universität innsbruck

Resource-aware programming ...

or what can we learn from Meltdown and Spectre

Georg Moser

cbr.uibk.ac.at

Remember these Guys . . .

[1] Editor’s Letter, CACM Vol. 61, No. 9

Remember these Guys . . .

High-Level Analysis[1]

Because Spectre and Meltdown exploit the performance visibility of speculative actions to create information side channels, they extend the functional specification of the architecture to include its detailed performance.

[1] Editor's Letter, CACM Vol. 61, No. 9

Remember these Guys . . .

High-Level Analysis[1]

Because Spectre and Meltdown exploit the performance visibility of speculative actions to create information side channels, they extend the functional specification of the architecture to include its detailed performance.

[M]aking strong assurances of application security on a computing system requires detailed performance information.

[1] Editor's Letter, CACM Vol. 61, No. 9

Resource as First-Order Citizens

Example

```
/*
sorting of a list |l| using |compare| as a comparison function
*/
sort :: (l: list A) -> (compare: A -> A -> bool) -> list A

|assuming|
   the number of elements of |l| is bounded by |n|
   the size of the elements of |l| is bounded by |m|
|then|
   the number of elements of the result is bounded by |n|
   the size of the elements of the result is bounded by |m|
   the number of calls to |compare| is bounded by |n * log (n)|
   the size of both arguments in all calls to |compare| are
   bounded by |m|
|requiring|
   sequential time |8 * n * log(n) + 4 * n + 3|
   parallel time |6 * log(n) * log(n) + 2|
   storage space |3 * n * m + 2 * m|
```

Outline

- **Logical Foundations and Potential Use Cases**
- **TiML: A Functional Language for Practical Complexity Analysis with Invariants**
- **Complexity of Interaction**

universität
innsbruck

Logical Foundations and Potential Use Cases

TiML: A Functional Language for Practial Complexity Analysis with Invariants

- ML-like language with time-complexity annotations in types
- uses indexed types to express size and worst-case runtime complexity
- allows refinment sorts to constrain indices
- focus is on user-defined annotations, efficient type checking and usability
- allows pattern based type inference, eg. incorporating the Master Theorem

TiML: A Functional Language for Practial Complexity Analysis with Invariants

- ML-like language with time-complexity annotations in types
- uses indexed types to express size and worst-case runtime complexity
- allows refinment sorts to constrain indices
- focus is on user-defined annotations, efficient type checking and usability
- allows pattern based type inference, eg. incorporating the Master Theorem

Complexity of Interaction

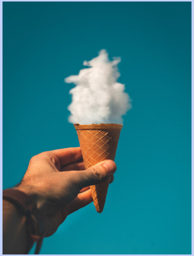
- runtime and space complexity analysis of interaction net systems
- uses sized types and scheduled types, the latter govern productivity of rules in parallel computation
- INs provide an intermediary representation of ML-like languages
- graph-based computation model generalising linear logic proof nets

Use Cases

High Performance Computing

Edge/Fog/Cloud Computing

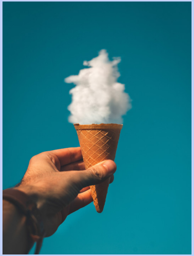
Cloud Tenant



Cloud Provider

Edge/Fog/Cloud Computing

Cloud Tenant



Cloud Provider

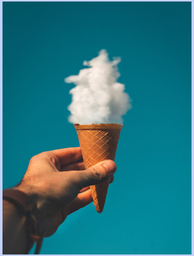
Execution Platform

Edge/Fog/Cloud Computing

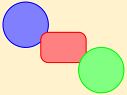
Cloud Tenant

Automation aka

Type Inference



Cloud Provider



Resource-Aware

Programming

Execution Platform

Edge/Fog/Cloud Computing

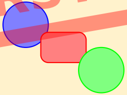
Automation aka

Type Inference

Cloud Tenant

Cloud Provider

FIRST CLASS PROPERTIES



Resource-Aware Programming

Execution Platform

universität
innsbruck

TiML: A Functional Language for Practical Complexity Analysis with Invariants

A “third way” for Resource Analysis

Example

```
datatype list α : {ℕ} = Nil of list α {0}
| Cons of α * list α {n} --> list a {n+1}

fun foldl [α β] {m n : ℕ} (f : α * β -m-> β acc (l : list α {n})
    return β using (m+4) * n =
    case l of
    [] => acc
    | x :: xs => foldl f (f (x, acc)) xs
```

