A *Formal Methods* Approach Towards Trustworthy Global Computing

Diego Latella

Consiglio Nazionale delle Ricerche
Ist. di Scienza e Tecnologie dell'Informazione "A. Faedo"
Formal Methods && Tools Lab

Bridging Global Computing with Grid
November 28-29, 2006

Tribute

Many concepts and ideas presented here are based on results on formal modeling and analysis of stochastic aspects of system behaviour achieved in the last few years by many colleagues and friends, a.o.:

- C. Baier et al. (Tec. Univ. of Dresden, D);
- M. Bernardo et al. (Univ. of Urbino, IT);
- R. Gorrieri et al. (Univ. of Bologna, IT);
- B. Haverkort et al. (Univ. of Twente, NL);
- H. Hermanns (Univ. of Saarbruecken, D);
- J. Hillston et al. (Univ. of Edinburg, UK);
- J.P. Katoen et al. (Univ. of Aachen, D);
- M. Kwiatkowska et al. (Univ. of Birmingham, UK);
- . . . and many others!

AGILE and SENSORIA

The focus on GC and SOC is the subject of cooperative work in the context of the EU Projects

AGILE and SENSORIA

Many thanks to:

- R. De Nicola (Univ. Firenze)
- J. P. Katoen (Univ. Aachen)
- M. Loreti (Univ. Firenze)
- M. Massink (CNR/ISTI, Pisa)

Outline

1. Background;

Outline

1. Background;
2. Challenges;

Outline

1. Background;
2. Challenges;
3. (Approaches to) Solutions and Opportunities;

Outline

1. Background;
2. Challenges;
3. (Approaches to) Solutions and Opportunities;
4. Open Issues;

Background - Formal Methods

"All engineering disciplines make progress by employing mathematically based notations and methods. Research on 'formal methods' follows this model and attempts to identify and develop mathematical approaches that can contribute to the task of creating computer systems"

[C. Jones 2000]

Background - Formal Methods

"All engineering disciplines make progress by employing mathematically based notations and methods. Research on 'formal methods' follows this model and attempts to identify and develop mathematical approaches that can contribute to the task of creating computer systems"

[C. Jones 2000]

Attempt to provide the (software) engineer with "concepts and techniques as thinking tools, which are clean, adequate, and convenient, to support him (or her) in describing, reasoning about, and constructing complex software and hardware systems"

[W. Thomas 2000]

Background - Formal Methods

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

Background - Formal Methods

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

For Supporting System Engineering

Background - Formal Methods

Applying $\left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\}$ Computer Science

For Supporting System Engineering

Emphasis on

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

For Supporting System Engineering

Emphasis on

- Construction

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

For Supporting System Engineering

Emphasis on

- Construction
- Pragmatics

Background - Formal Methods

$$\text{Applying}\left\{\frac{\text{Logic in}}{\text{Theoretical}}\right\}\text{Computer Science}$$

For Supporting System Engineering

Emphasis on

- Construction
- Pragmatics
- Automatic, often push-botton, Software Tool Support

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

For Supporting System Engineering

Emphasis on

- Construction
- Pragmatics
- Automatic, often push-botton, Software Tool Support

rather than

Background - Formal Methods

$$\text{Applying} \left\{ \frac{\text{Logic in}}{\text{Theoretical}} \right\} \text{Computer Science}$$

For Supporting System Engineering

Emphasis on

- Construction
- Pragmatics
- Automatic, often push-botton, Software Tool Support

rather than

- classical issues like completeness.

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.

Complex control software for space applications
(e.g. NASA Mars rovers [COMPUTER, Jan. 04])

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

- Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.
  - Complex control software for space applications (e.g. NASA Mars rovers [COMPUTER, Jan. 04])
  - Automotive & Railways (e.g. train interlocking & on board control systems)

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

- Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.
  - Complex control software for space applications (e.g. NASA Mars rovers [COMPUTER, Jan. 04])
  - Automotive & Railways (e.g. train interlocking & on board control systems)
  - Low level device control (Microsoft SLAM Project)

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

- Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.
  - Complex control software for space applications (e.g. NASA Mars rovers [COMPUTER, Jan. 04])
  - Automotive & Railways (e.g. train interlocking & on board control systems)
  - Low level device control (Microsoft SLAM Project)
  - ...

