Quantitative Analysis of KLAIM Nets

Michele Loreti

joint work with

Francesco Calzolai   Rocco De Nicola   Diego Latella   Mieke Massink

Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze

Camerino, 15/09/2010

Linda communication model...

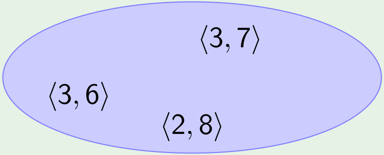
Basic ingredients:

- Asynchronous communication;
- Shared tuple space;
- Pattern matching

Linda communication model...

Basic ingredients:

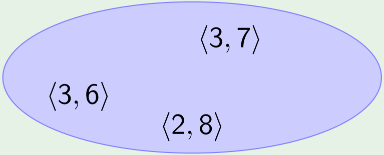
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

- Asynchronous communication;
- Shared tuple space;
- Pattern matching

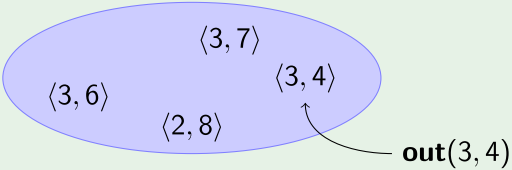


$\mathbf{out}(3, 4)$

Linda communication model...

Basic ingredients:

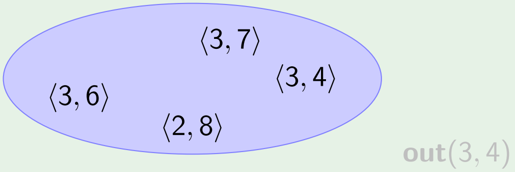
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model. . .

Basic ingredients:

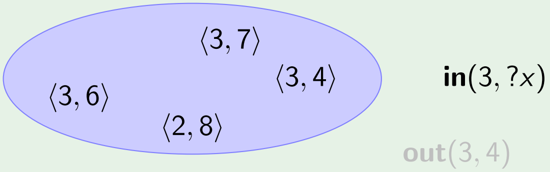
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

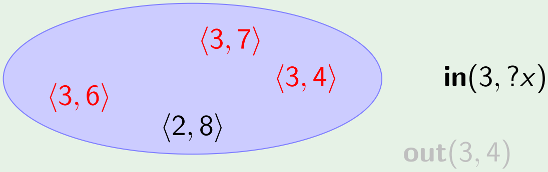
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

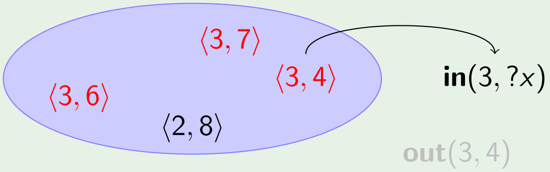
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

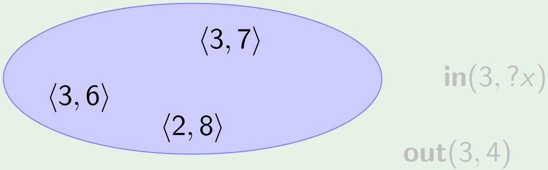
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

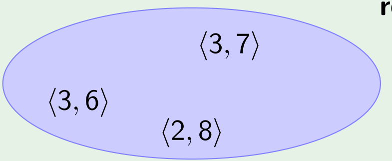
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model. . .

Basic ingredients:

- Asynchronous communication;
- Shared tuple space;
- Pattern matching



$\mathbf{read}(?x, ?y)$

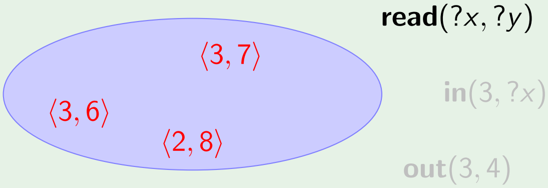
$\mathbf{in}(3, ?x)$

$\mathbf{out}(3, 4)$

Linda communication model...

Basic ingredients:

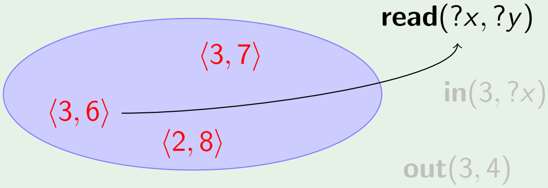
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model...