indexed type system induces the following constraint problem

$$\forall m, n, n' \; n' + 1 = n \Rightarrow m + 4 + (m+4)n' \leqslant (m+4)n$$

Peng Wang, Di Wang, and Adam Chlipala.
TiML: A Functional Language for Practical Complexity Analysis with Invariants.
Proc. ACM on Programming Languages, 1(OOPSLA):79:1–79:26, 2017.

Type Checking and Inference

- evaluated on medium-sized benchmarks; list and tree operations as well as amortised data structures
- type checking is fast; annotation burden is significant

Type Checking and Inference

- evaluated on medium-sized benchmarks; list and tree operations as well as amortised data structures
- type checking is fast; annotation burden is significant
- type inference allows big-O notation in abstract indices
- eg index sort T_msort represents $O(mn \log n)$

Type Checking and Inference

- evaluated on medium-sized benchmarks; list and tree operations as well as amortised data structures
- type checking is fast; annotation burden is significant
- type inference allows big-O notation in abstract indices
- eg index sort T_msort represents $O(mn \log n)$
- pattern-based type inference is restrictive
- a number of benchmark example can be analysed fully automatically by various tools (sorting, functional queues, etc.)

Type Checking and Inference

- evaluated on medium-sized benchmarks; list and tree operations as well as amortised data structures
- type checking is fast; annotation burden is significant
- type inference allows big-O notation in abstract indices
- eg index sort T_msort represents $O(mn \log n)$
- pattern-based type inference is restrictive
- a number of benchmark example can be analysed fully automatically by various tools (sorting, functional queues, etc.)

Usability

[...] an undergraduate student with background in SML took just one day to become fluent in writing and annotating TiML programs.

Interlude: Automated Amortised Resource Analysis

Example (TiML benchmark example)

```
empty x = (nil,nil);

checkF (f,r) = match f with
                | nil -> (rev(r),nil)
                | (x::xs) -> (f,r);

snoc (queue,x) = match queue with
                | (f,r) -> checkF(f,x::r);

enq n = match n with
                | 0 -> empty()
                | S n' -> snoc(enq(n'),n');

main = enq 3;
main = ([0],[3,2,1])
```

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1 : A_1, \ldots, x_n : A_n \,|\!\!\frac{p}{\quad}\, f(x_1, \ldots, x_n) : C}$$

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1 : A_1, \ldots, x_n : A_n \mid\!\!\frac{p}{\quad} f(x_1, \ldots, x_n) : C}$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \mid\!\!\frac{p_0}{\quad} f(x_1, \ldots, x_n) : C \quad \Gamma_1 \mid\!\!\frac{p_1}{\quad} t_1 : A_1 \ \cdots \ \Gamma_n \mid\!\!\frac{p_n}{\quad} t_n : A_n}{\Gamma_1, \ldots, \Gamma_n \mid\!\!\frac{p}{\quad} f(t_1, \ldots, t_n) : C}$$

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1 \colon A_1, \ldots, x_n \colon A_n \vdash^{p} f(x_1, \ldots, x_n) \colon C} \qquad \frac{}{x \colon A \vdash^{0} x \colon A}$$

$$\frac{x_1 \colon A_1, \ldots, x_n \colon A_n \vdash^{p_0} f(x_1, \ldots, x_n) \colon C \quad \Gamma_1 \vdash^{p_1} t_1 \colon A_1 \ \cdots \ \Gamma_n \vdash^{p_n} t_n \colon A_n}{\Gamma_1, \ldots, \Gamma_n \vdash^{p} f(t_1, \ldots, t_n) \colon C}$$

$$\frac{\Gamma \vdash^{p} t \colon C}{\Gamma, x \colon A \vdash^{p} t \colon C}$$