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

- Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.
  - Complex control software for space applications (e.g. NASA Mars rovers [COMPUTER, Jan. 04])
  - Automotive & Railways (e.g. train interlocking & on board control systems)
  - Low level device control (Microsoft SLAM Project)
  - . . .
- (Automatic) Theorem Proving, e.g.
  - Avionics systems (e.g. Boeing)
  - ...

Background - Success stories

Classical FM have been successfully used for modeling and analyzing functional aspects of complex systems, for example:

- Model-checking models of *trillions* of states (or more ... 10^{30}), e.g.
  - Complex control software for space applications (e.g. NASA Mars rovers [COMPUTER, Jan. 04])
  - Automotive & Railways (e.g. train interlocking & on board control systems)
  - Low level device control (Microsoft SLAM Project)
  - . . .
- (Automatic) Theorem Proving, e.g.
  - Avionics systems (e.g. Boeing)
  - ...

Background - Success stories

Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability.

Background - Success stories

Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability.

Bill Gates, April 18, 2002.
Keynote address at WinHEC 2002

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages e.g.

Timed/Probabilistic/Stochastic/Security-Oriented Process Calculi

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages e.g.

  Timed/Probabilistic/Stochastic/Security-Oriented Process Calculi

  with solid formal semantics and equipped with mathematical theories for sound model manipulation (e.g. state-space minimization software tools)

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages e.g.

  Timed/Probabilistic/Stochastic/Security-Oriented Process Calculi

  with solid formal semantics and equipped with mathematical theories for sound model manipulation (e.g. state-space minimization software tools)
- High level (non-)functional requirement specification languages,

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages e.g.

  Timed/Probabilistic/Stochastic/Security-Oriented Process Calculi

  with solid formal semantics and equipped with mathematical theories for sound model manipulation (e.g. state-space minimization software tools)

- High level (non-)functional requirement specification languages, e.g.

  Timed/Probabilistic/Stochastic/Security-Oriented Temporal Logics

Background - Extensions of FM

A substantial contribution to the design of dependable systems can be provided by extensions of FM for the integrated modeling and analysis of functional and non-functional aspects of complex systems, e.g.

- High level model specification languages e.g.

Timed/Probabilistic/Stochastic/Security-Oriented Process Calculi

  with solid formal semantics and equipped with mathematical theories for sound model manipulation (e.g. state-space minimization software tools)

- High level (non-)functional requirement specification languages, e.g.

Timed/Probabilistic/Stochastic/Security-Oriented Temporal Logics

  based on the solid framework of Mathematical Logic and equipped with efficient decision procedures (e.g. stochastic model-checkers)

Challenges

- Rigorous modeling of the behaviour of systems characterized by

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty
  (e.g. due to size, e.g. on latency);

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty
  (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;

Challenges

- Rigorous modeling of the behaviour of systems characterized by
    - Parallelism
    - Distribution
    - Distribution awareness
    - Mobility
    - Performance and Dependability
    - Security
    - ...

  while coping with partial knowledge and uncertainty (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;
- Automatic tool support for modeling and reasoning;

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;
- Automatic tool support for modeling and reasoning;
  - Access to modeling and reasoning via engineering notations and tools;

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;
- Automatic tool support for modeling and reasoning;
  - Access to modeling and reasoning via engineering notations and tools;
  - Link to implementation and deployment;

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty
  (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;
- Automatic tool support for modeling and reasoning;
  - Access to modeling and reasoning via engineering notations and tools;
  - Link to implementation and deployment;
- Scalability

Challenges

- Rigorous modeling of the behaviour of systems characterized by
  - Parallelism
  - Distribution
  - Distribution awareness
  - Mobility
  - Performance and Dependability
  - Security
  - ...

  while coping with partial knowledge and uncertainty
  (e.g. due to size, e.g. on latency);
- Rigorous reasoning about such models;
- Automatic tool support for modeling and reasoning;
  - Access to modeling and reasoning via engineering notations and tools;
  - Link to implementation and deployment;
- Scalability

- Process (Algebra-like) Languages for modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution),

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to
    - download or read resources from

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to
    - download or read resources from
    - spawn processes (or move) to

Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to
    - download or read resources from
    - spawn processes (or move) to

    remote (as well as local) sites (distribution awareness and mobility);

Challenges, Solutions & Opportunities: Rigorous modeling

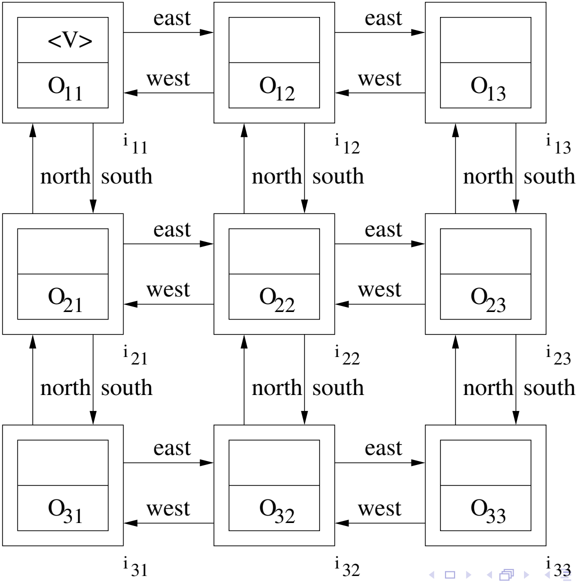
- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to
    - download or read resources from
    - spawn processes (or move) to

    remote (as well as local) sites (distribution awareness and mobility);
  - Resources which can be down-loaded using patter-matching mechanisms (discovery, SLA);

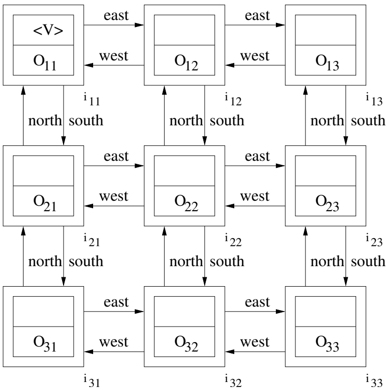
Challenges, Solutions & Opportunities: Rigorous modeling

- Process (Algebra-like) Languages for modeling
  - Networks composed of different sites (distribution), where
    - Processes are running;
    - Resources (including code) are stored.
  - Processes which can
    - upload resources to
    - download or read resources from
    - spawn processes (or move) to

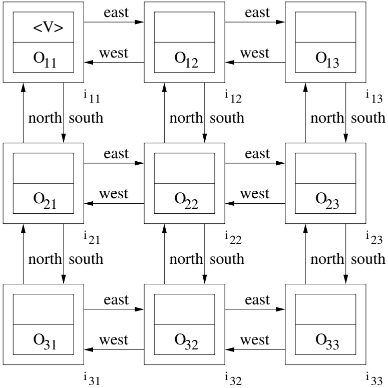
    remote (as well as local) sites (distribution awareness and mobility);
  - Resources which can be down-loaded using patter-matching mechanisms (discovery, SLA);
  - Uncertainty on operation execution times and/or possible choices by means of random variables and weights/probabilities (partial/approx. knowledge, performance/dependability analysis)



<V>
O11
O12
O13
O21
O22
O23
O31
O32
O33
east
west
north
south
i11
i12
i13
i21
i22
i23
i31
i32
i33

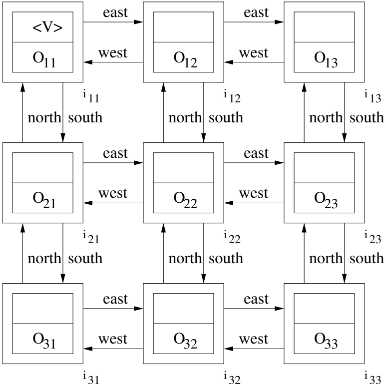


<V>
O_{11}
O_{12}
O_{13}
O_{21}
O_{22}
O_{23}
O_{31}
O_{32}
O_{33}
east
west
north
south
i_{11}
i_{12}
i_{13}
i_{21}
i_{22}
i_{23}
i_{31}
i_{32}
i_{33}



$$V \triangleq ((\mathbf{out}(V)@\mathit{north}, rn).\mathbf{nil}) + ((\mathbf{out}(V)@\mathit{south}, rs).\mathbf{nil}) + ((\mathbf{out}(V)@\mathit{east}, re).\mathbf{nil}) + ((\mathbf{out}(V)@\mathit{west}, rw).\mathbf{nil})$$