Basic ingredients:

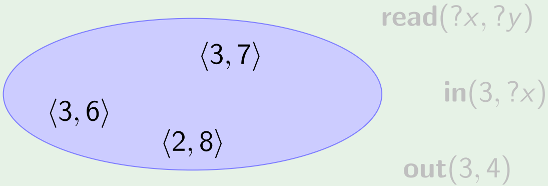
- Asynchronous communication;
- Shared tuple space;
- Pattern matching



Linda communication model. . .

Basic ingredients:

- Asynchronous communication;
- Shared tuple space;
- Pattern matching



KLAIM...

Kernel Language for Agent Interaction and Mobility

Explicit Distribution

- Multiple distributed tuple spaces;
- Code and Process mobility.

KLAIM Nodes

consist of:

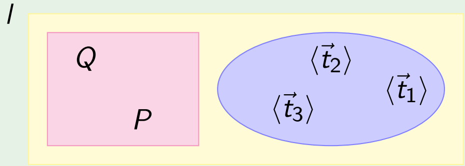
- a site
- a tuple space
- a set of parallel processes

KLAIM Nets...

...consist of a set of KLAIM nodes running in parallel

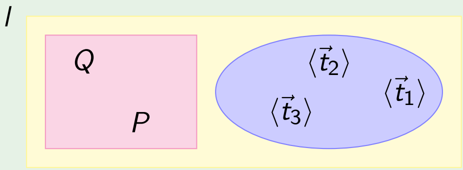
Klaim...

Kernel Language for Agent Interaction and Mobility



KLAIM...

Kernel Language for Agent Interaction and Mobility



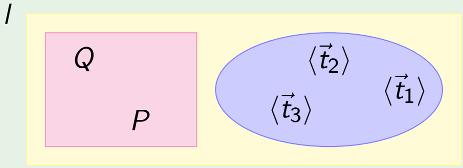
read$(\vec{t})@l$

in$(\vec{t})@l$

out$(\vec{d})@l$

Klaim. . .

Kernel Language for Agent Interaction and Mobility



eval$(R)@l$

read$(\vec{t})@l$

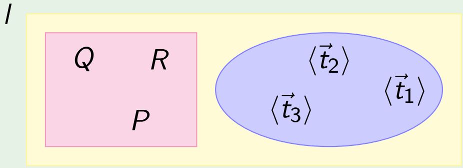
in$(\vec{t})@l$

out$(\vec{d})@l$

Klaim...

Kernel Language for Agent Interaction and Mobility

$\mathbf{eval}(R)@l$



$\mathbf{read}(\vec{t})@l$

$\mathbf{in}(\vec{t})@l$

$\mathbf{out}(\vec{d})@l$

StoKlaim: Stochastic Klaim

StoKlaim: Stochastic Klaim

StoKlaim is the stochastic extension of Klaim where:

- Actions execution take time
- Execution times is described by means of Random Variables

StoKlaim: Stochastic Klaim

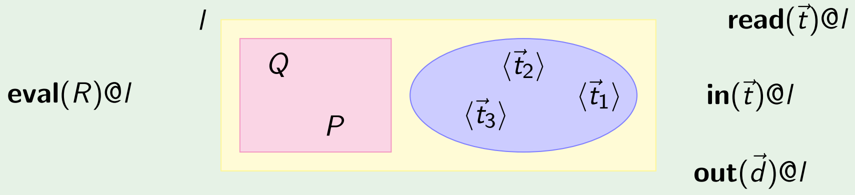
StoKlaim is the stochastic extension of Klaim where:

- Actions execution take time
- Execution times is described by means of Random Variables
    - are assumed to be exponentially distributed
    - are fully characterized by their rates

StoKlaim: Stochastic Klaim

StoKlaim is the stochastic extension of Klaim where:

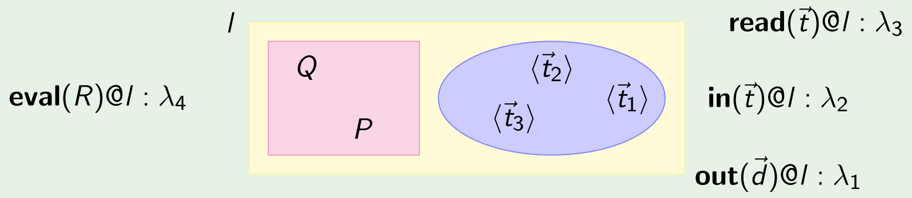
- Actions execution take time
- Execution times is described by means of Random Variables
  - are assumed to be exponentially distributed
  - are fully characterized by their rates



StoKlaim: Stochastic Klaim

StoKlaim is the stochastic extension of Klaim where:

- Actions execution take time
- Execution times is described by means of Random Variables
  - are assumed to be exponentially distributed
  - are fully characterized by their rates



A simple example. . .

