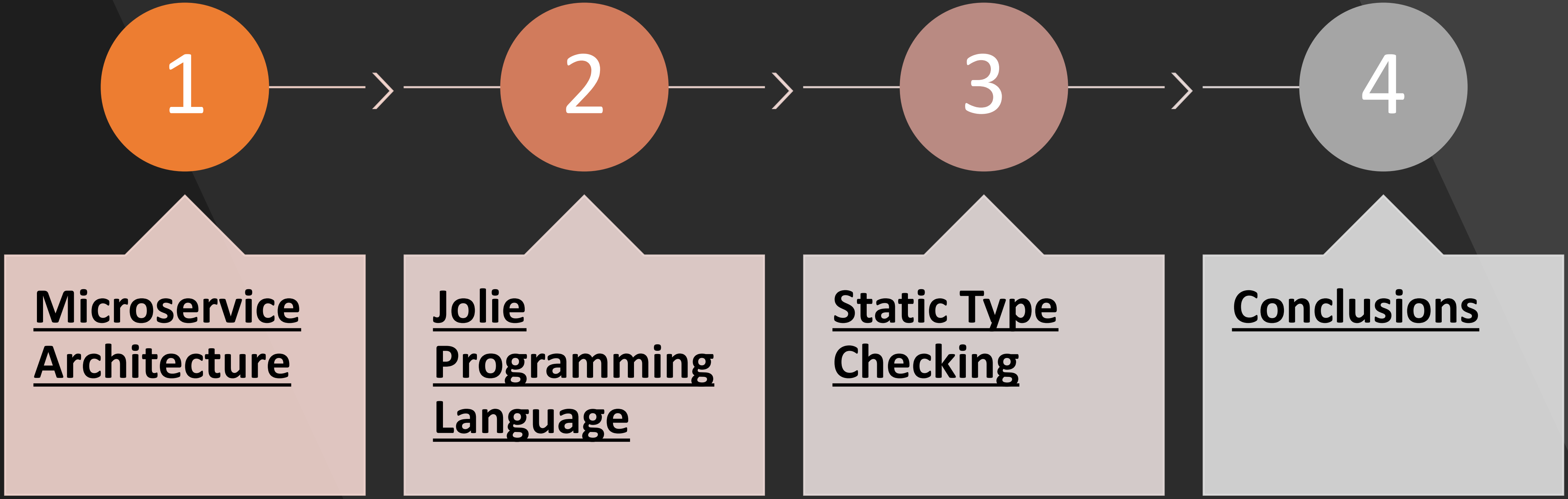
Towards Static Type-checking for Jolie

Bogdan Mingela, Nikolay Troshkov, Manuel Mazzara

Larisa Safina, Alexander Tchitchigin and Daniel De Carvalho

Eighth Workshop on Program Semantics, Specification and Verification: Theory and Applications - Moscow, June 26, 2017

Agenda



1 **Microservice Architecture**

2 **Jolie Programming Language**

3 **Static Type Checking**

4 **Conclusions**

1: Microservice Architecture

Microservices

Inspired by **SOA**

Developed around **business capabilities**

Each microservice implements a limited amount of functionality and runs its own process

Uses lightweight **communication** mechanisms

Supports pervasive distribution and scalability

What is a microservice?

- A very small service?
- How “small”?
- How do we measure size?
    - Line of codes
    - Size of executable
    - Number of classes (if OOP)
    - Number of modules
    - API
    - Size of team

What does this all mean?

