The Complexity of Interaction Abstract Machines

Beniamino Accattoli\textsuperscript{1} Ugo Dal Lago\textsuperscript{2} Gabriele Vanoni\textsuperscript{2}

\textsuperscript{1}INRIA & LIX, École Polytechnique
\textsuperscript{2}Università di Bologna & INRIA

Bath, 16 June 2020
The Problem: Implementing Higher-Order Functional PL

**Terms**
\[ t, u ::= x \mid \lambda x.t \mid tu \]

**Weak Head Contexts**
\[ H ::= \langle \cdot \rangle \mid Hu \]

**Weak Head Reduction**
\[ H\langle (\lambda x.t)u \rangle \rightarrow_{\beta} H\langle t[x\leftarrow u] \rangle \]

---

**Size Explosion**

There exists a family of \( \lambda \)-terms \( t_n \) such that:
\[
|t_n| = \Theta(n) \quad t_n \rightarrow_{\beta}^{n} \text{whnf}(t_n) \quad |\text{whnf}(t_n)| = \Theta(2^n)
\]

Weak head \( \beta \)-reduction is **not** atomic!
Abstract machines $M$ implement strategies (e.g. weak head reduction).

Their time complexity is a (low-order) polynomial in $n$ and $|t|$.

Is the talk already finished? NO!
Abstract machines allocate **space** for each encountered $\beta$-redex and never deallocate memory.

Space is used in the **worst** possible way.

**Is it possible to do better?** **YES!**
- **Mackie**: small runtime system for PCF.
- **Ghica**: compilation of higher-order functions to digital circuits.
- **Dal Lago** and **Schöpp**: functional programming in LOGSPACE.
We want to be **parsimonious** in **space**.

### Ideas

1. No tracing of $\beta$-redexes
   - $\Rightarrow$ space disentangled from time.

2. **Backtracking** to retrieve $\beta$-redexes
   - $\Rightarrow$ wasting time to gain space.

3. Computation is **local**
   - $\Rightarrow$ a **token** that travels inside the term.

4. The code never changes.
The position of the token inside the term \( t \) is represented via a pair \((u, C)\) such that \( C\langle u \rangle = t \).

Example

The position of the token ■ in the term \( \lambda x.t \) is represented by the position \((t, \lambda x.\langle \cdot \rangle)\).
Traversing $\beta$-redexes

No information is saved, traversing a $\beta$-redex $(\lambda x.t)u$. The token remains untouched.

The token ■ is at the root of the redex.
No information is saved, traversing a $\beta$-redex $(\lambda x.t)u$. The token remains untouched.

The token ■ moves to the left-hand side of the application, changing color.

$$ru \mid C \mid \square \rightarrow_1 r \mid C(\langle \cdot \rangle u) \mid \square$$
No information is saved, traversing a $\beta$-redex $(\lambda x. t)u$. The token remains untouched.

The token $\Box$ moves to the body of the abstraction, changing color again.

\[
\lambda x. t \mid C \mid \Box \rightarrow \bullet_2 \quad t \mid C(\lambda x. \cdot) \mid \blacksquare
\]
Computation in the $\lambda$-calculus is done by substituting variables for arguments.

Our machine first looks for variables, in red mode, going deep inside the term.

The token ■ is pointing a variable.
Computation in the $\lambda$-calculus is done by substituting variables for arguments.

Our machine first looks for variables, in red mode, going deep inside the term.

The token ■ moves locally to the binder, changing its mode to blue, i.e. the machine is now looking for the argument of the variable.

\[
x \mid C\langle\lambda x.D\rangle \mid ■ \rightarrow_{\text{var}} \lambda x.D\langle x\rangle \mid C \mid ■
\]
When an argument is found, it has to be evaluated.

The token @, i.e. looking for an argument is on the left-hand side of an application.
When an argument is found, it has to be evaluated.

The token moves to the argument, changing its mode to red, i.e. looking for the head variable.

\[
t \mid C\langle\cdot\rangle u \quad \rightarrow_{\arg} \quad u \mid C\langle t\rangle \mid
\]
There is also a **backtracking** mechanism (not explained here).

It uses position **enriched** with history/log.