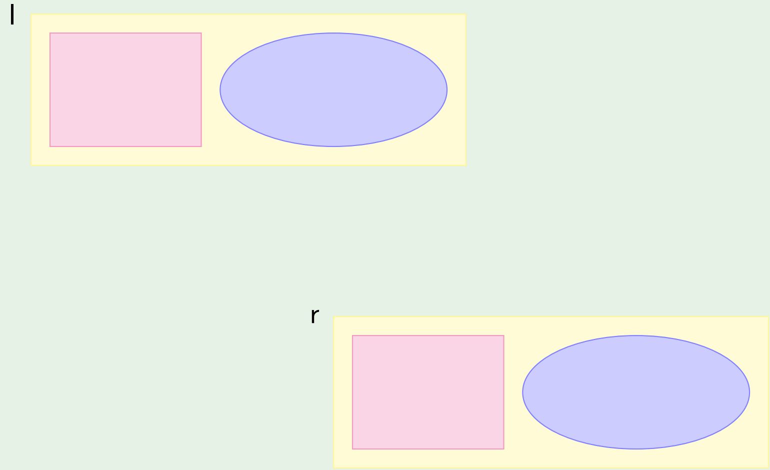
DMS: Distributed Mobile Service

A DMS is a network service that exploits capabilities of different network resources

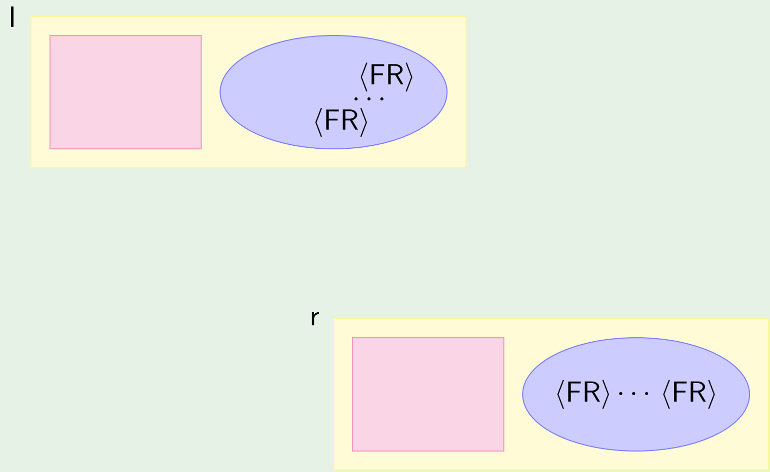
- The service relies on two sites, say l and r.
  - Client software is assumed to run only on l.
- A service dispatcher, running at l, receives service requests from local users and dispatches them to the appropriate sites.
  - There are two types of services, S1 and S2.
  - S1-type service is a simple service that requires only local resources.
  - S2-type service requires first resources in l and then resources in r.