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1{:}A_1, \dots, x_n{:}A_n \vdash^{p} f(x_1, \dots, x_n){:}C} \qquad \frac{}{x{:}A \vdash^{0} x{:}A}$$

$$\frac{x_1{:}A_1, \dots, x_n{:}A_n \vdash^{p_0} f(x_1, \dots, x_n){:}C \quad \Gamma_1 \vdash^{p_1} t_1{:}A_1 \quad \cdots \quad \Gamma_n \vdash^{p_n} t_n{:}A_n}{\Gamma_1, \dots, \Gamma_n \vdash^{p} f(t_1, \dots, t_n){:}C}$$

$$\frac{\Gamma \vdash^{p} t{:}C}{\Gamma, x{:}A \vdash^{p} t{:}C} \qquad \frac{\Gamma, x{:}A_1, y{:}A_2 \vdash^{p} t[x, y]{:}C \quad \curlyvee(A \mid A_1, A_2)}{\Gamma, z{:}A \vdash^{p} t[z, z]{:}C}$$

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1{:}\,A_1, \dots, x_n{:}\,A_n \mid\!\!\frac{p}{\quad} f(x_1, \dots, x_n){:}\,C} \qquad \frac{}{x{:}\,A \mid\!\!\frac{0}{\quad} x{:}\,A}$$

$$\frac{x_1{:}\,A_1, \dots, x_n{:}\,A_n \mid\!\!\frac{p_0}{\quad} f(x_1, \dots, x_n){:}\,C \quad \Gamma_1 \mid\!\!\frac{p_1}{\quad} t_1{:}\,A_1 \quad \cdots \quad \Gamma_n \mid\!\!\frac{p_n}{\quad} t_n{:}\,A_n}{\Gamma_1, \dots, \Gamma_n \mid\!\!\frac{p}{\quad} f(t_1, \dots, t_n){:}\,C}$$

$$\frac{\Gamma \mid\!\!\frac{p}{\quad} t{:}\,C}{\Gamma, x{:}\,A \mid\!\!\frac{p}{\quad} t{:}\,C} \qquad \frac{\Gamma, x{:}\,A_1, y{:}\,A_2 \mid\!\!\frac{p}{\quad} t[x,y]{:}\,C \quad \curlyvee(A \mid A_1, A_2)}{\Gamma, z{:}\,A \mid\!\!\frac{p}{\quad} t[z,z]{:}\,C}$$

Theorem

let TRS $\mathcal{R}$ and subsitution σ be well-typed, suppose $\Gamma \mid\!\!\frac{p}{\quad} t{:}\,A$ and $t\sigma \xrightarrow{\mathrm{i}}{}^{m}_{\mathcal{R}} v$ then

$$\Phi(\sigma{:}\,\Gamma) - \Phi(v{:}\,A) + p \geqslant m$$

Definition (Annotated Type System for TRSs (selection))

$$\frac{f \text{ a function symbol} \quad [A_1 \times \cdots \times A_n] \xrightarrow{p} C \in \mathcal{F}(f)}{x_1\colon A_1, \dots, x_n\colon A_n \stackrel{p}{\vdash} f(x_1, \dots, x_n)\colon C} \qquad \frac{}{x\colon A \stackrel{0}{\vdash} x\colon A}$$

$$\frac{x_1\colon A_1, \dots, x_n\colon A_n \stackrel{p_0}{\vdash} f(x_1, \dots, x_n)\colon C \quad \Gamma_1 \stackrel{p_1}{\vdash} t_1\colon A_1 \quad \cdots \quad \Gamma_n \stackrel{p_n}{\vdash} t_n\colon A_n}{\Gamma_1, \dots, \Gamma_n \stackrel{p}{\vdash} f(t_1, \dots, t_n)\colon C}$$

$$\frac{\Gamma \stackrel{p}{\vdash} t\colon C}{\Gamma, x\colon A \stackrel{p}{\vdash} t\colon C} \qquad \frac{\Gamma, x\colon A_1, y\colon A_2 \stackrel{p}{\vdash} t[x,y]\colon C \quad \curlyvee(A \mid A_1, A_2)}{\Gamma, z\colon A \stackrel{p}{\vdash} t[z,z]\colon C}$$

