$\rangle$

Example 1:

#Let curry f $xy = \text{fun}(x, y)$;;

curry: $((A * B) \to C) \to (A \to (B \to C))$ = $\langle$**fun**$\rangle$

#Let uncurry f $(x, y) = \text{fun}xy$;;

uncurry: $(A \to (B \to C)) \to ((A * B) \to C)$ = $\langle$**fun**$\rangle$

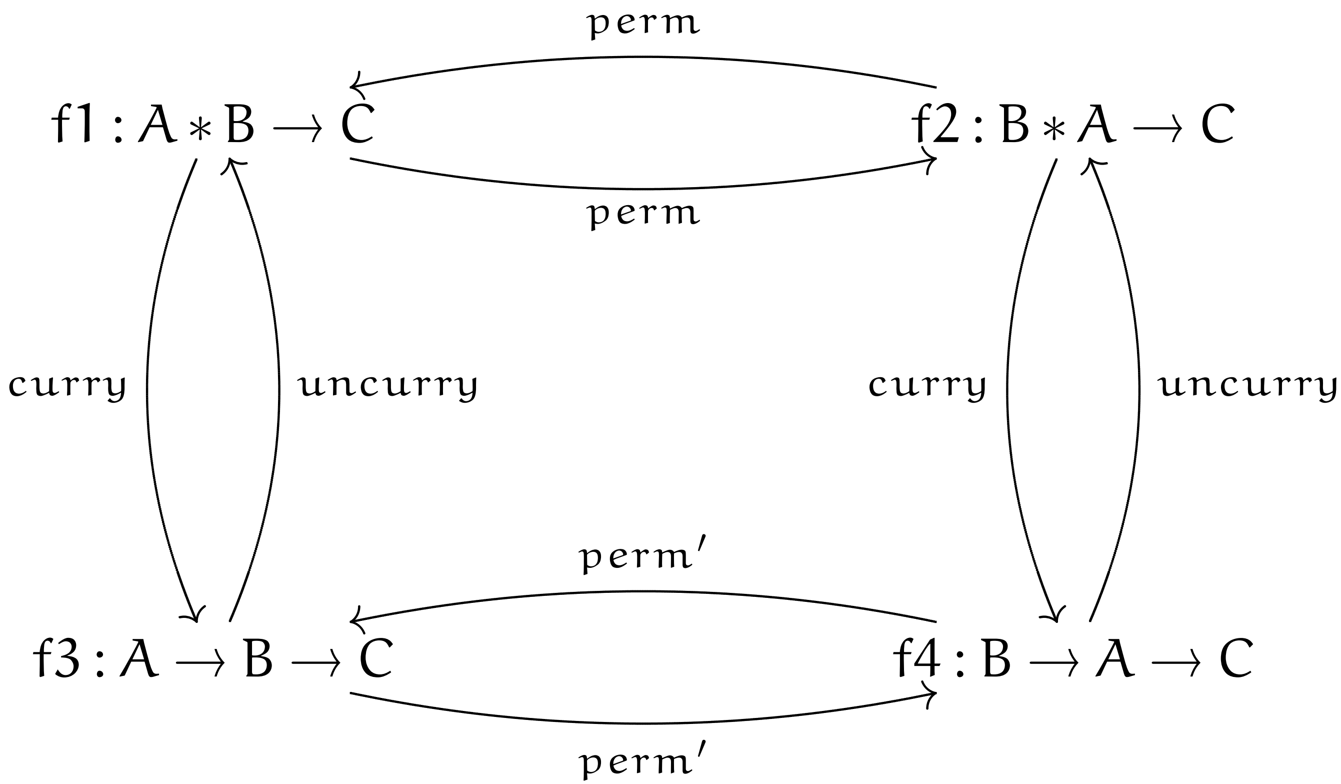
#Let perm f $xy = \text{fun}yx$;;

perm : $(A \to (B \to C)) \to (B \to (A \to C))$ = $\langle$**fun**$\rangle$

#Let perm' f $(x, y) = \text{fun}(y, x)$;;

perm' : $((A * B) \to C)) \to ((B * A) \to C))$= $\langle$**fun**$\rangle$

Example 1:


perm
f1 : A ∗ B → C
f2 : B ∗ A → C
perm
curry
uncurry
curry
uncurry
perm′
f3 : A → B → C
f4 : B → A → C
perm′

Definable isomorphisms:

Definition ***(Type Isomorphisms)*** *Two type* A *and* B *are definably isomorphic* $(A \cong_d B)$ *if and only if there exists programs* $M : A \to B$ *and* $N : B \to A$ *such that* $M \circ N = Id$ *and* $N \circ M = Id$*, the identities of type* A *and* B*.*

Definition ***(Program Isomorphisms)*** *Two programs* $P : A$ *and* $Q : B$ *are definably isomorphic* $(P \cong_d Q)$ *if and only if* $(A \cong_d B)$ *by an invertible program* $F : A \to B$ *such that* $FP = Q$*.*

Example 2:

#**Let rec** fact $n =$ **if** $(n = 0)$ **then** 1
else $n*$ fact$(n-1)$;;
fact: $\mathrm{int} \to \mathrm{int}$ = ⟨**fun**⟩

#**Let rec** rev $l =$ **if** $(l = [])$ **then** []
else $\mathrm{append}'([\mathrm{head}(l)],$ rev$(\mathrm{tail}(l)))$;;
rev: $\mathrm{int}^* \to \mathrm{int}^*$ = ⟨**fun**⟩

By "abstracting", we can see that they are instances of a "iterator"

#**Let rec** iter $fx =$ **if** (ax) **then** (bx) **else** $f((cx),$ iter $f(dx))$;;
iter: $\underline{(A \times A \to A)} \to A \to (A \to A)$= ⟨**fun**⟩

Where $a : A \to$Bool, b, c and $d : A \to A$

Remark: append' is an isomorphic version of the usual append

Example 2:

Thus:

σ_1(iter $*n$) = fact n $\qquad$ σ_2(iter $\mathtt{append}'\ l$) = rev l

$$\sigma_1 \begin{cases} a \leftarrow \lambda x.(x == 0); \\ b \leftarrow \lambda x.1; \\ c \leftarrow \lambda x.x; \\ d \leftarrow \lambda x.(x - 1); \end{cases} \qquad \sigma_2 \begin{cases} a \leftarrow \lambda x.(x == []); \\ b \leftarrow \lambda x.0; \\ c \leftarrow \lambda x.[(\mathtt{head}\ x)]; \\ d \leftarrow \lambda x.(\mathtt{tail}\ x); \end{cases}$$

Matching

Definition ***(Matching terms)*** *Let* M *be term, and* N *be a close term, we say that* M *is matchable to* N *if and only if there exists a substitution* σ *such that* $\sigma(M) = N$

Definition ***(Matching programs)*** *Let* P *and* Q *be programs, we say that* P *is matchable to* Q *if and only if there exists inputs* $in_1, ..., in_n$ *such that* $P in_1 ... in_n = Q$

Example 2:

Optimization: Consider the following "Loop" function

#**Let rec** loop pfx = **if** (px) **then** x **else** loop $pf(fx)$;;

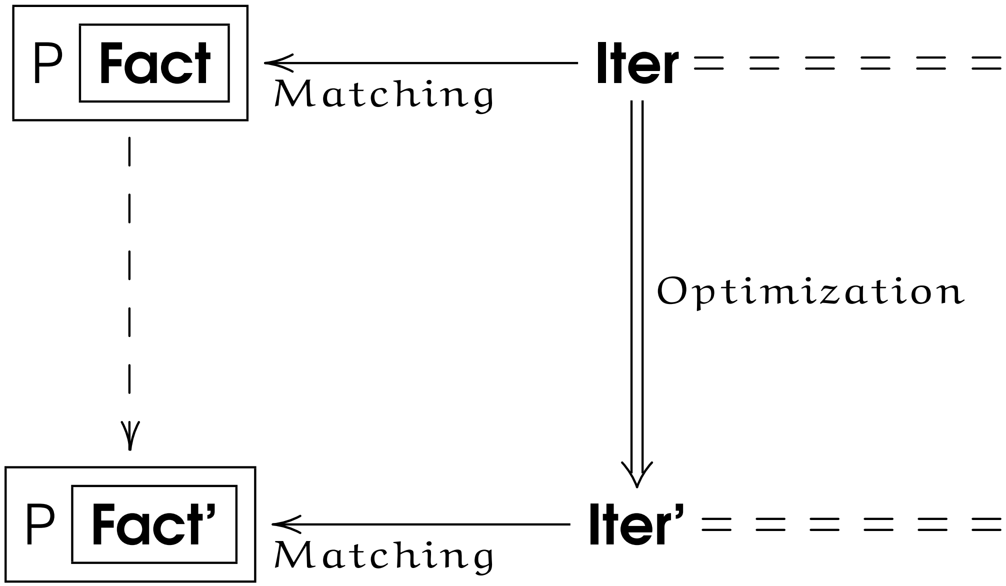
loop: $(A \rightarrow \mathrm{Bool}) \rightarrow (A \rightarrow A) \rightarrow (A \rightarrow A)$= $\langle$**fun**$\rangle$

We gain a more efficient stack usage, and...