DMS in StoKlaim

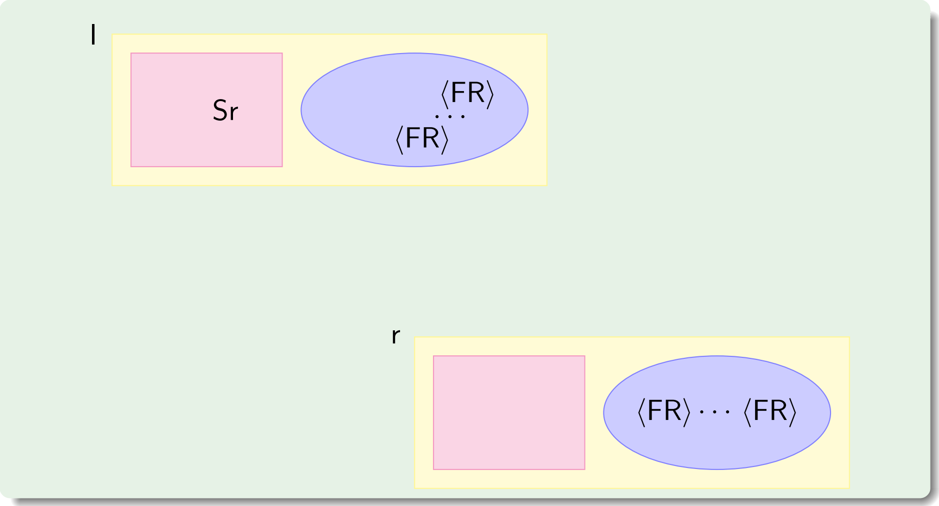
DMS in StoKlaim



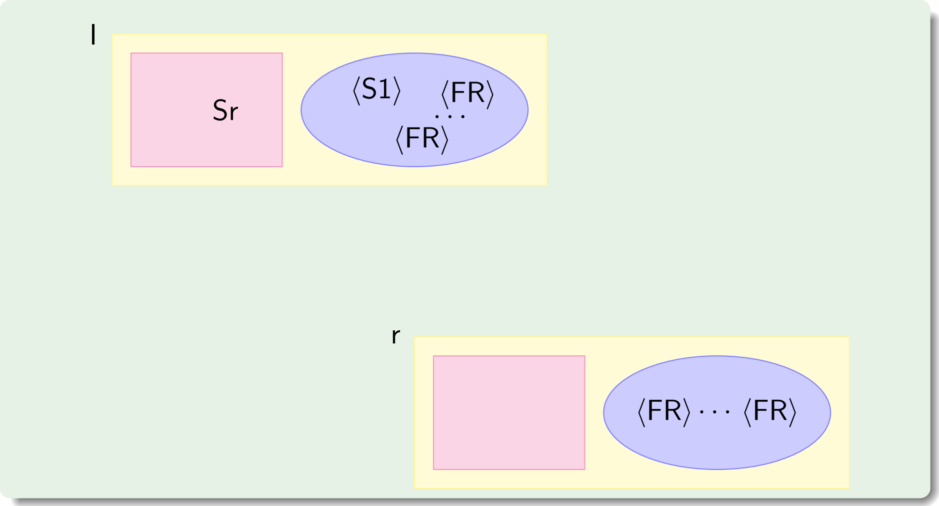
DMS in StoKlaim



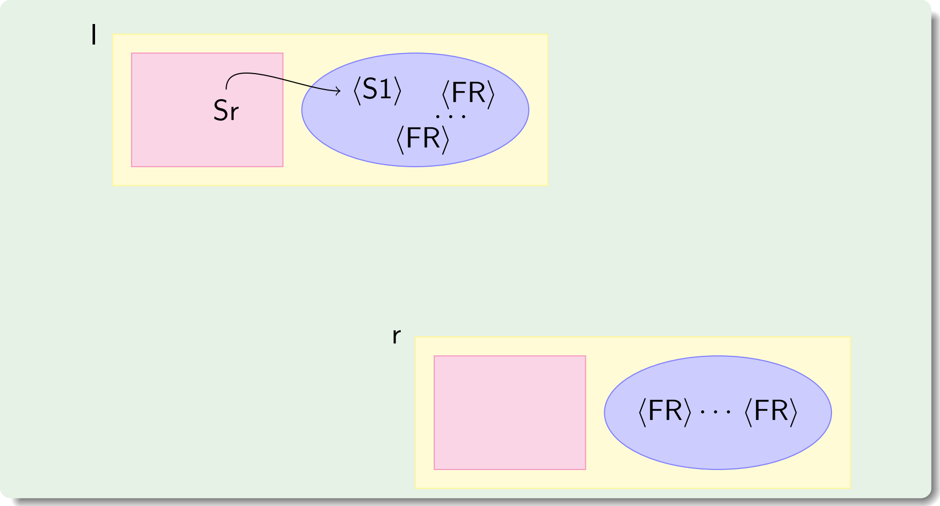
