Complexity Information Flow in a Multi-threaded Imperative Language

Jean-Yves Marion and Romain Péchoux

Université de Lorraine, LORIA, Nancy, France

TAMC 2014

Implicit Computational Complexity

ICC's goal is to **find machine independent characterizations** of complexity classes on **distinct paradigms**:

- **Functional languages**:
  - Function algebra (Cook, ...), Lights logics (Girard, ...), Interpretations of TRS (Marion, ...), Non-size increasing principle (Hofmann, ...)
- **Imperative languages**:
  - Matrices calculus for imperative programs(Jones, ...), Imperative pointer graph languages (Hofmann, ...)

Mixture

Marion's idea (Lics 2011) is to **mix**:

- **Safe recursion** by Bellantoni and Cook [1992]
- **Non-interference** by Volpano et al [1996]

in order to obtain a **polynomial time characterization** on imperative languages.

Safe recursion

The functions defined by:

- constants, successor, ...
- **safe composition**:

$$f(\overline{x}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) = h(r(\overline{x}^{\mathbf{1}};);t(\overline{x}^{\mathbf{1}};\overline{a})^{\mathbf{0}})$$

- and **safe recursion**:

$$\begin{aligned} f(0,\overline{x}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) &= g(\overline{x}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) \\ f(i(x)^{\mathbf{1}},\overline{y}^{\mathbf{1}};\overline{a}) &= h_i(x^{\mathbf{1}},\overline{y}^{\mathbf{1}};f(x^{\mathbf{1}},\overline{y}^{\mathbf{1}};\overline{a})^{\mathbf{0}}) \qquad i \in \{0,1\}, \end{aligned}$$

is exactly **FPtime**.

Non-interference

Security levels $\tau \in \{L, H\}$.

Typing rules of the shape:

$$\frac{\Gamma \vdash E : \tau \quad \Gamma \vdash I : \tau\ Cmd}{\Gamma \vdash \texttt{while}(E)\{I\} : \tau\ Cmd}\ (Wh)$$

+ command subtyping:

$$\frac{\Gamma \vdash I : \tau\ Cmd \quad \tau < \tau'}{\Gamma \vdash I : \tau'\ Cmd}\ (Sub)$$

Non-interference example

It prevent us from typing the following program:

```
while(x>0 : H) {
    x = x-- ; : H Cmd
    y = y++ ; : L Cmd
}
```

if x is High and y is Low and provided that $H < L$.

Small imperative language

Every data type is encoded by words over $\mathbf{W}$.

- Expressions :
  $E ::= \quad \mathtt{x} \mid c \mid \mathtt{true} \mid \mathtt{false} \mid op(\overline{E})$
- Instructions :
  $I ::= \quad ; \mid [\tau]\ \mathtt{x}{:=}E; \mid I_1\ I_2 \mid \mathtt{while}(E)\{I\}$
  $\mid \mathtt{if}(E)\{I_1\}\mathtt{else}\{I_2\}$

The types τ will be tiers in $\{\mathbf{0}, \mathbf{1}\}$ such that $\mathbf{0} < \mathbf{1}$.

Typing rules : expressions

Variable

$$\frac{\Gamma(\mathtt{x}) = \tau}{\Gamma \vdash \mathtt{x} : \tau}$$

Constant

$$\frac{}{\Gamma \vdash n : \tau}$$

Destructor

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash op(e) : \tau}$$

Constructor

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash op(e) : \mathbf{0}}$$

Typing rules : commands

Assign

$$\frac{\Gamma \vdash \mathtt{x} : \tau \qquad \Gamma \vdash E : \tau'}{\Gamma \vdash \mathtt{x} := E : \tau} \; \tau \leq \tau'$$

Compose

$$\frac{\Gamma \vdash I_1 : \tau \qquad \Gamma \vdash I_2 : \tau'}{\Gamma \vdash I_1 \; I_2 : \tau \vee \tau'}$$

If

$$\frac{\Gamma \vdash e : \tau \qquad \Gamma \vdash I_i : \tau}{\Gamma \vdash \mathtt{if}(E)\{I_1\}\mathtt{else}\{I_2\} : \tau}$$

While

$$\frac{\Gamma \vdash E : \mathbf{1} \qquad \Gamma \vdash I : \tau}{\Gamma \vdash \mathtt{while}(E)\{I\} : \mathbf{1}}$$

Example: addition

```
int add(int x,int y)
{
while (x>0)
          {
          x--;
          y++;
          }
return y
}
```

- y is necessarily of tier **0**
- x is necessarily of tier **1**
- and, consequently, add :: $\mathbf{1} \times \mathbf{0} \to \mathbf{0}$

Example: multiplication

```
int mult(int x,int y)
{
int z=0;
while (x)
        {
        x--;
        z = add(y,z);
        }
return z;
}
```

- the output of add is **0**. Consequently, z is of tier **0**.
- both x and y are of tier **1**
- and, consequently, `mult` :: $\mathbf{1} \times \mathbf{1} \to \mathbf{0}$

Example: exponential

```
int expo(int x)
{
int y=1;
while (x)
          {
          x--;
          y = add(y,y);
          }
return y;
}
```

- x is of tier **1**,
- the output of add is of tier **0**,
- but y has to be of tier **1** in the first argument of add !!!

Results

Theorem [Subject reduction (TAMC 2014)]

If $\sigma \vDash I \to \sigma' \vDash I'$ and $\Gamma \vdash I : \tau$ then $\Gamma \vdash I' : \tau'$ where $\tau' \leq \tau$.

Theorem [FPtime characterization (Marion, Lics 2011)]

The set of functions computable by a typable and terminating program with FPtime computable operators is exactly FPtime.

Theorem [Type inference decidability (Fossacs 2013)]

Type inference can be done in polynomial time.

Multi-threaded

Now we consider multi-threads M to be a fixed collection of commands:

$$M(\alpha) = I,\ \alpha \in dom(M)$$

and non-deterministic reduction:

$$\frac{M(\alpha) = I \quad \sigma \vDash I \to \sigma_1 \vDash I_1}{\sigma \vDash M \ \to\ \sigma_1 \vDash M[\alpha := I_1]}\ (Step) \qquad \frac{M(\alpha) = I \quad \sigma \vDash I \to \sigma_1}{\sigma \vDash M \ \to\ \sigma_1 \vDash M - \alpha}\ (Stop)$$

and we extend the typing rule by:

$$\frac{\forall \alpha \in dom(M),\ \exists \tau,\ \Gamma \vdash M(\alpha) : \tau}{\Gamma \vdash M : \diamond}\ (Multi)$$

Results

Theorem [Polynomial time soundness (TAMC 2014)]

A typable and strongly normalizing multi-thread terminates in a polynomially bounded number of transitions.

The strong normalization assumption can be weakened under a fair scheduling policy (depending only on M and tier **1** values):

Theorem [Marion and Péchoux (TAMC 2014)]

A typable a multi-thread terminating under a fair scheduling policy terminates in a polynomially bounded number of transitions.

Theorem [Type inference decidability (Fossacs 2013)]

Type inference can be done in polynomial time.

Conclusion

A static analysis for resource consumption dealing with threads and polynomial time.

Drawbacks and Open questions

- Not intentionnally complete: improve expressiveness by program transformation
- Capture Thread creation (work in progress)
- Do the implementation
- Extend the characterizations (BPP, ...)