The token is actually made of two data structures, tape and log.
## The Lambda Interaction Abstract Machine: Transitions

<table>
<thead>
<tr>
<th>Sub-term</th>
<th>Context</th>
<th>Log</th>
<th>Tape</th>
<th>Sub-term</th>
<th>Context</th>
<th>Log</th>
<th>Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>$tu$</td>
<td>$C$</td>
<td>$L$</td>
<td>$T$</td>
<td>$\lambda x.t$</td>
<td>$C$</td>
<td>$L$</td>
<td>$\bullet \cdot T$</td>
</tr>
<tr>
<td>$\lambda x.t$</td>
<td>$C$</td>
<td>$L$</td>
<td>$\bullet \cdot T$</td>
<td>$\lambda x.t$</td>
<td>$C\langle\lambda x.\cdot\rangle$</td>
<td>$L$</td>
<td>$T$</td>
</tr>
<tr>
<td>$x$</td>
<td>$C\langle\lambda x.D_n\rangle$</td>
<td>$L_n \cdot L$</td>
<td>$T$</td>
<td>$\lambda x.D_n\langle x\rangle$</td>
<td>$C\langle\lambda x.D_n\rangle$</td>
<td>$L_n \cdot L$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\lambda x.D_n\langle x\rangle$</td>
<td>$C$</td>
<td>$L$</td>
<td>$(x, \lambda x.D_n, L_n) \cdot T$</td>
<td>$\lambda x.D_n\langle x\rangle$</td>
<td>$C\langle\lambda x.D_n\rangle$</td>
<td>$L_n \cdot L$</td>
<td>$T$</td>
</tr>
<tr>
<td>$t\langle\cdot\rangle$</td>
<td>$L$</td>
<td>$\bullet \cdot T$</td>
<td>$\lambda x.t$</td>
<td>$\lambda x.t$</td>
<td>$C\langle\lambda x.\cdot\rangle$</td>
<td>$L$</td>
<td>$\bullet \cdot T$</td>
</tr>
<tr>
<td>$u\langle\cdot\rangle$</td>
<td>$l \cdot L$</td>
<td>$T$</td>
<td>$u$</td>
<td>$u\langle\cdot\rangle$</td>
<td>$C\langle\cdot\rangle t$</td>
<td>$L$</td>
<td>$l \cdot T$</td>
</tr>
<tr>
<td>$\lambda x.D_n\langle x\rangle$</td>
<td>$C\langle\lambda x.D_n\rangle$</td>
<td>$L_n \cdot L$</td>
<td>$T$</td>
<td>$\lambda x.D_n\langle x\rangle$</td>
<td>$C\langle\lambda x.D_n\rangle$</td>
<td>$L_n \cdot L$</td>
<td>$T$</td>
</tr>
</tbody>
</table>

Accattoli, Dal Lago, Vanoni  
The Complexity of Interaction Machines  
Bath, 16 June 2020  
16 / 22
The λ-term \( t \) has weak head normal form if and only if the IAM terminates on \((t, \langle \cdot \rangle, \epsilon, \epsilon)\).
The KAM is the reference abstract machine for call-by-name.

**Theorem**

\[ \lambda\text{-IAM-SPACE}(t) \leq \text{KAM-SPACE}(t) \text{ for each } \lambda\text{-term } t. \]

Moreover, the gap can be **exponential**.

Space consumption of the \( \lambda\)-IAM is disentangled from time.
Unfortunately, space efficiency come at a cost.

**Theorem**

\[
\lambda\text{-IAM\text{-TIME}}(t) \geq \text{KAM\text{-TIME}}(t) \text{ for each } \lambda\text{-term } t.
\]

Again, the gap can be **exponential**.

The KAM is reasonable in time.

\[ \Rightarrow \text{ The } \lambda\text{-IAM is not reasonable in time.} \]
Is it possible to have both time and space efficiency?

- Danos and Regnier devised the $\lambda$-JAM, as a time \textit{optimization} of the $\lambda$-IAM.
- They do not give complexity results.
- Our work: the $\lambda$-JAM is \textit{reasonable} in time, but requires inflationary space.

Roughly, the $\lambda$-JAM fits the same category of the KAM.

It seems that there is an inherent \textit{trade-off} between time and space.
Connections with Game Semantics

Danos and Regnier:
- the $\lambda$-IAM is deeply connected to AJM games;
- the $\lambda$-JAM is isomorphic to the PAM;
- the PAM is deeply connected to HO games.

Therefore:
- AJM = space-efficient;
- HO = time-efficient.
Conclusions

Work in progress

A non-idempotent intersection type system that characterizes space and time consumption of the λ-IAM.

Future work

- Investigate the space-optimality of these machines.
- Hints for space-invariant cost models?
- Formalize complexity properties in suitable proof-assistants.