DMS in StoKlaim



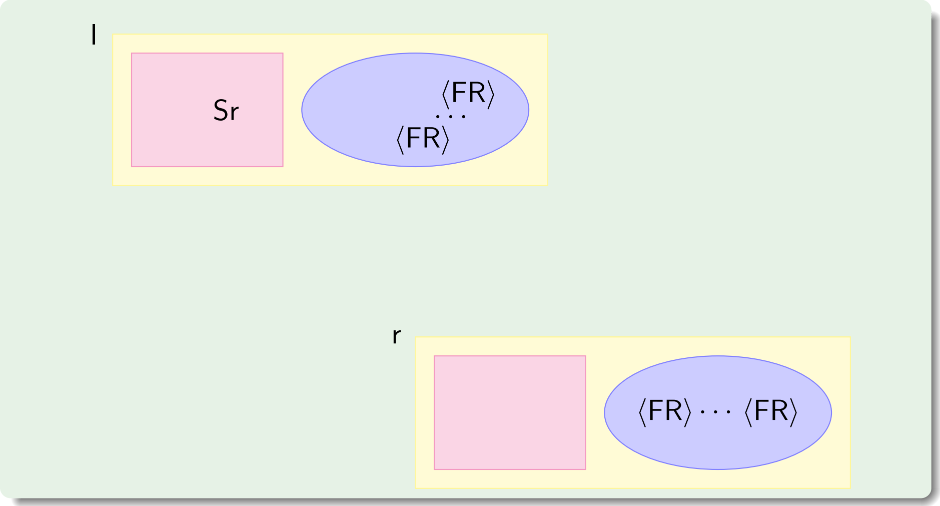
DMS in StoKlaim



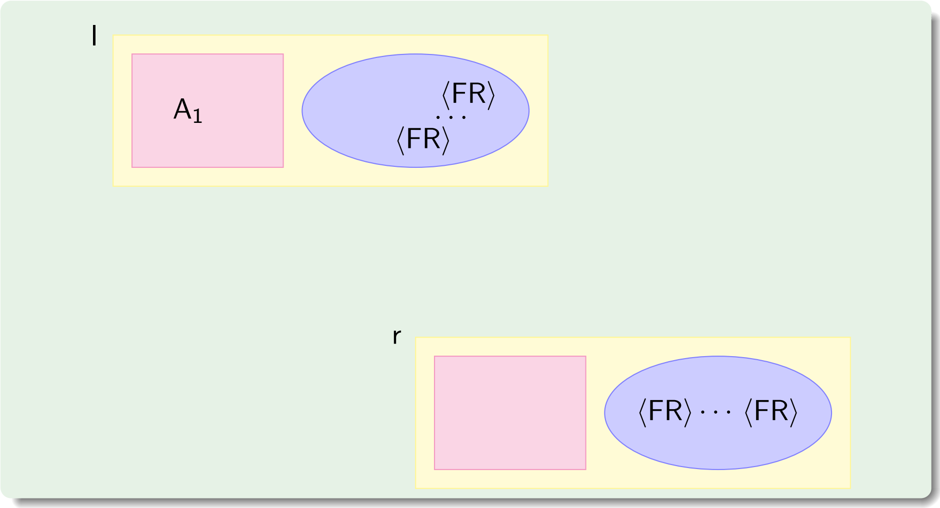
DMS in StoKlaim



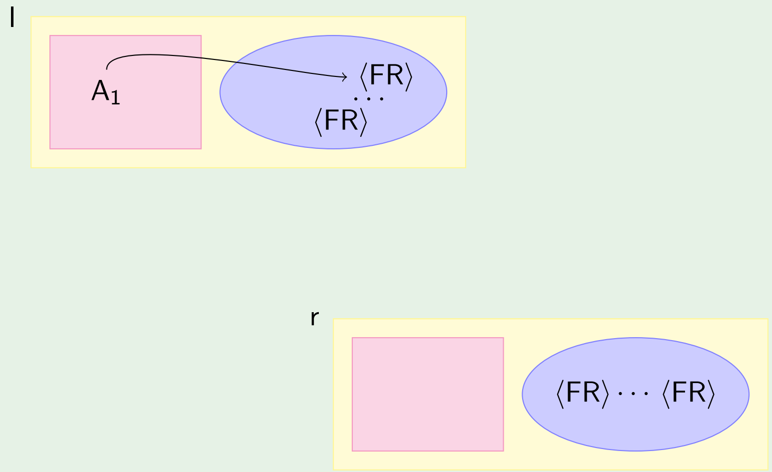
