Efficient normalization by evaluation

Mathieu Boespflug

École Polytechnique

15 August 2009

The conversion test

$$\frac{\Gamma \vdash A : s \qquad \Gamma \vdash B : s \qquad \Gamma \vdash t : A}{\Gamma \vdash t : B} \; A \equiv_\beta B$$

The conversion test

$$\frac{\Gamma \vdash A : s \qquad \Gamma \vdash B : s \qquad \Gamma \vdash t : A}{\Gamma \vdash t : B} \; A \equiv_\beta B$$

A simple algorithm:

1. Reduce A and B to their canonical forms.
2. Compare canonical forms.

fast

fast cheap

fast cheap general

- Plenty of existing (fast) reduction devices.

- Plenty of existing (fast) reduction devices.
- Solution: reuse them!

- Plenty of existing (fast) reduction devices.
- Solution: reuse them!
- Advantage: separation of concerns.

```
Eval vm_compute in fib 30.
```

```
Eval vm_compute in fib 30.
```

B. Grégoire and X. Leroy, "A compiled implementation of strong reduction," Proceedings ICFP'02, 2002.

Extended Terms:

$$b ::= x \mid \lambda x.\ b \mid b_1\ b_2 \mid [\tilde{x}\ v_1\ \dots\ v_n]$$
$$v ::= \lambda x.\ b \mid [\tilde{x}\ v_1\ \dots\ v_n]$$

Symbolic weak reduction:

$$(\lambda x.\ b)\ v \to b[x := v]$$
$$[x\ v_1\ \dots\ v_n]\ v \to [x\ v_1\ \dots\ v_n\ v]$$
$$\Gamma_v(a) \to \Gamma_v(a') \quad \text{if } a \to a'$$

with $\Gamma_v ::= []v \mid b[]$.

$$\mathcal{N}(b) = \mathcal{R}(\mathcal{V}(b)) \tag{1}$$

$$\mathcal{R}(\lambda x.\ b) = \lambda y.\ \mathcal{N}((\lambda x.\ b)\ [\tilde{y}]) \tag{2}$$

$$\mathcal{R}([\tilde{x}\ v_1 \ldots\ v_n]) = x\ \mathcal{R}(v_1)\ \ldots\ \mathcal{R}(v_n) \tag{3}$$

Grégoire and Leroy propose a virtual machine to implement symbolic weak reduction and normalization.

Grégoire and Leroy propose a virtual machine to implement symbolic weak reduction and normalization.

The name of the game: avoid untagging during applications.

Grégoire and Leroy propose a virtual machine to implement symbolic weak reduction and normalization.

The name of the game: avoid untagging during applications.

Semi-cheap: Requires modification of the runtime environment.

Grégoire and Leroy propose a virtual machine to implement symbolic weak reduction and normalization.

The name of the game: avoid untagging during applications.

Semi-cheap: Requires modification of the runtime environment.

Objective: be cheap, not just semi-cheap.

Normalization by Evaluation

Optimizations

- Uncurrying
- Embedding pattern matching

Benchmarks

Conclusion

$$\begin{aligned}
&\textbf{data}\ \mathit{Code} = \mathit{Con}\ \mathit{String} \\
&\quad \mid \mathit{Lam}\ (\mathit{Code} \to \mathit{Code}) \\
&\quad \mid \mathit{Neu}\ \mathit{Code}\ \mathit{Code}
\end{aligned}$$

Interpretation

$$
\begin{aligned}
[\![\underline{x}]\!]\ n &= \hat{x} && \text{if } x < n \\
[\![\underline{x}]\!]\ n &= \mathit{Con}\ \underline{x} && \text{otherwise} \\
[\![\underline{\lambda.\ t}]\!]\ n &= \mathit{Abs}\ (\lambda \hat{n} \to [\![t]\!]\ (n+1)) \\
[\![\underline{t_1\ t_2}]\!]\ n &= \mathit{app}\ ([\![t_1]\!]\ n)\ ([\![t_2]\!]\ n)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{app}\ (\mathit{Abs}\ t_1)\ t_2 = t_1\ t_2 \\
&\mathit{app}\ t_1\ t_2 = \mathit{Neu}\ t_1\ t_2
\end{aligned}
$$

Interpretation

$$
\begin{aligned}
[\![\underline{x}]\!]\ n &= \hat{x} && \text{if } x < n \\
[\![\underline{x}]\!]\ n &= \mathit{Con}\ \underline{x} && \text{otherwise} \\
[\![\underline{\lambda.\ t}]\!]\ n &= \mathit{Abs}\ (\lambda \hat{n} \to [\![t]\!]\ (n+1)) \\
[\![\underline{t_1\ t_2}]\!]\ n &= \mathit{app}\ ([\![t_1]\!]\ n)\ ([\![t_2]\!]\ n)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{app}\ (\mathit{Abs}\ t_1)\ t_2 = t_1\ t_2 \\
&\mathit{app}\ (\mathit{Con}\ x)\ t_2 = \mathit{Neu}\ (\mathit{Con}\ x)\ t_2 \\
&\mathit{app}\ (\mathit{Neu}\ t_1\ t_1')\ t_2 = \mathit{Neu}\ (\mathit{Neu}\ t_1\ t_1')\ t_2
\end{aligned}
$$