- A Microservice is not just “a very small service”
- There is not a predefined “size limit” that define whether a service is a microservice or not
- Indeed “microservice” is a somehow misleading definition
  - Or better there is not definition at all, or not a unique one

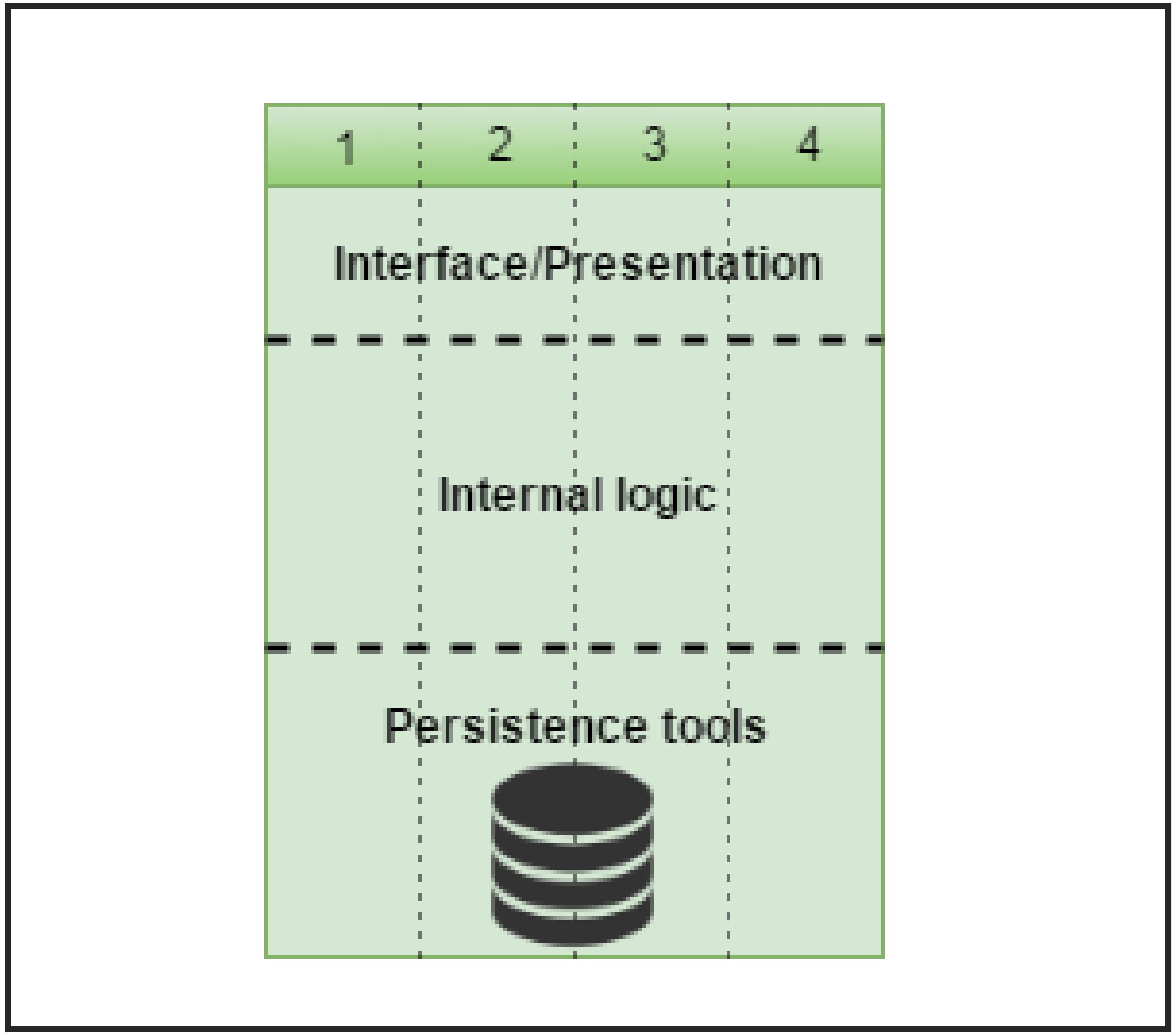
Microservice, definition

- **A microservice is a cohesive, independent process interacting via messages**

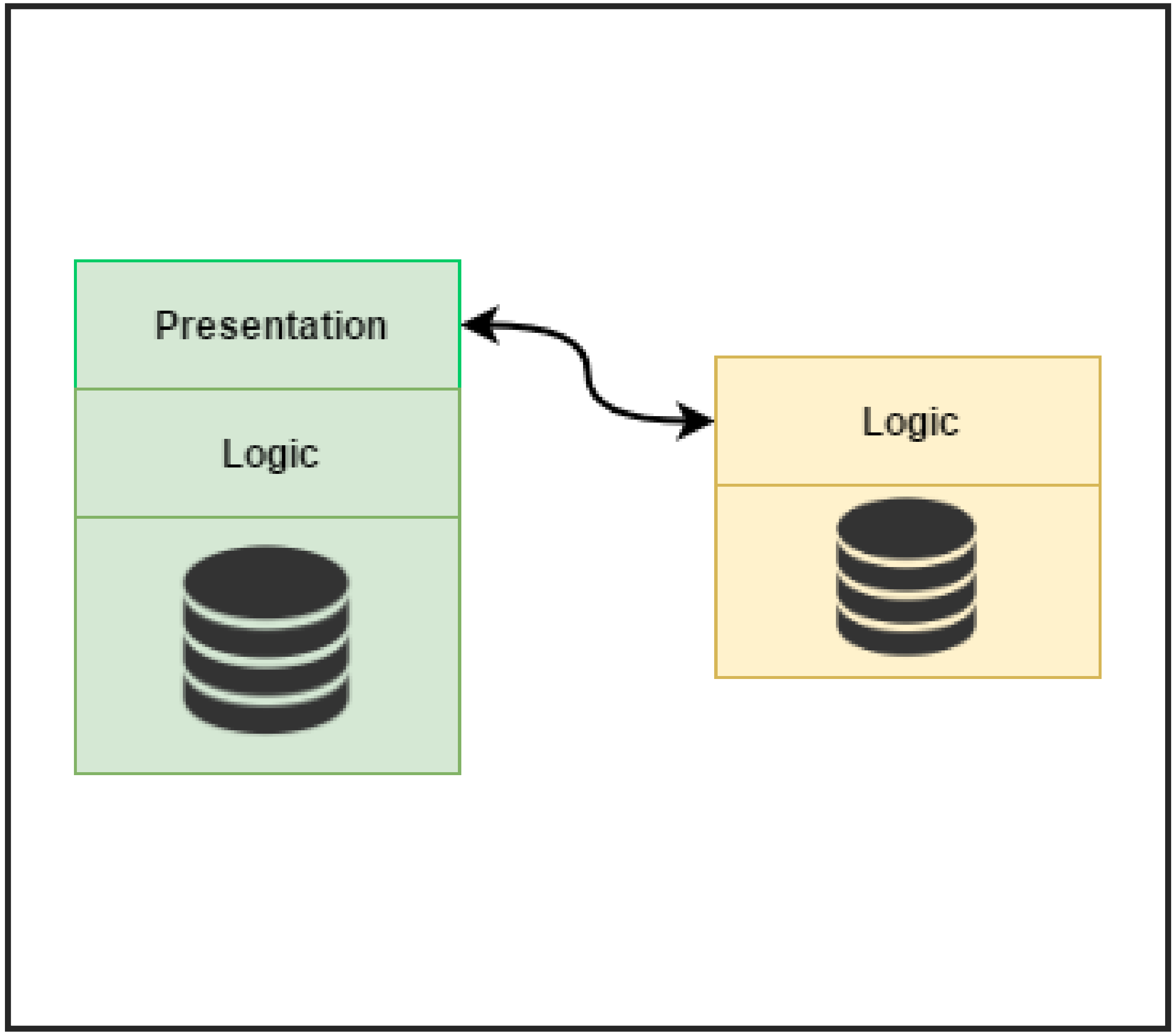
- **“Cohesive”** indicates that a service implements only functionalities strongly related to the concern that it is meant to model, this implies the code base to be functionally limited

- **“Micro”** refers to the sizing: a microservice must be manageable by a single development team (5-9 developers)

Monolith


1
2
3
4
Interface/Presentation
Internal logic
Persistence tools

Microservices



Distinctive Characteristics

- **Size** : The size is comparatively small wrt. a typical service
  - Focus on providing only a **single business capability**
  - Benefits in terms of service maintainability and extendibility

