

# CODE EXTRACTION FROM AGDA TO HVM

Agda is a dependently typed programming language which can ensure that run-time errors cannot happen. It is also a proof assistant which means that it is possible to formally prove properties of Agda code with Agda itself. Can Agda be compiled to the HVM language? How does it perform?

Author  
Matteo Meluzzi  
M.Meluzzi@student.tudelft.nl

Supervisors  
Jesper Cockxs  
J.G.H.Cockx@tudelft.nl  
Lucas Escot  
lucas.escot@ens-lyon.fr

## 1 - Motivation

- Agda has the potential to empower programmers to write robust software easily
- Not yet used in industry
- Partly because all the Agda compilers have some flaws
- Agda compilers target 'classical' languages
- Can we do better if we target a completely different category of programming languages?
- Chosen category: optimal reduction machine (HVM)

## 2 - HVM

- Functional language
- Simple syntax
- Lazy
- Supports lambdas
- Pattern matching
- Untyped
- Agda does the type-checking already
- Uses Interaction Nets to perform computations and memory management
- Never performs same computation twice
- No garbage collector
- Automatic parallelization
- Problem: still a prototype
- Problem: run-time overhead

## 3 - Questions

- For which programs is HVM better/worse?
- Is HVM truly optimal?
- Does HVM use more memory to achieve optimality?
- To what extent is HVM still a prototype?
- Are there programs which make HVM crash and thus undermine the safety of Agda?

## 4 - Analysis

- Running time using HVM compiler (blue) and Haskell compiler (orange)
- The test program computes the sum of a binary tree with  $2^n$  elements (parallelizable)
- After  $n=22$ , HVM is faster because it uses all the CPU cores (6 in the benchmark)

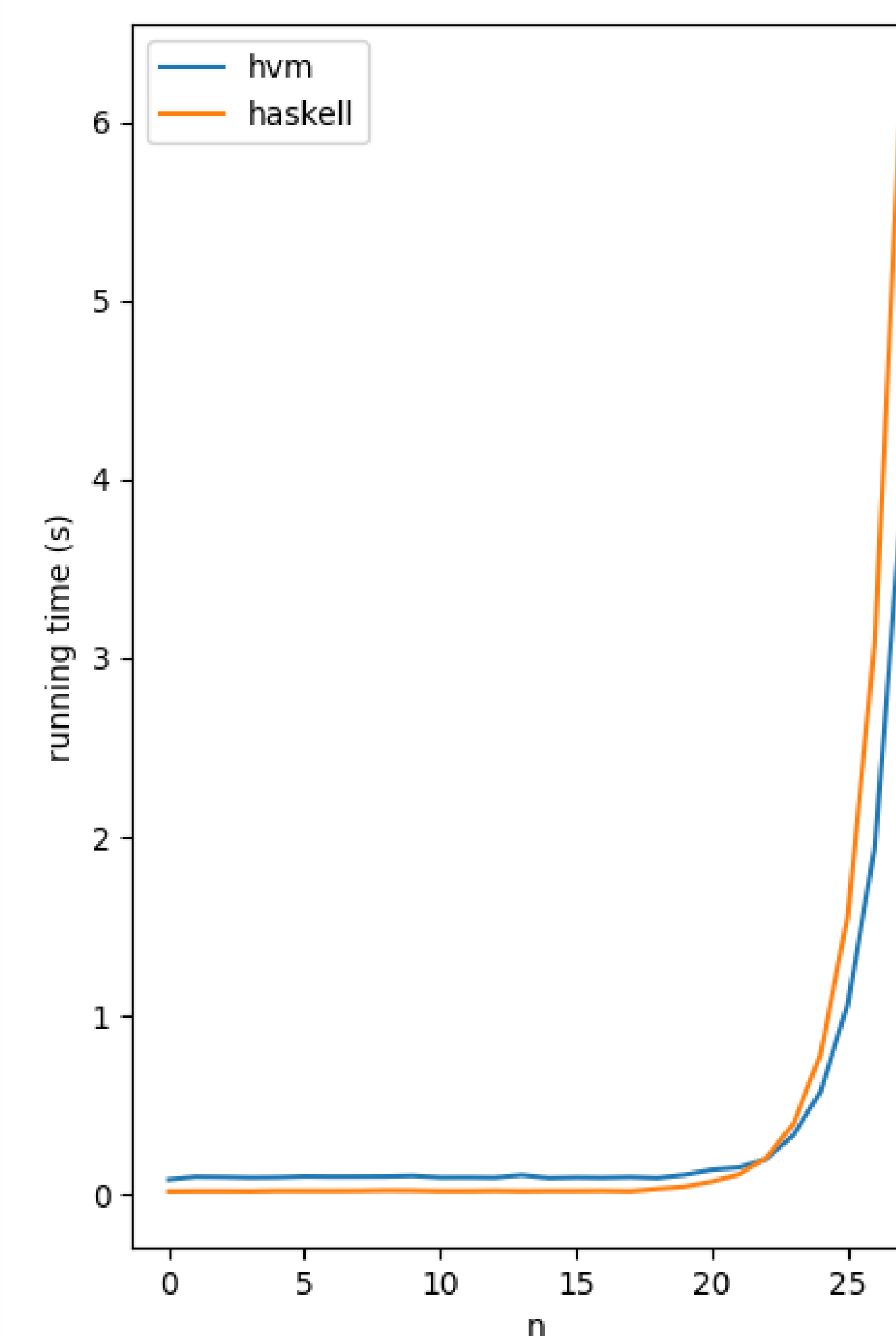


Figure 1: Parallelizable benchmark

- HVM can share computations inside lambdas
- Haskell treats them like black-boxes
- This tests program performs  $2^n$  function compositions
- HVM runs in linear time whereas Haskell in exponential time

