Process, Systems and Tests: Three Layers in Concurrent Computation (Short Paper)

Combined 27th International Workshop on Expressiveness in Concurrency and 17th Workshop on Structural Operational Semantics

Clément Aubert[1] Daniele Varacca[2]

[1] Augusta University, School of Computer & Cyber Sciences, GA, USA

[2] LACL UPEC, France

~~Vienna, Austria~~ ONLINE August 31st, 2020

Contextual Reasoning: "Morris-style"

Framework

1. A syntax and a semantics $P \to P'$
2. Some observations $P \downarrow$
3. An order[a]: $Q \leq Q'$ if for every context $C[\,]$ if $C[Q] \to\downarrow$ then $C[Q'] \to\downarrow$

[a] With nondeterminism, play the (bi)-simulation game.

Motto

"*Two terms are equivalent if they produce the same observations in every context.*"

But. . .

who does the context represent?

Contextual Reasoning: Who Is the Context?

A context can represent either…

a (hostile) environment,

some other part of the program.

Not the same point of view!

Let's take the perspective of a programming language, and see the difference between a snippet of code and a function.

Contextual Reasoning: A Java Perspective

A snippet

```
while(i < 10){
    System.out.print(i);
    i ++;
}
```

Contextual Reasoning: A Java Perspective

A snippet

```
public class HelloWorld{
    public static void main(String []args){
        int i = 0;
        while(i < 10){
            System.out.print(i);
            i ++;
        }
    }
}
```

What can the context do?

Add more instructions, declare the variables, declare the function name, . . .

Contextual Reasoning: A Java Perspective

A snippet

```java
public class HelloWorld{
    public static void main(String []args){
        int i = 0;
        for (; i<9; i++){
            System.out.print(i+1);

        }
    }
}
```

Who is the context?

The context is the programmer. Two snippets are the same if the programmer can subtitute one for the other during programming.

Contextual Reasoning: A Java Perspective

A Function

```java
public class Toto{
  public int f(int x){
    int i = 0; int a = 0;
    while(i < x){
      a = a+i;
      i ++;
    }
    return a;
  }}
```

What can the context do?

Call the function!

Contextual Reasoning: A Java Perspective

A Function

```
public class Toto{
  public int f(int x){
    int i = 0; int a = 0;
    while(i < x){
      a = a+i;
      i ++;
    }
    return a;
  }}
```

```
public class HelloWorld{
  public static void main(){
    Toto t = new Toto();
    System.out.print(
        t.f(12)
    );
  }
}
```

Who is the context?

The context is the user (potentially hostile). Two functions are the same if the user cannot distinguish them by calling them.

Contextual Reasoning: A π-Calculus Perspective

Usual Syntax

$$P = 0 \mid a(x).P \mid \bar{a}(M).P \mid (\nu a)P \mid P \parallel P \mid P + P \mid \cdots$$

Which contexts can we use?

Programmer "All!"
"I'm *building* my process."

User "Only parallel composition."
"I'm *communicating* with my process."

Yet, contexts are *on the surface* often defined as a monolithic concept.

Contextual Reasoning: Open and Closed Terms

On Open and Closed Terms

Open terms Still being built.

Closed terms Ready to be released.

The original π calculus presentation blurs the distinction ("Everything is a name").
Later presentations reintroduce it.

Contextual Reasoning: Open and Closed Terms

On Open and Closed Terms

Open terms Still being built.

Closed terms Ready to be released.

The original π calculus presentation blurs the distinction ("Everything is a name").
Later presentations reintroduce it.

~~Open or Close~~ $\Rightarrow$ Ready to be dispatched or not

Our Proposal: Processes, Systems and Tests

1. Define processes.

2. Define deployment criteria to transform processes into systems.

3. Define tests on systems.

Our Proposal: Processes, Systems and Tests

1. Define processes.
    1. Choose the *construction operators*.
    2. The programmer use them to write *processes*.
2. Define deployment criteria to transform processes into systems.
3. Define tests on systems.

Our Proposal: Processes, Systems and Tests

1. Define processes.
    1. Choose the *construction operators*. (= All operators)
    2. The programmer use them to write *processes*.
2. Define deployment criteria to transform processes into systems.
3. Define tests on systems.

Our Proposal: Processes, Systems and Tests

1. Define processes.

2. Define deployment criteria to transform processes into systems.
    1. Binding of variables
    2. The construction operators at top-level
    3. *Deployment operators*

   A system is *complete*.

3. Define tests on systems.

Our Proposal: Processes, Systems and Tests

1. Define processes.

2. Define deployment criteria to transform processes into systems.
    1. Binding of variables (= restriction)
    2. The construction operators at top-level (= no unguarded sum at top level)
    3. *Deployment operators* (it could be more general, and restricts, expands or intersects w/ construction operators)

   A system is *complete*.

3. Define tests on systems.

Our Proposal: Processes, Systems and Tests

1. Define processes.

2. Define deployment criteria to transform processes into systems.

3. Define tests on systems.
    1. Observables, i.e. functions from systems to a subset of a set of atomic proposition (="emits barb *a*", "terminates", "contains recursion operator", etc.),
    2. A notion of context, that should come with its own set of *testing operators* and reduction rules.

Our Proposal: Processes, Systems and Tests

1. Define processes.

2. Define deployment criteria to transform processes into systems.

3. Define tests on systems.
    1. Observables, i.e. functions from systems to a subset of a set of atomic proposition (=“emits barb *a*”, “terminates”, “contains recursion operator”, etc.),
    2. A notion of context, that should come with its own set of *testing operators* and reduction rules.

For instance: traditionally in π one may consider only *static contexts*.

Our Proposal: Processes, Systems and Tests

(Pre-) Congruences

are defined thanks to test as

“observationally,”

“contextually-closed,”

“reduction-closed[a]”,

relations.

[a] In presence of nondeterminism

Our Proposal: A Conceptual Clarification

Two (of the) Perks

1. Every step allows the introduction of operators,
2. Multiple notions of systems or tests can (and should!) co-exist in the same process algebra, one being targeted to e.g. programmers, and another to e.g. users.

No need to decide which set of observations is more basic.

Conclusion: On the Context Lemma

(Usual) Context Lemma

Given

fixed sets of processes, systems and observations,

two sets of testing contexts $S \subseteq S'$'

the observational congruences defined by the sets of context S and S' coincide.

Examples

In the π-calculus the context $a\langle M\rangle.[]$ does not change the congruence,

In the λ-calculus applicative contexts suffice.

Can we define a metatheorem?