DMS in StoKlaim



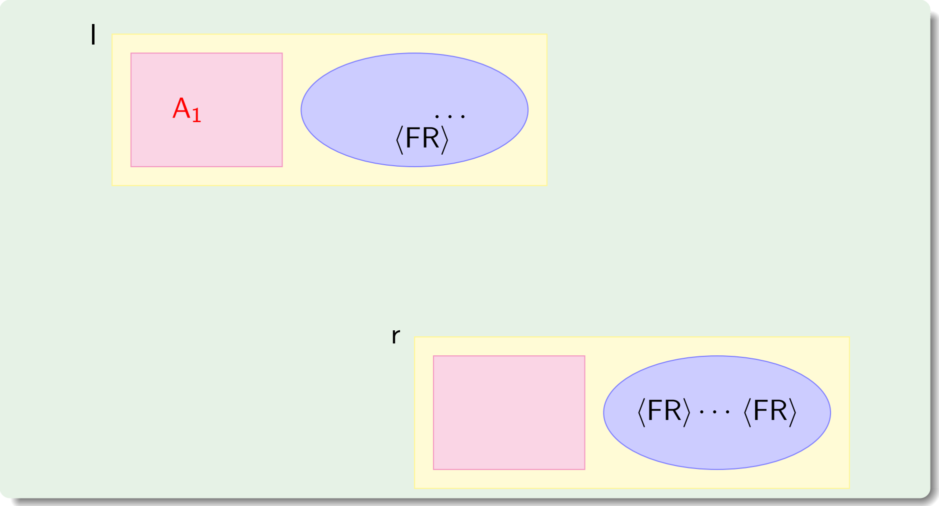
DMS in StoKlaim



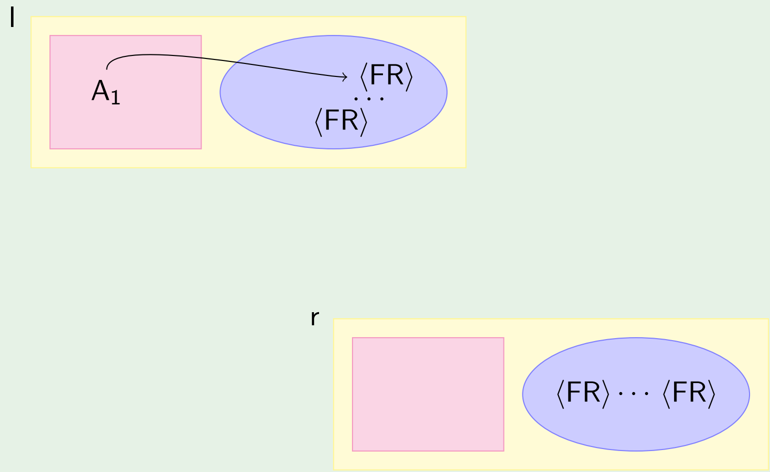
DMS in StoKlaim



