Ensuring Liveness Properties of Distributed Systems

(A Research Agenda)

Rob van Glabbeek

Data61, CSIRO, Sydney, Australia

University of New South Wales, Sydney, Australia

September 2016

Distributed Systems

systems consisting of multiple components that interact with each other

- distributed databases
- communication networks
- many operating systems
- industrial control systems
- a bank with a network of teller machines
- a group of people organising a workshop
- workflow of a publisher
- an airline
- etc., etc.

Harder to think of anything that isn't a distributed system.

How to ensure their correct working?

Formal methods

How to ensure their correct working?

Formal methods

- Specification formalisms
  - for the intended requirements of a distributed system

    what should it accomplish?

    e.g. Temporal Logic (LTL, CTL)

How to ensure their correct working?

Formal methods

- Specification formalisms
  - for the intended requirements of a distributed system

    what should it accomplish?

    e.g. Temporal Logic (LTL, CTL)
  - for the intended behaviour of a distributed system

    how does it do that?

    e.g. Process algebra, Petri nets

How to ensure their correct working?

Formal methods

- Specification formalisms
  - for the intended requirements of a distributed system

    what should it accomplish?

    e.g. Temporal Logic (LTL, CTL)
  - for the intended behaviour of a distributed system

    how does it do that?

    e.g. Process algebra, Petri nets
- Tools and analysis methods to study and reason about vital properties of the system

  e.g. (statistical) model checking

How to ensure their correct working?

Formal methods

- Specification formalisms
  - for the intended requirements of a distributed system

    what should it accomplish?

    e.g. Temporal Logic (LTL, CTL)
  - for the intended behaviour of a distributed system

    how does it do that?

    e.g. Process algebra, Petri nets
- Tools and analysis methods to study and reason about vital properties of the system

  e.g. (statistical) model checking
- Mathematically rigorous methods to verify that
  1. a system specification ensures the required properties
  2. an implementation meets the specification.

  Formal proofs - manual, automatic or interactive

How to ensure their correct working?

- Formal methods
- Alternatives

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

- Tools and analysis methods to study and reason about vital properties of the system
  - Simulation
  - Test-bed experiments

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

- Tools and analysis methods to study and reason about vital properties of the system
  - Simulation
  - Test-bed experiments
  - *important and valid methods for system evaluation*

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

- Tools and analysis methods to study and reason about vital properties of the system
  - Simulation
  - Test-bed experiments
  - *important and valid methods for system evaluation*
  - *resource-intensive and time-consuming*

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

- Tools and analysis methods to study and reason about vital properties of the system
  - Simulation
  - Test-bed experiments
  - *important and valid methods for system evaluation*
  - *resource-intensive and time-consuming*
  - *No general guarantees about correct system behaviour for a wide range of unpredictable deployment scenarios.*

How to ensure their correct working?

Formal methods

Alternatives

- Specification methods

  descriptions in English or other natural languages

  riddled with ambiguities, contradictions and under-specification.

- Tools and analysis methods to study and reason about vital properties of the system
  - Simulation
  - Test-bed experiments
  - *important and valid methods for system evaluation*
  - *resource-intensive and time-consuming*
  - *No general guarantees about correct system behaviour for a wide range of unpredictable deployment scenarios.*
- Verification None

Process algebra and related formalisms

- algebraic languages for the specification of processes
- algebraic laws to reason about processes
- induction principles to derive behaviours of infinite systems

Mayor toolsets that have been successfully applied to the specification and verification of industrial size distributed systems:

		Equivalence checking Refinement	Model checking	Other
CADP	INRIA	✓	✓	
mCRL2	Eindhoven	✓	✓	
FDR	Oxford	✓	✓	
TLA	Microsoft			✓
SPIN	Bell Labs		✓	
UPPAAL	Aalborg & Uppsala		✓	
PRISM	Oxford		✓	
Psi-calculi workbench	Uppsala	✓	✓	

LOTOS, XEludo, PSF, Concurrency workbench, FC2tools, Xeve, Circal, XVersa, SMV, NuSMV, XMC, Murϕ, COSPAN, STeP, Kronos, SGM.

Crucial correctness properties of systems

Safety

Something bad will never happen.

Liveness

Something good will happen eventually.

Crucial correctness properties of systems

Safety

Something bad will never happen.

Liveness

Something good will happen eventually.

Whether a liveness property holds often depends on underlying fairness assumptions one chooses to make.

Fairness assumptions

Weak fairness

If a task, from some point onwards, is perpetually enabled (meaning in each state) it will eventually be scheduled.

Strong fairness

If a task is enabled infinitely often, but allowing interruptions during which it is not enabled, it will eventually be scheduled.

Fairness

Strong or weak fairness can be

- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

Fairness

Strong or weak fairness can be

- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

- could be a spec. of a mobile phone
  - b is a successful dialing attempt
  - a is an unsuccessful dialing attempt.

  Fairness amounts to saying that if you try often enough, dialing will succeed.