- **Bounded context :** related functionalities are combined into a single business capability, which is then implemented as a service

- **Independency** : Each service is operationally independent from other, and the only form of communication between services is through their published interfaces

Advantages of Microservices

- Smaller code base
  - Simpler to develop / test / deploy / scale
- Easier for new developers
  - Start faster
- Polyglot architecture
  - Each service may use individual technology
- Evolutionary design
  - Remove, add, replace…

SOA vs. Microservices

- In SOA Services are not required to be **self-contained** with data and UI
  - No focus on **independent deployment** units and its consequences

- Focused on enabling **business-level programming** through **business processing engines** and languages such as BPEL and BPMN

- **Service orchestration**

2: Jolie Programming Language

Language-based

- The fine granularity of microservices moves the complexity of applications from the implementation of services to their **coordination**

- **Communication, interfaces, and dependencies are central to the development of microservice applications**

- Such concepts should be available as ***first class entities*** in a language that targets microservices

Programming language for microservices

- Four concepts are identified to be ***first class entities*** in a programming language for microservices
  - **Interfaces**
  - **Ports**
  - **Workflows**
  - **Processes**

- **Jolie** (Java Orchestration Language Interpreter Engine) includes all of them

Jolie Programming Language

- A language for microservice
  - Imperative with standard constructs such as **assignments**, **conditionals** and **loops**
  - Constructs dealing with distribution, communication and services
  - Variables are trees to for easy marshal/unmarshal (XML)
  - Separation of concerns between **behaviour and deployment** information

- Jolie takes inspiration from **WS-BPEL** and **CCS**
  - transfers these ideas into a full-fledged programming language

Innopolis and the community

Jolie has a broad community of both industrial and academic partners

- Denmark, Russia, Italy, UK, France
- http://www.jolie-lang.org/academia.html

Innopolis is a full partner of the project

- We contributed on the development of the language itself, the type system, a static type checker and IDE

3: Static Type Checking

Static type checking

Effective technique of program verification

Identify bugs on the level of compilation

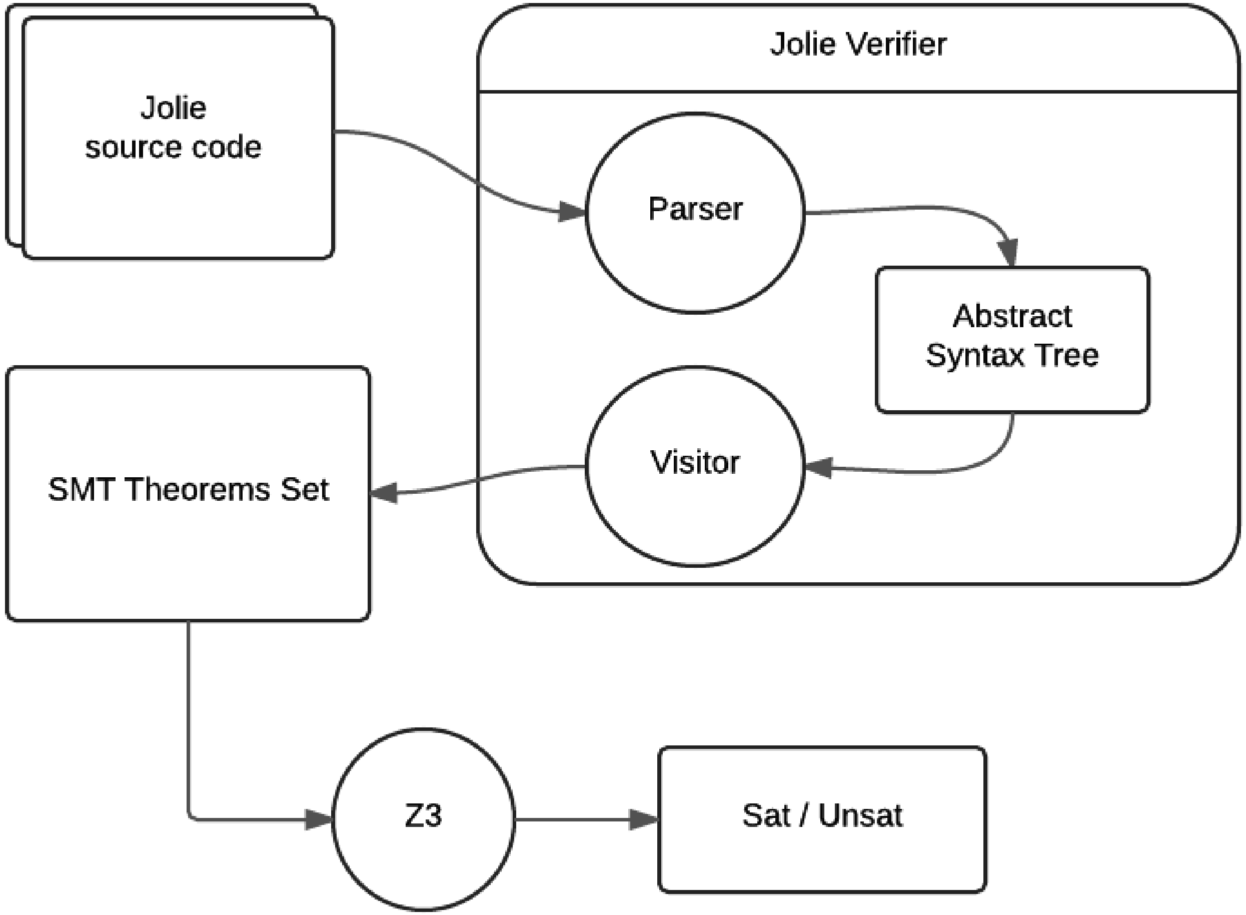
Improve software quality and lower number of bugs

