Feasible Reactivity for Synchronous Cooperative Threads

F. Dabrowski
INRIA Sophia-Antipolis

Joint Work With R.M. Amadio

Systems described by means of *cooperative threads*

- execution proceeds in **synchronous rounds (instants)**
- which interact by means of **shared signals (broadcast)**

The SL language[Boussinot-De Simone, 95]

Reaction to the absence of a signal (**boolean**) within an instant can only happen at the next instant (relaxation of the ESTEREL model [Berry-Gonthier,1992]).

Main Property : Reactivity

Goal

Ensure the reactivity with signals carrying values

Reactivity

Each thread should have the opportunity to react

- threads must cooperate (presence)
- instants must terminate (absence) $\longleftarrow$ stronger condition

Feasible Reactivity

- Instants must terminate in reasonable time
- Polynomial time seems to be a good candidate

Goals

Termination of the instants

Well-founded orders: Term rewriting systems

Polynomial time

A program that reacts in polynomial time should not produce values of, say, exponential size.
Quasi-interpretation + control flow analysis: The size of the values computed within an instant is polynomially bounded in the size of the parameters at the beginning of the instant.

Asymptotic Control

Parameters hiding:(call graph) Control the size of the parameters at the beginning of the instants.

Types and Constructors

(1) $t = \cdots \mid \mathsf{c}\ \mathit{of}\ t_1, \ldots, t_n \mid \cdots$

$$\begin{array}{lcl} \mathit{nat} & = & \mathsf{zero} \mid \mathsf{succ}\ \mathit{of}\ \mathit{nat} \\ \mathit{list} & = & \mathsf{nil} \mid \mathsf{cons}\ \mathit{of}\ \mathit{nat}, \mathit{list} \end{array}$$

(2) $t = \mathit{Sig}(t')$ with $\cdots \mid \mathsf{r} := v \mid \cdots$

t is the type of signals carrying values of type t'

r is a value of type t with v as initial value

Functions

$$f(x_1, \ldots, x_n) = eb$$

$$\begin{array}{lcl} eb & ::= & e \parallel \mathit{match}\ x\ \mathit{with}\ \ldots p \Rightarrow eb \ldots \\ e & ::= & x \parallel \mathsf{c}(e, \ldots, e) \parallel f(e, \ldots, e) \end{array}$$

$$nat = \mathsf{zero} \mid \mathsf{succ}\ of\ nat$$

$$min(x, y) = match\ \mathbf{x}\ with$$

$$\mathsf{zero} \Rightarrow x$$

$$\mathsf{succ}(\mathbf{z}) \Rightarrow match\ \mathbf{y}\ with$$

$$\mathsf{z} \Rightarrow y$$

$$\mathsf{s}(\mathbf{t}) \Rightarrow \mathsf{s}(min(\mathbf{z}, \mathbf{t}))$$

$$\boxed{f(x_1, \ldots, x_n) = b}$$

$$
\begin{array}{rll}
b ::= & & stop \\
& \mid & f(e, \ldots, e) \\
& \mid & match\ x\ with \cdots p \Rightarrow b \cdots \\
& \mid & yield.b \\
& \mid & next.f(e, \ldots, e) \\
& \mid & \varrho := e.b \\
& \mid & read\ \varrho\ with\ \ldots\ p \Rightarrow b \ldots [x] \Rightarrow f(e, \ldots, e)
\end{array}
$$

reset of signals at the beginning of each instant.

Example

Client

$$
\begin{aligned}
g(s, r, y) \;&= read\ s\ with\ l \Rightarrow \\
&\qquad s := \mathrm{cons}(\mathrm{req}(r, y), l). \\
&\qquad yield.read\ r\ with\ z \ldots
\end{aligned}
$$

Server

$$
\begin{aligned}
f(s, x) \;&=\; yield.read\ s\ with\ l \Rightarrow f'(s, x, l) \\
\\
f'(s, x, l) \;&=\; match\ l\ with \\
&\qquad\quad \mathrm{nil} \Rightarrow next.f(s, x) \\
&\qquad \mid\ \mathrm{cons}(\mathrm{req}(r, y), l') \Rightarrow \\
&\qquad\qquad r := h_1(y, x).f'(s, h_2(y, x), l')
\end{aligned}
$$

The read once condition

Hypothesis

Each thread performs any given read instruction at most once in an instant (control flow analysis).

Auxiliary parameters: assign a fresh label to each read instruction and consider the signature extended by these new parameters.

Example

$$f^{+}(s, x, \mathbf{y}) \quad = \quad \mathit{yield}.\mathit{read}_{\langle \mathbf{y} \rangle}\ s\ \mathit{with}\ l \Rightarrow f'^{+}(s, x, l)[\mathbf{y}/l]$$

Remark: the computation of f within an instant is a *function* of the parameters and the values read within the instant.

Constraints

$$f(p_1, \ldots, p_n) \succeq_i e \quad i \in \{0, 1, 2\}$$

0	:	termination of instants	*
$0, 1$	:	bound on the size of values (in the instant)	*
$0, 1, 2$	:	control for arbitrary many instants	

* Previous paper at Concur'04 by Amadio and Dalzilio

Definition

We say that a program **reacts in polynomial time** if there exists a polynomial (in the parameters at the beginning of the computation) that bounds the lenght of the instants.