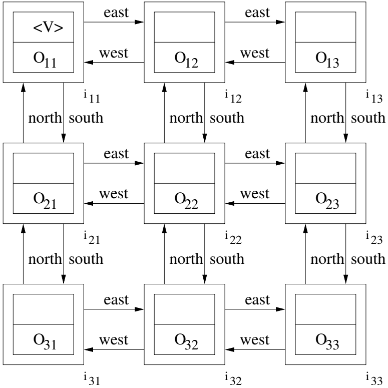
Challenges, Solutions & Opportunities: Rigorous modeling



$$V \triangleq ((\mathbf{out}(V)@north, rn).\mathbf{nil}) + ((\mathbf{out}(V)@south, rs).\mathbf{nil}) + ((\mathbf{out}(V)@east, re).\mathbf{nil}) + ((\mathbf{out}(V)@west, rw).\mathbf{nil})$$

$$O_{jk} \triangleq ((\mathbf{in}(!X)@\mathrm{self}, d_{jk}).O_{jk}) + ((\mathbf{in}(!X)@\mathrm{self}, u_{jk}).(\mathbf{eval}(X)@\mathrm{self}, r_{jk}).O_{jk})$$

Challenges, Solutions & Opportunities: Rigorous modeling

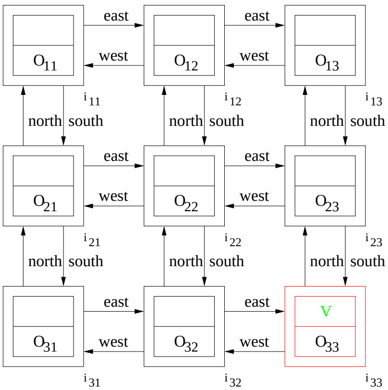


$$V \triangleq ((\mathbf{out}(V)@north, rn).\mathbf{nil}) + ((\mathbf{out}(V)@south, rs).\mathbf{nil}) + ((\mathbf{out}(V)@east, re).\mathbf{nil}) + ((\mathbf{out}(V)@west, rw).\mathbf{nil})$$

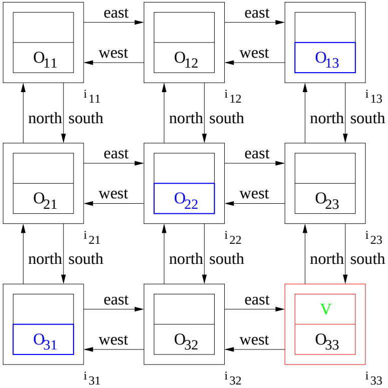
$$O_{jk} \triangleq ((\mathbf{in}(!X)@\mathsf{self}, d_{jk}).O_{jk}) + ((\mathbf{in}(!X)@\mathsf{self}, u_{jk}).(\mathbf{eval}(X)@\mathsf{self}, r_{jk}).O_{jk})$$

$$i_{11} ::_{\rho_{11}} \langle V \rangle \parallel \left(\big\|_{\substack{1 \leq j \leq 3 \\ 1 \leq k \leq 3}} i_{jk} ::_{\rho_{jk}} O_{jk} \right)$$

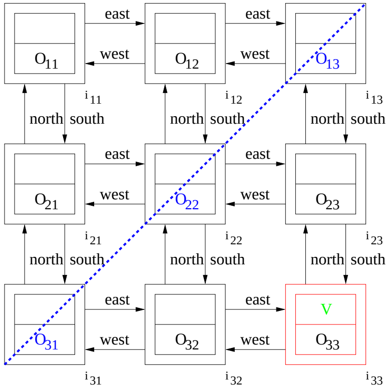
Challenges, Solutions & Opportunities: Rigorous reasoning