#**Let** fact′ $n =$
snd(loop (fun $(m, y) \rightarrow m = 0$)
(fun $(m, y) \rightarrow (m - 1, m * y)$)
$(n, 1)$);;
fact′: $int \rightarrow int$ = $\langle$**fun**$\rangle$

#**Let** rev′ $l =$
snd(loop (fun $(l, y) \rightarrow l = []$)
(fun $(l, y) \rightarrow (tail(l), append'([head(l)], y))$)
$(l, [])$);;
rev′: $int^* \rightarrow int^*$ = $\langle$**fun**$\rangle$

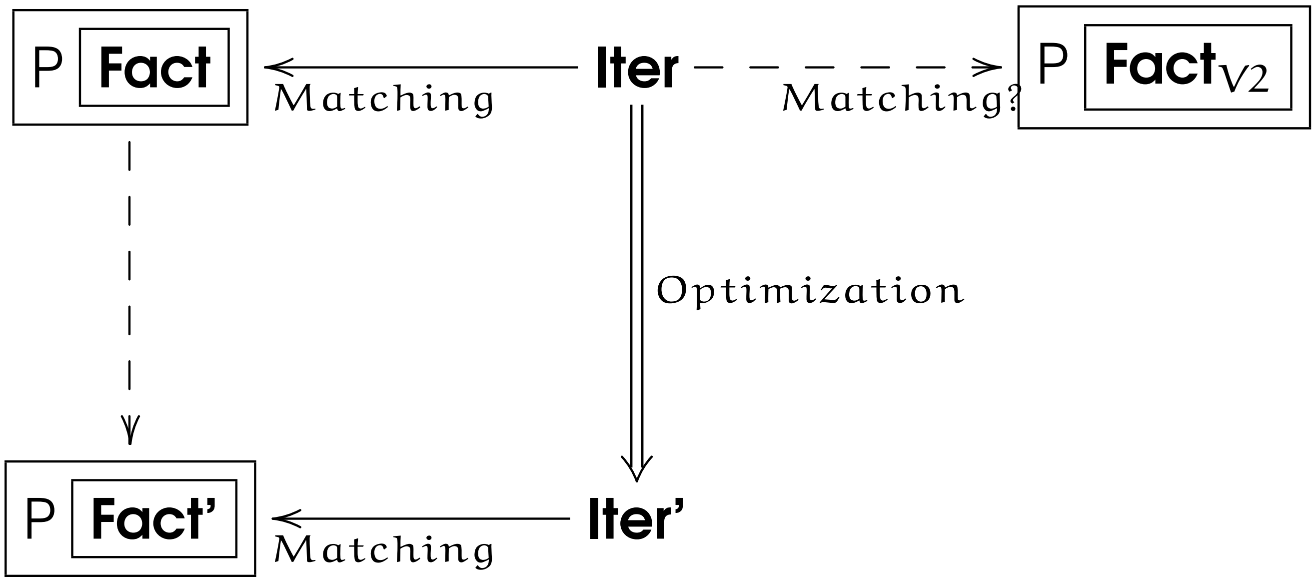
Example 2:



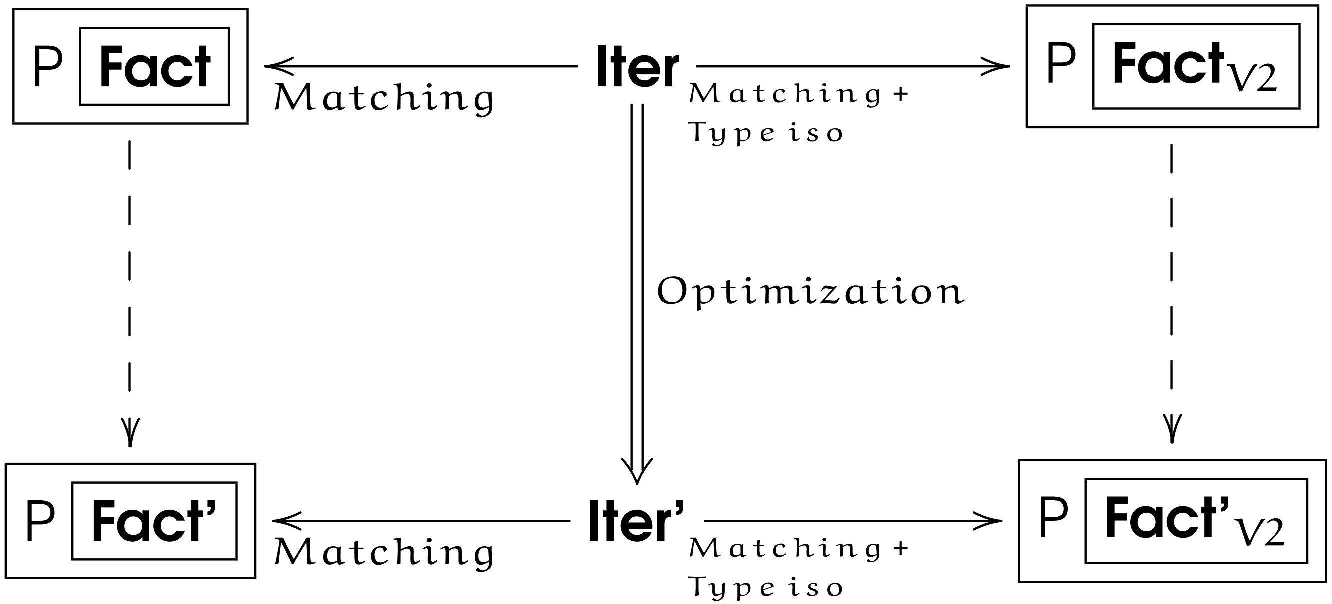

```
#Let rec iter f x = if ( a x ) then ( b x )
        else f ( ( c x ) , iter f ( d x ) ) ;;
```

```
#Let iter' f x = snd( loop
(fun ( x , y ) → ( a x ))
  (fun ( x , y ) → ( ( d x ) , f ( ( c x ) , y ) ))
  ( x , ( b x ) ) );;
```

Example 3:


P Fact
Matching
Iter
Matching?
P $\mathbf{Fact}_{V2}$
Optimization
P Fact'
Matching
Iter'

Example 3:


P Fact
Matching
Iter
Matching +
Type iso
P Fact$_{V2}$
Optimization
P Fact'
Matching
Iter'
Matching +
Type iso
P Fact'$_{V2}$

Matching modulo Type isomorphisms

Definition ***(Matching terms)*** *Let* M *be term, and* N *be a close term, we say that* M *is matchable to* N *if and only if there exists a substitution* σ *such that* $\boxed{\sigma(M) \cong_d N}$

Definition ***(Matching programs)*** *Let* P *and* Q *be programs, we say that* P *is matchable to* Q *if and only if there exists inputs* $in_1, ..., in_n$ *such that* $\boxed{P in_1 ... in_n \cong_d Q}$

References:

Bruce K., Di Cosmo R., Longo G.
Provable Isomorphism of Types, *Mathematical Structures in Computer Science*, **2**, 231 - 247, 1991.

Dezani-Ciancaglini M.
Characterization of Normal Forms Possessing Inverse in the $\lambda_{\beta\eta}$-Calculus, *Theoretical Computer Science*, **2**, 323 - 337, 1976.

Di Cosmo R.
Isomorphism of Types: from λ-calculus to information retrieval and language design, *Birkhäuser*, 1995.

References:

Rittri M.
Retrieving Library Functions by Unifying Types Modulo Linear Type Isomorphism, *Theoretical Informatics and Applications*, **27**, 71 -89, 1993.

Soloviev S.
The Category of Finite Sets and Cartesian Closed Categories, *Journal of Soviet Mathematics*, **22(3)**, 1387 - 1400, 1983.

Zibin Y., Gil J., Considine J.
Efficient Algorithm for Isomorphism of Simple Types, *Proceedings POPL'03 in ACM SIGPLAN*, **38**, 160 - 171, 2003.

References:

Huet G. and Lang B.,
Proving and Applying Program Transformations Expressed with Second Order Patterns, *Acta Informatica*, 11:31–55, 1978.