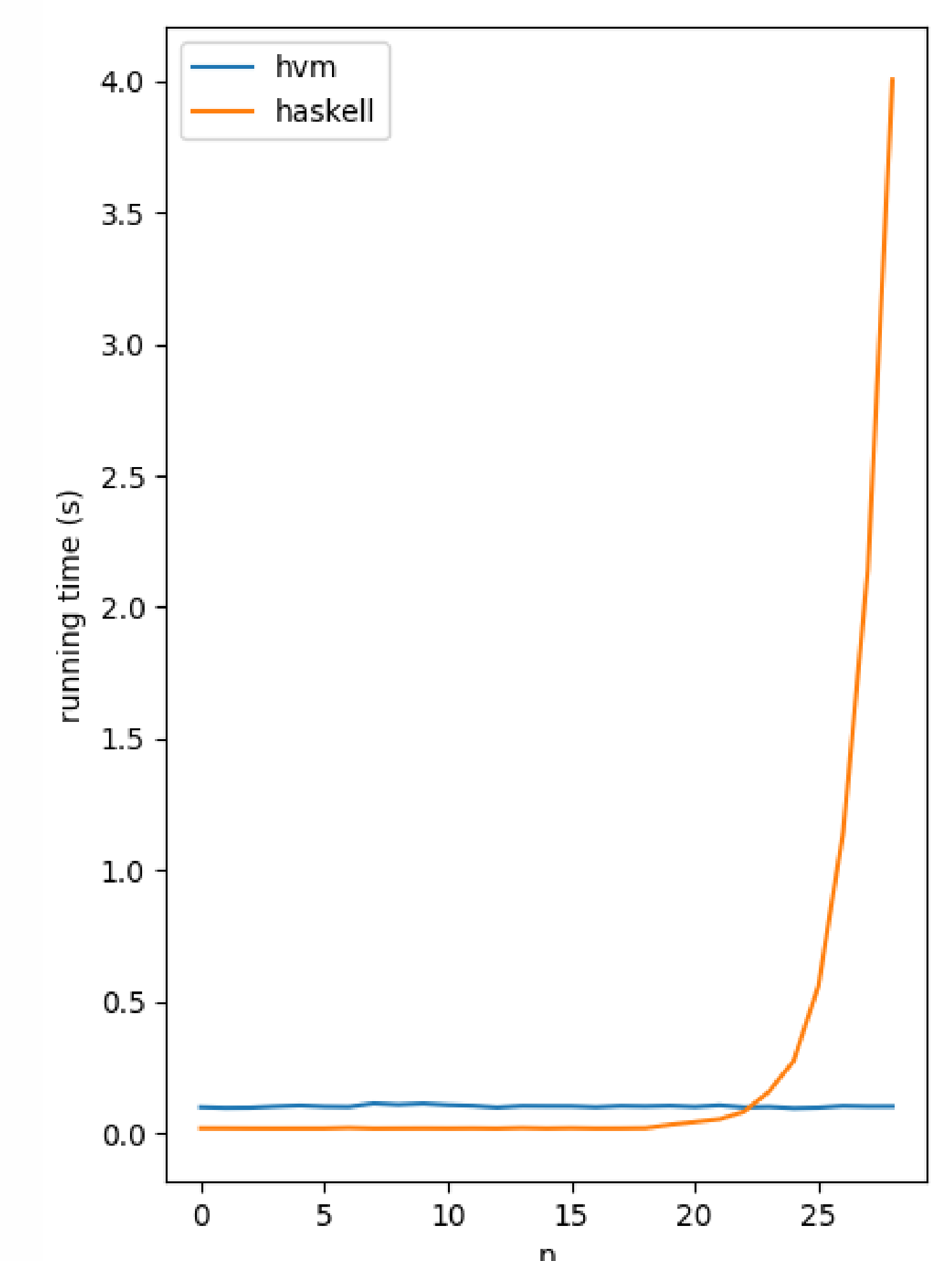


Figure 2: Optimal benchmark

## 5 - Findings

| Test                       | HVM vs Haskell        | Reason                             |
|----------------------------|-----------------------|------------------------------------|
| Optimal                    | Asymptotically faster | Shares computations inside lambdas |
| Parallelizable             | Faster                | Uses all CPU cores                 |
| General case               | Slower                | Less optimised code                |
| Sum of pythagorean triples | Asymptotically slower | Unknown                            |

- Is HVM asymptotically slower because of a bug? Or an error in the Interaction Nets theory? Because of high memory usage?
- There are programs which make HVM behave unexpectedly (stack overflow, bus error, segmentation fault)

## 6 - Conclusion

- HVM is not ready to be used as a core language for Agda
- It is surprising that a prototype can beat a mature compiler such as GHC
- Interaction Nets might be a breakthrough in computational efficiency of functional programming languages
- Further research and tests are needed to understand why HVM can have worse run-time than non-optimal languages