Preventing avoidable errors

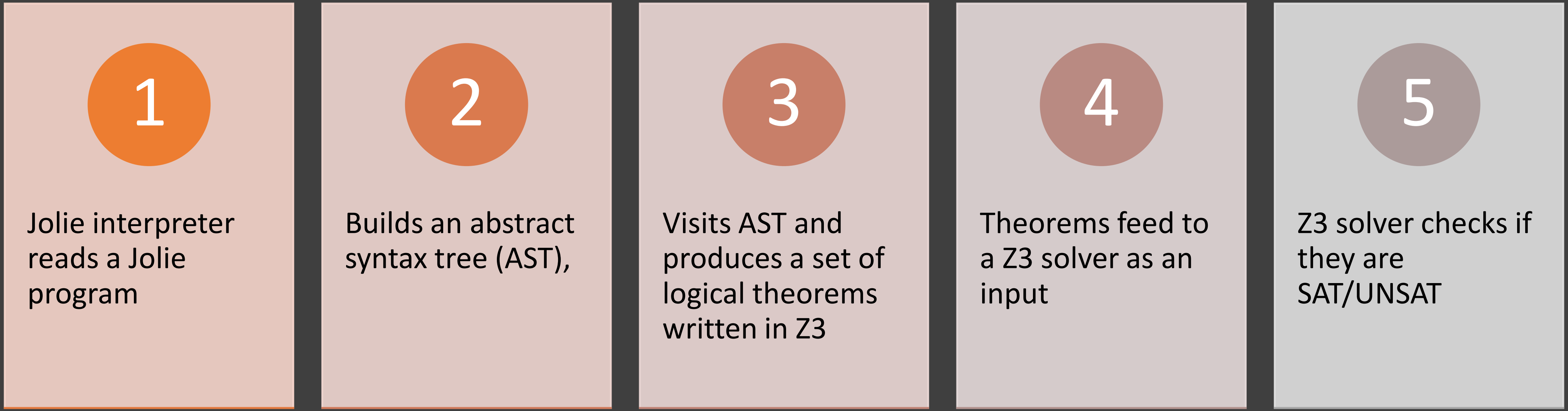
Jolie type checker

- At the moment the language is dynamically typed
- Static Type system has been formally defined
  - “A Type System for the Jolie Language” by J. Nielsen
- Prototype implemented for the core fragment of the Jolie language
  - excluding recursive types, arrays, subtyping, faults and deployment instructions