Asymptotic Control

Problem : Control the size of the parameters when going from one instant to the other (Non-Size Increasing property)

1. Some values may depend on a read instruction
2. Some parameters may be discarded

We hide some of the parameters in instantaneous function calls.

$$
\begin{array}{rcl}
f(s,x,\mathbf{y}) & = & \mathit{yield}.\mathit{read}_{\mathbf{y}}\ s\ \mathit{with}\ l \Rightarrow f'(s,x,l)^{(a)}\mathbf{0,2} \\
f'(s,x,l) & = & \mathit{match}\ l\ \mathit{with} \\
 & & \quad \mathrm{nil} \Rightarrow \mathit{next}.f(s,x)^{(b)}\mathbf{2} \\
 & & \mid\ \mathrm{cons}(\mathrm{req}(r,y),l') \Rightarrow \\
 & & \qquad r := h_1(y,x)^{(c)}\mathbf{1}.f'(s,h_2(y,x),l')^{(d)}\mathbf{0,2}
\end{array}
$$

CONSTRAINTS

Constraints :

(a) $f^{+}(s,x;l) \succeq_0 f'^{+}(s,x,l;)$

$f^{+}(s,x;0) \succeq_2 f'^{+}(s,x,0;)$

(b) $f'^{+}(s,x,0;) \succeq_2 f^{+}(s,x;0)$

(c) $f'^{+}(s,x,\mathrm{cons}(\mathrm{req}(r,y),l');) \succeq_1 h_1(y,x)$

(d) $(\ldots)$

QUASI-INTERPRETATIONS [BONFANTE,MARION,MOYEN,01]

- Bound the size of the values computed by a program
- Inspired by Polynomial interpretations (termination of TRS)
- Synthesis Problem : [Amadio, 2003]

Assignments

1. If c is a constructor with arity 0

$$q_c = 0$$

2. If c is a constructor with arity $n > 0$

$$q_c(x_1, \ldots, x_n) = d + \Sigma_{i \in 1..n} x_i, \qquad \text{where } d \geq 1 \in \mathbf{N}.$$

3. If f is a function symbol with arity n

$$q_f : (\mathbf{N})^n \rightarrow \mathbf{N} \qquad \text{is a monotonic function.}$$

We associate with a closed expression e a natural number q_e as follows:

$$q_{h(e_1,\ldots,e_n)} = q_h(q_{e_1}, \ldots, q_{e_n}) \; .$$

and we define

$$\begin{array}{l} q \models e_1 \succeq e_2 \text{ if } \forall\,\sigma \;\; q_{\sigma e_1} \geq q_{\sigma e_2} \\ q \models e_1 \succ e_2 \text{ if } \forall\,\sigma \;\; q_{\sigma e_1} > q_{\sigma e_2} \end{array}$$

Size Bounds

An assignment q is a **quasi-interpretation** if

- it satisfies the constraints of index 1 and 2
- $f(x_1, \ldots, x_n) \geq x_i$

Theorem

If a program P has a polynomial quasi-interpretation then the size of the largest value computed by P is polynomial in the size of the parameters of the program at the beginning of the computation.

More Precisely,

1. control the parameters at the beginning of the instants.
2. values computed during an instant are bounded by a polynomial in the size of
   - the parameters at the beginning of the instant
   - the values read during the instant
3. values computed during an instant are bounded by $U^{n.m+1}(c)$.

   (m: #reads, n: #threads, U: poly. bound on the Q.I, c: bound on the Q.I of the parameters at the beginning of the instant.)

Constraints of index 0

$$f(p_1, \ldots, p_n) \succeq_0 C[g(e_1, \ldots, e_n)]$$

where

C	is a one hole context,
$f =_G g$	according to the least preorder induced by function calls.

- Assume those constraints are **linear**.
- Associate a status $st \in \{mul, lex\}$ to each function symbol.

Definition

The quasi-interpretation of a program is **compatible** with the order if in all constraints

$$q \models (p_1, \ldots, p_n) >_{lex} (e_1, \ldots, e_n)$$

Remark: This condition is a particular case of the **Size Change Principle** (multi-set and lexicographic comparison).

Theorem

Hypothesis:

the program has a **compatible** and **polynomially bounded** quasi-interpretation.

Conclusion:

any instant terminates in time **polynomial** in the size of the parameters at the **beginning of the instant**.

Corollary

A program that admits a **compatible** and **polynomially bounded** quasi-interpretation reacts in polynomial time.

Summary

Static analysis method which guarantees

- Polynomial bounds on the memory needed to run programs.
- Reactivity in polynomial time.

Asynchronous cycles

We consider cyclic behaviors where

One cycle = One Instant

But some computations may require more than one instant

First solution: Synchronous cycles

two kinds of instants, not modular

Second solution: Asynchronous cycles

Each behavior defines its own cycles

Asynchronous cycles

Problem

The read once condition is not stable through parallel composition of (asynchronous) cycles

Solution

Control flow through typing (type and effect system)

Extensions

Registers

$$\begin{array}{rcl} f(x) & = & \textit{read } x \textit{ with } [y] \Rightarrow g(x,y) \\ g(x,y) & = & x := y.f(x) \end{array}$$

- Static analysis enforces non-size increasing values in registers.
- Stratification: no cyclic read/write computations

Any questions...