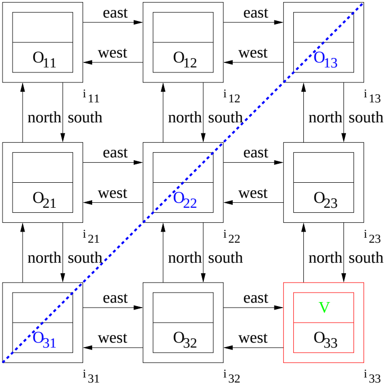
Challenges, Solutions & Opportunities: Rigorous reasoning



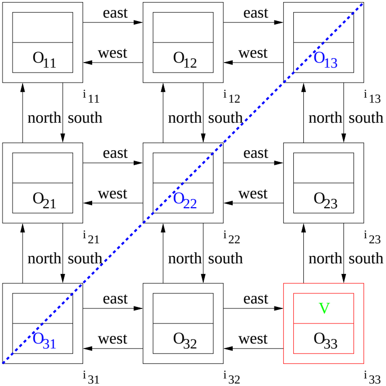
Challenges, Solutions & Opportunities: Rigorous reasoning



Challenges, Solutions & Opportunities: Rigorous reasoning



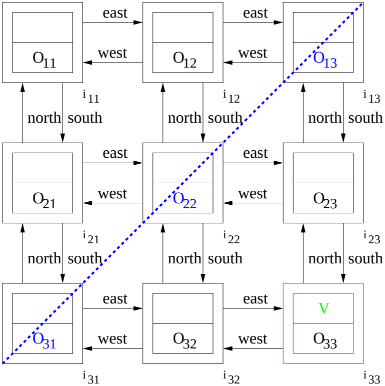
The probability is less then 0.2 that the infection develops (i.e. the virus is running) at site i_{33} by time t, assuming that at time 0 the only site infected is (i.e. the virus is stored at) i_{11}.



The probability is less then 0.2 that the infection develops (i.e. the virus is running) at site i_{33} by time t, assuming that at time 0 the only site infected is (i.e. the virus is stored at) i_{11}.

In MObile Stochastic Temporal Logic(extension of CSL [Baier et al.]):

$$\mathcal{P}_{\leq 0.2}(\neg V@i_{33} \;\; \mathcal{U}^{<t} \; V@i_{33})$$

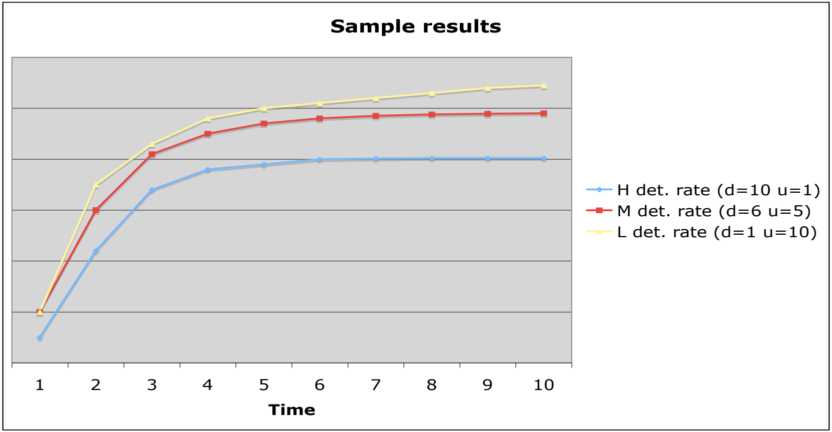


The probability is less then 0.2 that the infection develops (i.e. the virus is running) at site i_{33} by time t, assuming that at time 0 the only site infected is (i.e. the virus is stored at) i_{11}.

In MObile Stochastic Temporal Logic(extension of CSL [Baier et al.]):

$$\mathcal{P}_{\leq 0.2}(\neg V@i_{33}\ \ \mathcal{U}^{<t}\ V@i_{33})$$

... and by means of Automatic Stochastic Model-checking ...