Jolie type checker architecture



Jolie type checker implementation



1. Jolie interpreter reads a Jolie program

2. Builds an abstract syntax tree (AST),

3. Visits AST and produces a set of logical theorems written in Z3

4. Theorems feed to a Z3 solver as an input

5. Z3 solver checks if they are SAT/UNSAT

Notation

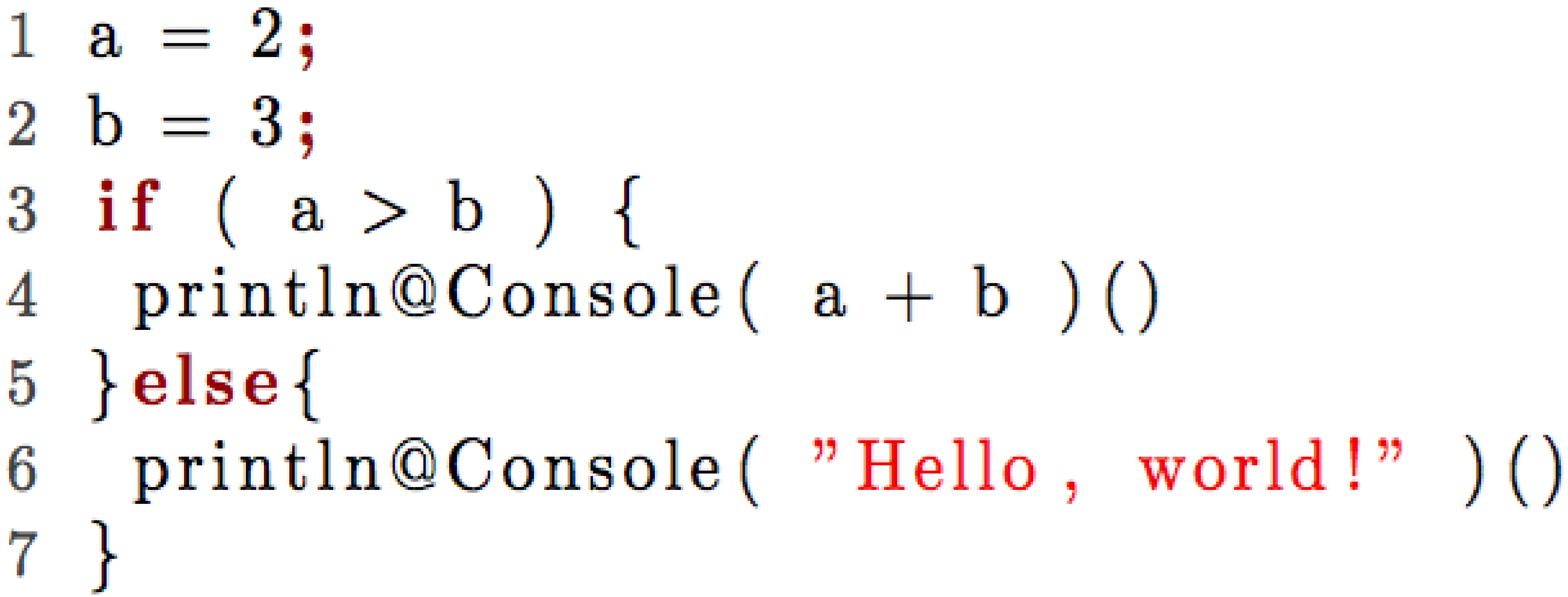
$$\Gamma \vdash_B B \triangleright \Gamma'$$

- A behaviour (program) B, typed with respect to an environment Γ, updates Γ to Γ’