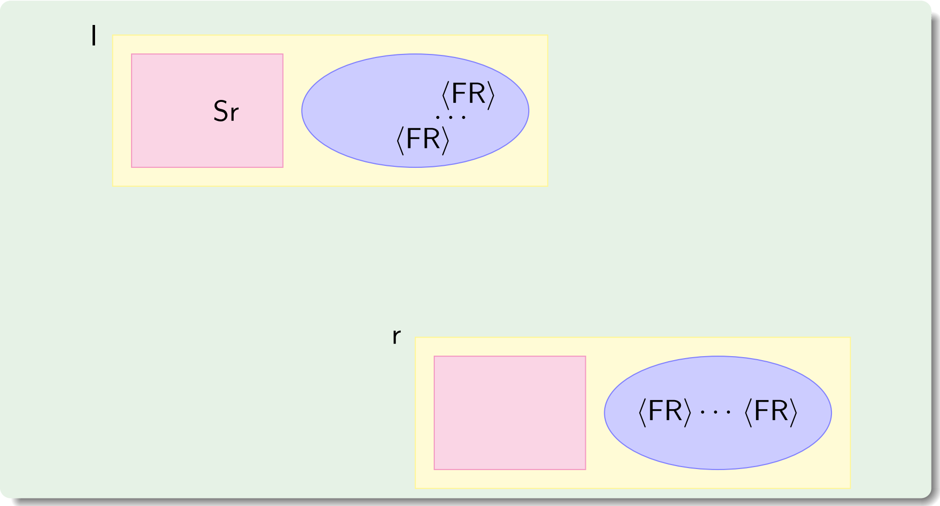
DMS in StoKlaim



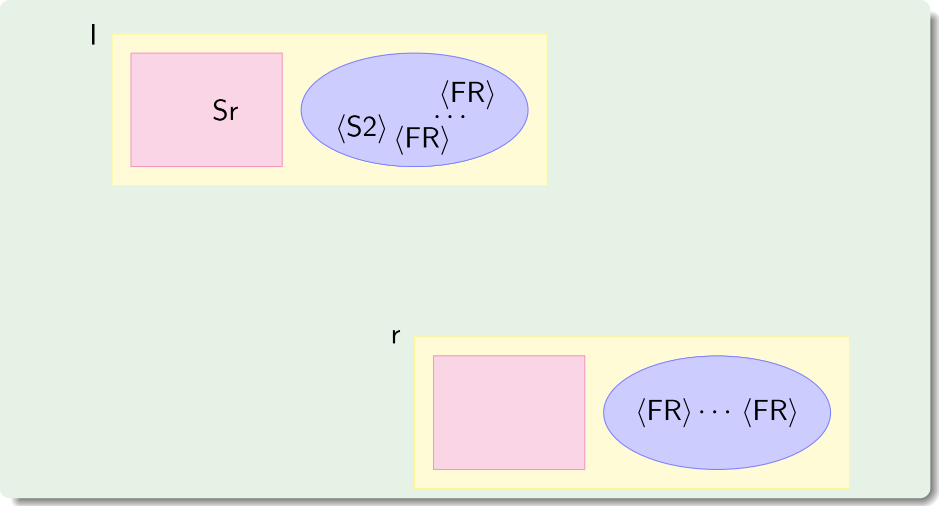
DMS in StoKlaim



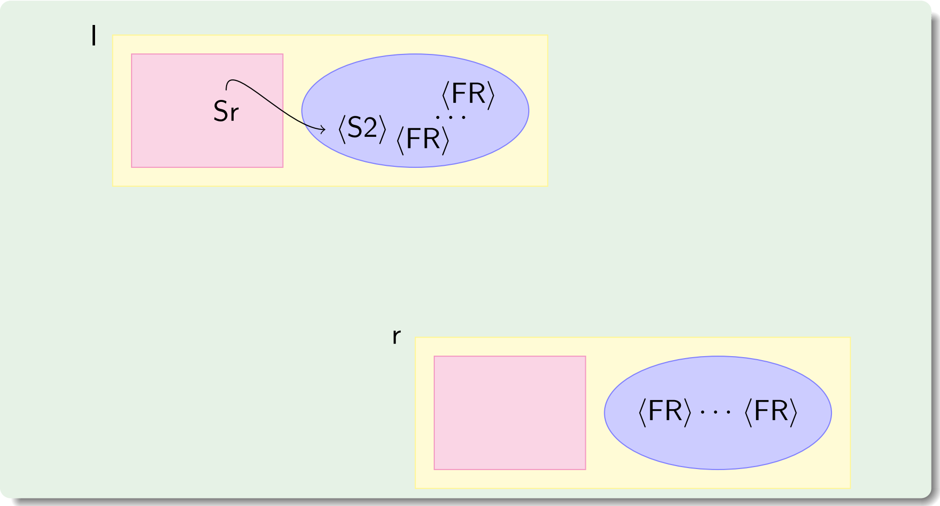
DMS in STOKLAIM