Sample results
1
2
3
4
5
6
7
8
9
10
Time
H det. rate (d=10 u=1)
M det. rate (d=6 u=5)
L det. rate (d=1 u=10)

Logical characterization (and automatic verification) of

Steady State Probability measures

Logical characterization (and automatic verification) of

Steady State Probability measures

In the long term,

$$\mathcal{S} \quad (\qquad)$$

Challenges, Solutions & Opportunities: Rigorous reasoning

Logical characterization (and automatic verification) of

Steady State Probability measures

In the long term, the probability is greater than 0.9

$$\mathcal{S}_{>0.9}(\qquad)$$

Logical characterization (and automatic verification) of

Steady State Probability measures

In the long term, the probability is greater than 0.9 that value f is present at site i.

$$\mathcal{S}_{>0.9}(\langle f \rangle @ i)$$

Logical characterization (and automatic verification) of

Steady State Probability measures

In the long term, the probability is greater than 0.9 that value f is present at site i.

$$\mathcal{S}_{>0.9}(\langle f \rangle @ i)$$

Logical characterization (and automatic verification) of Steady State Probability measures

In the long term, the probability is greater than 0.9 that value f is present at site i.

$$\mathcal{S}_{>0.9}(\langle f \rangle @ i)$$

In the long term, the probability is smaller than 0.2 that process P is running at site i

$$\mathcal{S}_{<0.2}(P @ i)$$

Logical characterization (and automatic verification) of

Steady State Probability measures

In the long term, the probability is greater than 0.9 that value f is present at site i.

$$\mathcal{S}_{>0.9}(\langle f \rangle @ i)$$

In the long term, the probability is smaller than 0.2 that process P is running at site i

$$\mathcal{S}_{<0.2}(P @ i)$$

P could be a malicious (or faulty) process: the formula would characterize the average fraction of time the infection (or faulty component) is active at site i.

Transient probability measures

Transient probability measures

The probability is

$$\mathcal{P} \quad (\qquad\qquad)$$

Transient probability measures

The probability is at least 0.8

$$\mathcal{P}_{>0.8}(\qquad)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down

$$\mathcal{P}_{>0.8}(\quad \neg\, down)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t}\ \neg\, down)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\Diamond^{t} \neg \mathit{down})$$

Instantaneous availability at time t.

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t} \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t} \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

The probability is

$$\mathcal{P}\quad(\qquad\qquad)$$

Challenges, Solutions & Opportunities: Rigorous reasoning

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t}\ \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

The probability is less than 0.01

$$\mathcal{P}_{<0.01}(\qquad\qquad\qquad)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t} \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

The probability is less than 0.01 that the system goes down

$$\mathcal{P}_{<0.01}(\qquad\qquad down)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t}\ \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

The probability is less than 0.01 that the system goes down within time t

$$\mathcal{P}_{<0.01}(\qquad \mathcal{U}^{<t}\ down)$$

Transient probability measures

The probability is at least 0.8 that the system is *not* down at time t

$$\mathcal{P}_{>0.8}(\ \Diamond^{t} \neg\, down)$$

Instantaneous availability at time t.

More general path-based measures

The probability is less than 0.01 that the system goes down within time t without having first raised an alarm signal

$$\mathcal{P}_{<0.01}(\neg alarm\ \ \mathcal{U}^{<t}\ down)$$

Nested measures

Nested measures

In equilibrium, the probability is

$$\mathcal{S} \quad (\qquad\qquad\qquad\qquad)$$

Nested measures

In equilibrium, the probability is at least 0.87,

$$\mathcal{S}_{>0.87}(\qquad\qquad\qquad\qquad)$$

Nested measures

In equilibrium, the probability is at least 0.87, that

$$\mathcal{S}_{>0.87}(\qquad)$$

Nested measures

In equilibrium, the probability is at least 0.87, that
in at least 75% of the cases

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(\quad))$$

Nested measures

In equilibrium, the probability is at least 0.87, that
in at least 75% of the cases the system will have resumed

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(\qquad \neg down))$$

Challenges, Solutions & Opportunities: Rigorous reasoning

Nested measures

In equilibrium, the probability is at least 0.87, that in at least 75% of the cases the system will have resumed within 500 time units

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(\mathit{down}\ \ \mathcal{U}^{<500}\ \neg \mathit{down}))$$

Nested measures

In equilibrium, the probability is at least 0.87, that
in at least 75% of the cases the system will have resumed within 500 time units

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(down\ \ \mathcal{U}^{<500}\ \neg down))$$

CTL as a special case

Challenges, Solutions & Opportunities: Rigorous reasoning

Nested measures

In equilibrium, the probability is at least 0.87, that in at least 75% of the cases the system will have resumed within 500 time units