Example: typing rule of IF

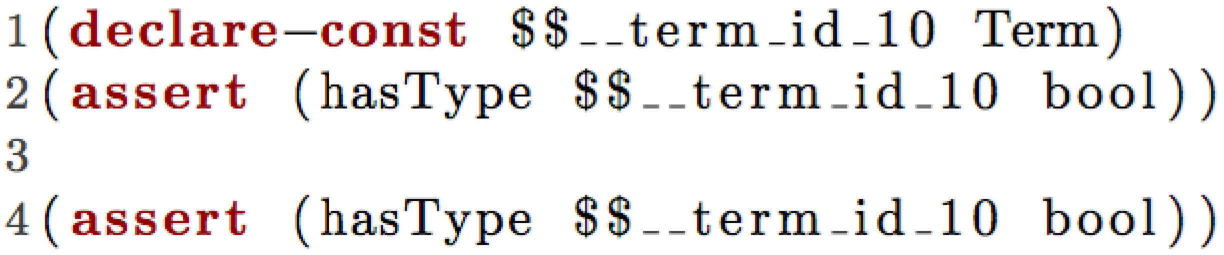
$$\frac{\Gamma \ \vdash \ e : bool \quad \Gamma \ \vdash_B B_1 \triangleright \Gamma' \quad \Gamma \ \vdash_B B_2 \triangleright \Gamma'}{\Gamma \ \vdash_B if(e)\ B_1\ else\ B_2 \triangleright \Gamma'}$$

Example of IF statement (correctly-typed)



```
a = 2;
b = 3;
if ( a > b ) {
  println@Console( a + b )()
}else{
  println@Console( "Hello, world!" )()
}
```

Z3 code

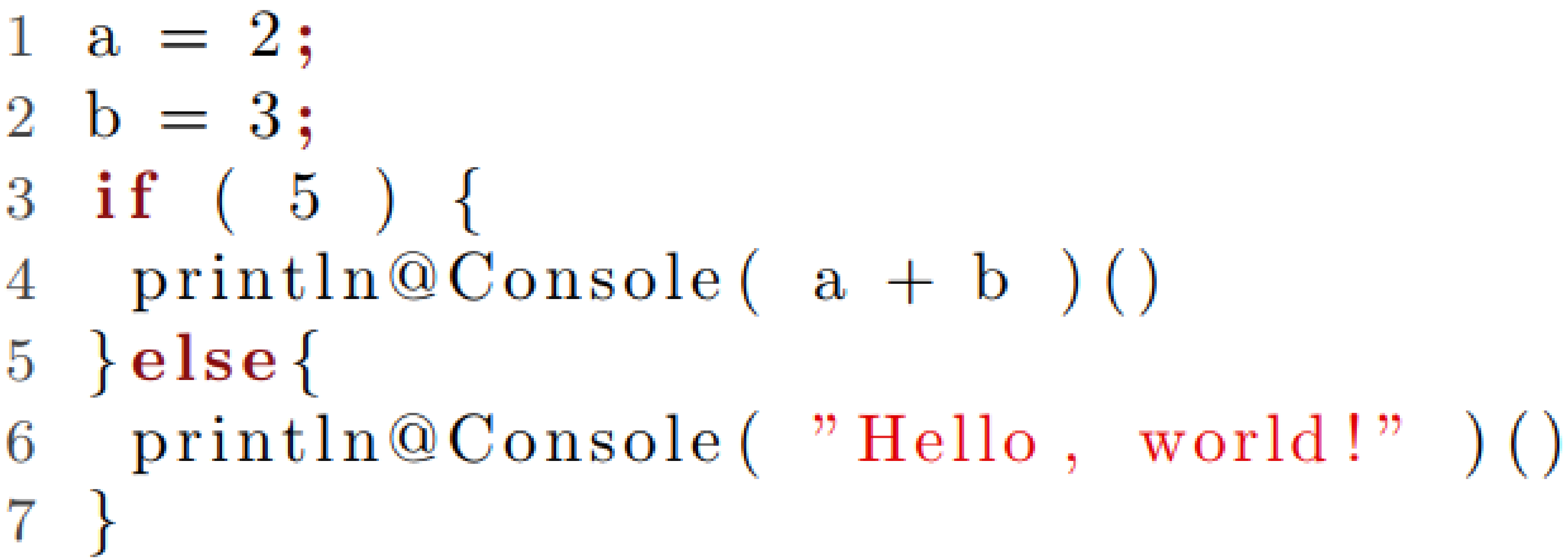


```
(declare-const $$__term_id_10 Term)
(assert (hasType $$__term_id_10 bool))

(assert (hasType $$__term_id_10 bool))
```