Example

$$[\![\underline{(\lambda x.\ (\lambda y.\ y\ x))\ z}]\!]$$

Example

$$\frac{[\![\underline{(\lambda x.\ (\lambda y.\ y\ x))\ z}]\!]}{\mathit{app}\ (\mathit{Abs}\ (\lambda x \rightarrow \mathit{Abs}\ (\lambda y \rightarrow \mathit{app}\ y\ x)))\ (\mathit{Con}\ \texttt{"0"})}$$

Algorithm

$$\begin{aligned}
norm\ n\ (Con\ c) &= \underline{c} \\
norm\ n\ (Abs\ t) &= \underline{\lambda.}\ (norm\ (n+1)\ (t\ (Con\ \hat{n}))) \\
norm\ n\ (Neu\ t_1\ \ t_2) &= (norm\ n\ t_1)\ \underline{@}\ (norm\ n\ t_2)
\end{aligned}$$

Eval/Apply

$$
\begin{aligned}
[\![\underline{x}]\!]\ n &= \hat{x} && \text{if } x < n \\
[\![\underline{x}]\!]\ n &= Con\ \underline{x} && \text{otherwise} \\
[\![\underline{\lambda.\ \cdots \lambda.\ t}]\!]\ n &= Abs_m\ (\lambda \hat{n}\ \ldots\ \widehat{n+m} \to [\![t]\!]\ (n+m)) \\
[\![\underline{t_1\ \ldots\ t_m}]\!]\ n &= ap_m\ ([\![t_1]\!]\ n) \ldots ([\![t_m]\!]\ n)
\end{aligned}
$$

A family of *ap* operators

1. $ap_n\ (Abs_m\ f)\ t_1\ \ldots\ t_n =$
   $Abs_{m-n}\ (f\ t_1\ \ldots\ t_n)$
2. $ap_n\ (Abs_m\ f)\ t_1\ \ldots\ t_n = f\ t_1\ \ldots\ t_n$
3. $ap_n\ (Abs_m\ f)\ t_1\ \ldots\ t_n =$
   $ap_{n-m}\ (f\ t_1\ \ldots\ t_m)\ t_{m+1}\ \ldots\ t_n$

Condition on (1): $n < m$
Condition on (2): $n = m$
Condition on (3): $n > m$.

Eval/Apply: Remarks

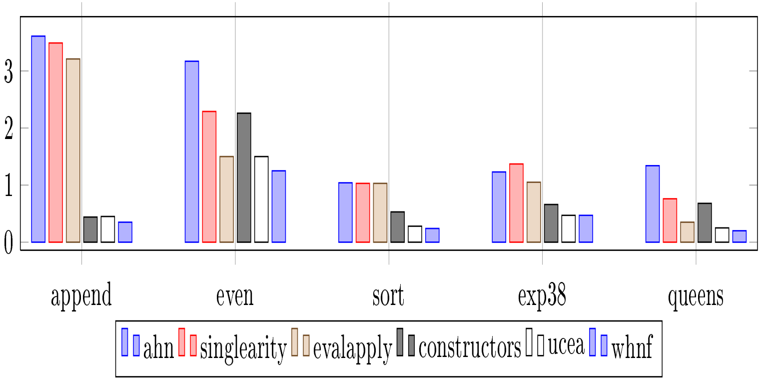
- Drastically reduces number of intermediate closures constructed in the common case.
- No help in pathological cases. But they are rare.
- Only need (small) finite number of *ap* operators.

Many runtime environments compile pattern matching problems to efficient backtracking automata or decision trees.

Idea: Extend model with constructors for all constructors in all datatypes introduced in the object language.

Many runtime environments compile pattern matching problems to efficient backtracking automata or decision trees.

Idea: Extend model with constructors for all constructors in all datatypes introduced in the object language.
$\longrightarrow$ Space and time efficient representation of data.



0
1
2
3
append
even
sort
exp38
queens
ahn
singlearity
evalapply
constructors
ucea
whnf

fast cheap general

fast ✓ cheap general

fast ✓ cheap ✓ general

fast ✓ cheap ✓ general ✓

Final words

Limitations:

- Fixed evaluation order
- Potential impedance mismatch between object-level pattern matching and meta-level pattern matching.

Future work:

- Coq integration
- Short-circuit evaluation.