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(\mathit{down}\ \ \mathcal{U}^{<500}\ \neg \mathit{down}))$$

CTL as a special case

$$\mathsf{A}\ \varphi \equiv \mathcal{P}_{\geq 1}(\varphi)$$

Nested measures

In equilibrium, the probability is at least 0.87, that
in at least 75% of the cases the system will have resumed within 500 time units

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(\mathit{down}\ \ \mathcal{U}^{<500}\ \neg \mathit{down}))$$

CTL as a special case

$$\mathsf{A}\ \varphi \equiv \mathcal{P}_{\geq 1}(\varphi) \qquad \mathsf{E}\ \varphi \equiv \mathcal{P}_{>0}(\varphi)$$

Challenges, Solutions & Opportunities: Rigorous reasoning

Nested measures

In equilibrium, the probability is at least 0.87, that in at least 75% of the cases the system will have resumed within 500 time units

$$\mathcal{S}_{>0.87}(\mathcal{P}_{>0.75}(down\ \ \mathcal{U}^{<500}\ \neg down))$$

CTL as a special case

A $\varphi \equiv \mathcal{P}_{\geq 1}(\varphi)$

E $\varphi \equiv \mathcal{P}_{>0}(\varphi)$

Functional and non functional issues of behaviour integrated in the same description language

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators

  for state-transition models

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators

  for state-transition models specified for instance in stochastic process-algebra(-like) languages

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

- Access from engineering notations and tools:

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

- Access from engineering notations and tools:

  Translations from e.g. UML, BPEL4WS, GRID-Skeletons
  to such languages and logics;

  [e.g. EU-HIDE, EU-SENSORIA]

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

- Access from engineering notations and tools:

  Translations from e.g. UML, BPEL4WS, GRID-Skeletons to such languages and logics;

  [e.g. EU-HIDE, EU-SENSORIA]

- Link to implementation:

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

- Access from engineering notations and tools:

  Translations from e.g. UML, BPEL4WS, GRID-Skeletons to such languages and logics;

  [e.g. EU-HIDE, EU-SENSORIA]

- Link to implementation:

  Model-checking of software programs (automatic model extraction from real code)

  [e.g. Java Pathfinder, NASA]

Challenges, Solutions & Opportunities: Automatic Tool Support

- For example stochastic simulators and (stochastic) mode-checkers

  for state-transition models specified for instance in (stochastic) process-algebra(-like) languages and requirements characterized via (stochastic) temporal logics.

- Access from engineering notations and tools:

  Translations from e.g. UML, BPEL4WS, GRID-Skeletons to such languages and logics;

  [e.g. EU-HIDE, EU-SENSORIA]

- Link to implementation:

  Model-checking of software programs
  (automatic model extraction from real code)

  [e.g. Java Pathfinder, NASA]

Challenges, Open Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

Challenges, Open Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

Challenges, Open Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

Challenges, Open Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

Challenges, Open Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

Challenges, (Open) Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

BUT

Challenges, (Open) Issues: Scalability

- Analysis techniques based on current explicit state-space methods able to cope with sizes of the order of 10^7.

- Novel techniques and technologies are being developed

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools,

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools, e.g.

- Parallelization of (stochastic) model-checking algorithms,

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools, e.g.

- Parallelization of (stochastic) model-checking algorithms, e.g. of CTMC solution techniques

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools, e.g.

- Parallelization of (stochastic) model-checking algorithms, e.g. of CTMC solution techniques
- Using GRID architectures

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools, e.g.

- Parallelization of (stochastic) model-checking algorithms, e.g. of CTMC solution techniques
- Using GRID architectures

State-space size of the order of $10^9 - 10^{10}$

e.g. [Kwiatkowska et al. 05]

Challenges, (Open) Issues: Scalability

Distributed implementation of analysis tools, e.g.

- Parallelization of (stochastic) model-checking algorithms, e.g. of CTMC solution techniques
- Using GRID architectures

State-space size of the order of $10^9 - 10^{10}$

e.g. [Kwiatkowska et al. 05]

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state,

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
    - The fraction of components which are in a certain state, as a function of time.