Fairness

Strong or weak fairness can be

- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

- could be a spec. of a mobile phone
  - b is a successful dialing attempt
  - a is an unsuccessful dialing attempt.

  Fairness amounts to saying that if you try often enough, dialing will succeed.
- This is wishful thinking.
  The real world is not fair.

Crucial question

How to ensure liveness properties of distributed system without (weak or strong) fairness assumptions?

Incompatibility of bisimulation equivalence and liveness

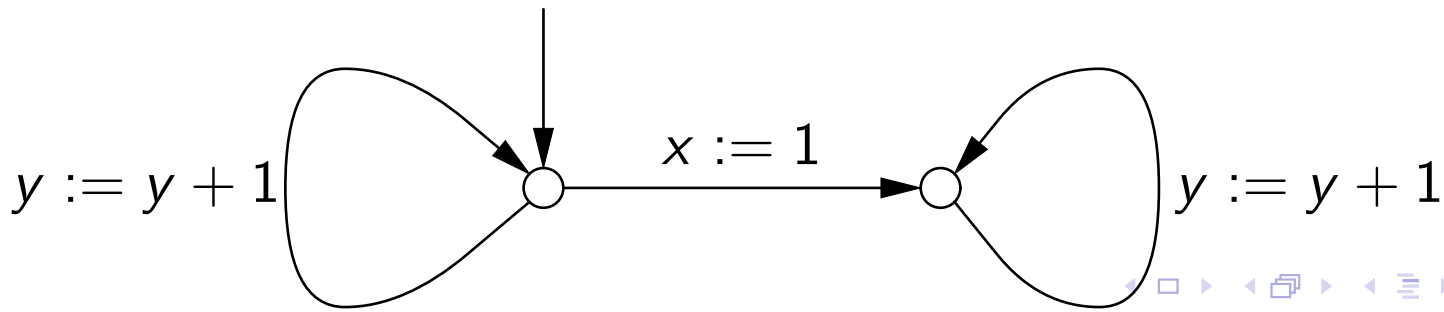
$$x := 1 \quad \| \quad \textbf{repeat} \;\; y := y + 1 \;\; \textbf{forever} \qquad (P)$$

Program P is the parallel composition of two non-interacting processes, one of which sets the variable x to 1, and the other repeatedly increments a variable y. I assume that both variables x and y are initialised to 0.

(Q)

repeat
  case
    if `True` **then** y:=y+1 **fi**
    if $x = 0$ **then** x:=1 **fi**
  end
forever

The programs P and Q are strongly bisimilar; both can be represented by means of the following labelled transition system:



Incompatibility of bisimulation equivalence and liveness

$$P \models \mathbf{AF}(y = 7) \quad ?$$

A:

$$Q \models \mathbf{AF}(y = 7) \quad ?$$

A:

$$Q \models \mathbf{AF}(x = 1) \quad ?$$

A:

$$P \models \mathbf{AF}(x = 1) \quad ?$$

A:

Incompatibility of bisimulation equivalence and liveness

$$P \models \mathbf{AF}(y = 7) \quad ?$$

A: Only when assuming *progress*.

$$Q \models \mathbf{AF}(y = 7) \quad ?$$

A:

$$Q \models \mathbf{AF}(x = 1) \quad ?$$

A:

$$P \models \mathbf{AF}(x = 1) \quad ?$$

A:

Incompatibility of bisimulation equivalence and liveness

$$P \models \mathbf{AF}(y = 7) \quad ?$$

A: Only when assuming *progress*.

$$Q \models \mathbf{AF}(y = 7) \quad ?$$

A: Same answer.

$$Q \models \mathbf{AF}(x = 1) \quad ?$$

A:

$$P \models \mathbf{AF}(x = 1) \quad ?$$

A:

Incompatibility of bisimulation equivalence and liveness

$$P \models \mathbf{AF}(y = 7) \quad ?$$

A: Only when assuming *progress*.

$$Q \models \mathbf{AF}(y = 7) \quad ?$$

A: Same answer.

$$Q \models \mathbf{AF}(x = 1) \quad ?$$

A: Only when assuming *(strong or weak) fairness*.

$$P \models \mathbf{AF}(x = 1) \quad ?$$

A:

Incompatibility of bisimulation equivalence and liveness

$$P \models \mathbf{AF}(y = 7) \quad ?$$

A: Only when assuming *progress*.

$$Q \models \mathbf{AF}(y = 7) \quad ?$$

A: Same answer.

$$Q \models \mathbf{AF}(x = 1) \quad ?$$

A: Only when assuming *(strong or weak) fairness*.

$$P \models \mathbf{AF}(x = 1) \quad ?$$

A: Yes, when assuming *justness* (or *strong progress*).

A challange

I am mostly interested in the wide class of applications where it is reasonable to assume justness, but not (weak or strong) fairness.

Here we saw two strongly bisimilar systems (P and Q) of which one satisfies a crucial liveness property and the other does not.

Hence we need a conceptual framework that is able to distinguish bisimilar systems when assessing liveness properties.

Contemporary process algebras and temporal logics are not suited for this task.