Theorem

let TRS $\mathcal{R}$ and subsitution σ be well-typed, suppose $\Gamma \stackrel{p}{\vdash} t\colon A$ and $t\sigma \xrightarrow{\mathrm{i}}{}^{m}_{\mathcal{R}} v$ then

$$\Phi(\sigma\colon \Gamma) - \Phi(v\colon A) + p \geqslant m$$

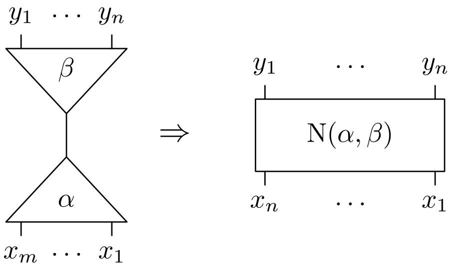
universität
innsbruck

Complexity of Interaction

A Logic-Based Computation Model for Distributed Computing

Definition

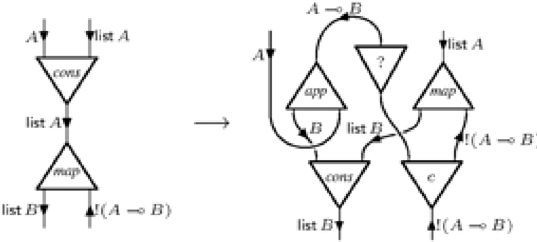
- graph-based
- linear logic proof nets
- benign parallel computations
- asynchronous, local inferences



A Logic-Based Computation Model for Distributed Computing

Definition

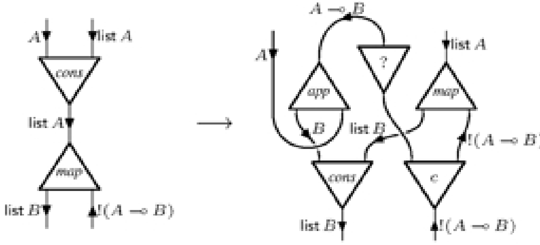
- graph-based
- linear logic proof nets
- benign parallel computations
- asynchronous, local inferences



A Logic-Based Computation Model for Distributed Computing

Definition

- graph-based
- linear logic proof nets
- benign parallel computations
- asynchronous, local inferences



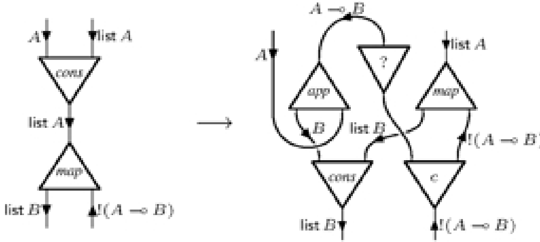
Remarks

- interaction nets provide a Turing-complete computation model, where distribution of computation is natively build in
- intermediary representation language, programs need to be compiled to

A Logic-Based Computation Model for Distributed Computing

Definition

- graph-based
- linear logic proof nets
- benign parallel computations
- asynchronous, local inferences



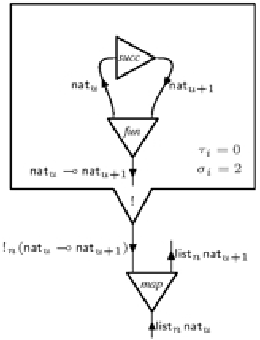
Remarks

- interaction nets provide a Turing-complete computation model, where distribution of computation is natively build in
- intermediary representation language, programs need to be compiled to
- resource analysis für sequential/parallel/distributed computation, no tool support

Implementation of Interaction Nets on a Grid

computation is localised

Complexity of Interaction



Definition

- the types associated to the ports are refined by sized types and scheduled types
- runtime/space/productivity analysis
- provides a resource analysis for sequential and parallel execution
- scheduled types guarantee availability pace of data
- resource analysis works for higher-order, based on a weak sequential cost model

S. Gimenez, GM.
The Complexity of Interaction.
In *Proc. 43th POPL*, pages 243-255, 2016

Thank You for Your Attention