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
    - The fraction of components which are in a certain state, as a function of time.
    - For systems with high total number of components (i.e. thousands or more)
- Examples:
    - Processor-Resources usage profiling
    - Warm attacks and sanitation profiling

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling
  - Warm attacks and sanitation profiling
  - Biochemical process analysis

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling
  - Warm attacks and sanitation profiling
  - Biochemical process analysis
  - GRID clusters on-line performability analysis (for efficient scheduling)

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling
  - Warm attacks and sanitation profiling
  - Biochemical process analysis
  - GRID clusters on-line performability analysis (for efficient scheduling)

State-space size of the order of $10^{10.000}$

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling
  - Warm attacks and sanitation profiling
  - Biochemical process analysis
  - GRID clusters on-line performability analysis (for efficient scheduling)

State-space size of the order of $10^{10.000}$

- Fully automatic translation from SPA specification to system of ODE.

[Hillston et al. 05]

Challenges, (Open) Issues: Scalability

Specification driven use of Ordinary Differential Equations

- Typical for studying issues like:
  - The fraction of components which are in a certain state, as a function of time.
  - For systems with high total number of components (i.e. thousands or more)
- Examples:
  - Processor-Resources usage profiling
  - Warm attacks and sanitation profiling
  - Biochemical process analysis
  - GRID clusters on-line performability analysis (for efficient scheduling)

State-space size of the order of $10^{10.000}$

- Fully automatic translation from SPA specification to system of ODE.

[Hillston et al. 05]

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals
- Time dependency

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals
- Time dependency (time-inhomogeneous stochastic models)

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.

- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals

- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals
- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve
- Modeling and analysis techniques capable of dealing both with randomness and with non-determinism (MDP)

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals
- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve
- Modeling and analysis techniques capable of dealing both with randomness and with non-determinism (MDP)

  e.g. non-determinism generated from an interleaving approach to modeling of parallelism

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.

- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals

- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve

- Modeling and analysis techniques capable of dealing both with randomness and with non-determinism (MDP)

  e.g. non-determinism generated from an interleaving approach to modeling of parallelism

- Run-time verification techniques complementing model-based techniques

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.
- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals
- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve
- Modeling and analysis techniques capable of dealing both with randomness and with non-determinism (MDP)

  e.g. non-determinism generated from an interleaving approach to modeling of parallelism
- Run-time verification techniques complementing model-based techniques

  e.g. Program model-checking, Probabilistic testing

General Open Issues

- Improving SMC diagnostic feedback

  counter-example generation, accuracy of result, etc.

- From Very-Large to Infinite state systems (CTMC)
- Continuous State Systems (CSDTMC)

  e.g. sampling of real-valued signals

- Time dependency (time-inhomogeneous stochastic models)

  e.g. hardware components failure rate typically follows a bathtub curve

- Modeling and analysis techniques capable of dealing both with randomness and with non-determinism (MDP)

  e.g. non-determinism generated from an interleaving approach to modeling of parallelism

- Run-time verification techniques complementing model-based techniques

  e.g. Program model-checking, Probabilistic testing

- Spatial, stochastic logics for mobility

Conclusion

"Software: the critical infrastructure *within* the Critical Infrastructures"

Conclusion

"Software: the critical infrastructure *within* the Critical Infrastructures"

[Theme of the
2nd National Software Summit hosted by
the Center for National Software Studies
Washington D.C. - May 10-12, 2004]

Conclusion

"Software: the critical infrastructure *within* the Critical Infrastructures"

[Theme of the
2nd National Software Summit hosted by
the Center for National Software Studies
Washington D.C. - May 10-12, 2004]

"[I]t is time for us, as a nation, to elevate software a to matter of national policy [...]"

Conclusion

"Software: the critical infrastructure *within* the Critical Infrastructures"

[Theme of the
2nd National Software Summit hosted by
the Center for National Software Studies
Washington D.C. - May 10-12, 2004]

"[I]t is time for us, as a nation, to elevate software a to matter of national policy [...]"

[SOFTWARE 2015: A NATIONAL SOFTWARE STRATEGY TO ENSURE U.S. SECURITY AND COMPETITIVENESS.
Report of the 2nd NSS - av. at www.cnsoftware.org]