SAT

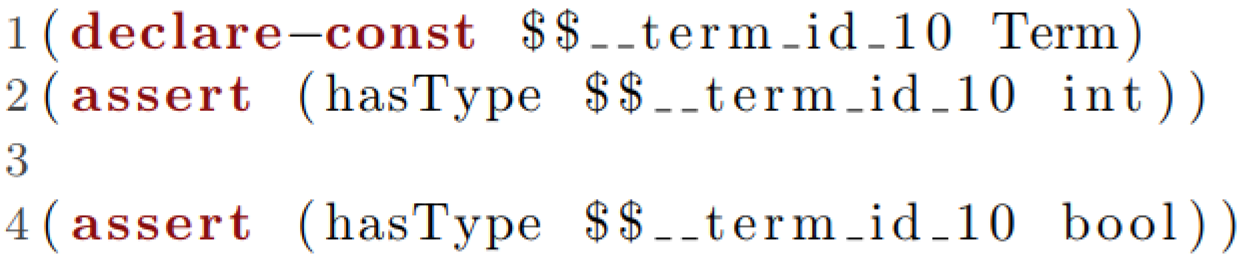
Example of IF statement (non correctly-typed)



```
a = 2;
b = 3;
if ( 5 ) {
  println@Console( a + b )()
}else{
  println@Console( "Hello, world!" )()
}
```

Z3 code →

UNSAT



```
(declare-const $$__term_id_10 Term)
(assert (hasType $$__term_id_10 int))

(assert (hasType $$__term_id_10 bool))
```

4: Conclusions

Microservices, summary

1

Microservices architecture is **more complex** than one based on monoliths

- The cost of growing and scaling easily

2

Companies of considerable size migrated their mission critical systems (of considerable size) into the new architectural style

- (not so) “Early” understanding of how critical scalability is

3

A language-based approach seems the best choice to cope with related challenges (not a new idea though)

Jolie, summary

- Native support for scalability and reusability
- **Communication** mediums and protocols support
- Structured **workflows**
- Reliable **parallel** coding
- **Formal** specifications
- Used both in **academia and industry**

Additional References

1. *N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina -* **Microservices: yesterday, today, and tomorrow**. In *Present and Ulterior Software Engineering*, Springer, 2017

2. *N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina.* ***Microservices: How to make your application scale.*** In *PSI 11th edition. Springer, 2017*

3. *N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara -* **Microservices: Migration of a mission critical system.** Technical report April 2017

   https://arxiv.org/abs/1704.04173

4. *Manuel Mazzara Fabrizio Montesi Claudio Guidi, Ivan Lanese.* **Microservices: a language-based approach.** In *Present and Ulterior Software Engineering.* Springer, 2017

5. *Alexey Bandura, Nikita Kurilenko, Manuel Mazzara, Victor Rivera, Larisa Safina, Alexander Tchitchigin -* ***Jolie Community on the Rise****.* 9th IEEE International Conference on Service-Oriented Computing, 2016

6. *Larisa Safina, Manuel Mazzara, Fabrizio Montesi, Victor Rivera -* ***Data-Driven Workflows for Microservices: Genericity in Jolie.*** 30th IEEE International Conference on Advanced Information Networking and Applications, 2016*.*

7. *Alexander Tchitchigin, Larisa Safina, Manuel Mazzara, Mohamed Elwakil, Fabrizio Montesi, Victor Rivera.* **Refinement types in jolie.** In Spring/Summer Young Researchers Colloquium on Software Engineering, SYRCoSE, 2016.