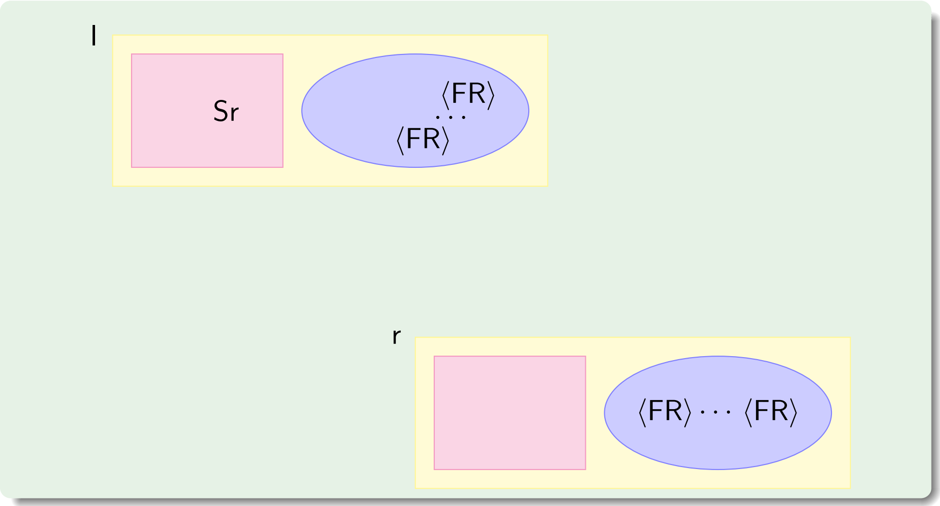
DMS in StoKlaim



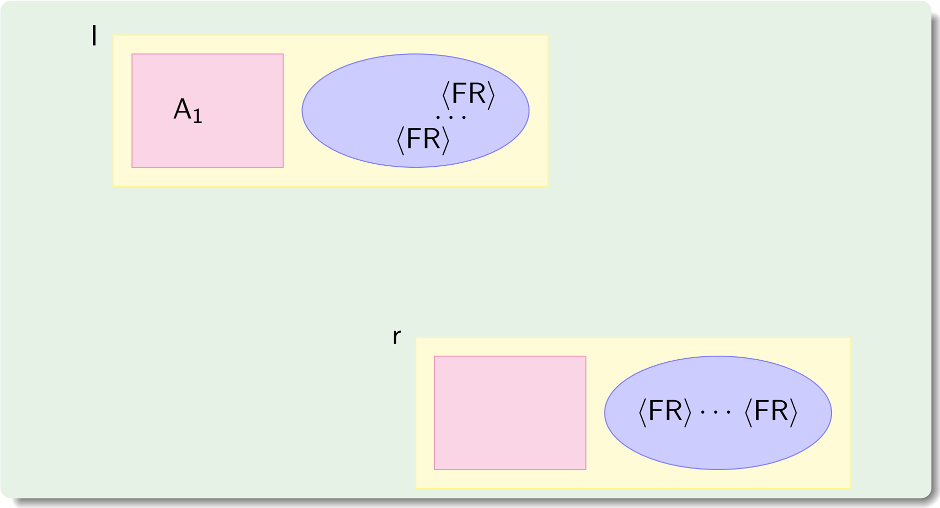
DMS in StoKlaim



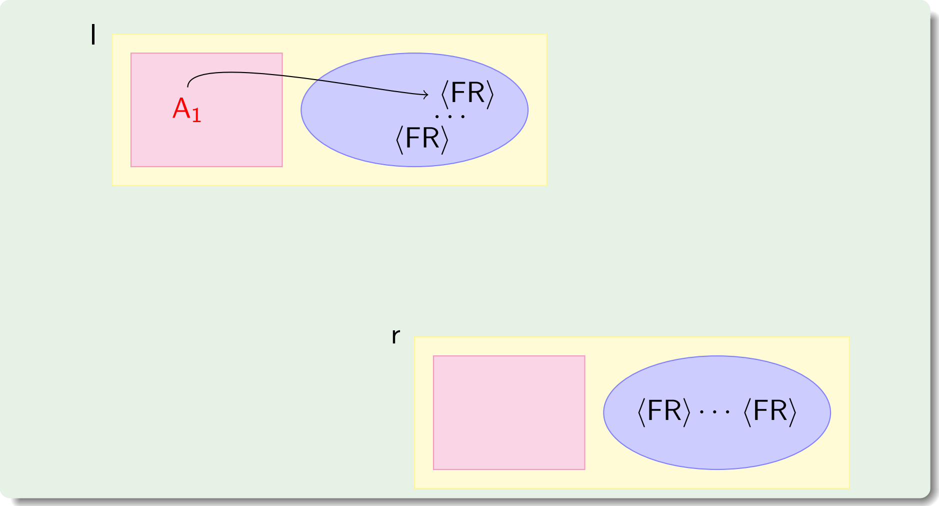
DMS in StoKlaim



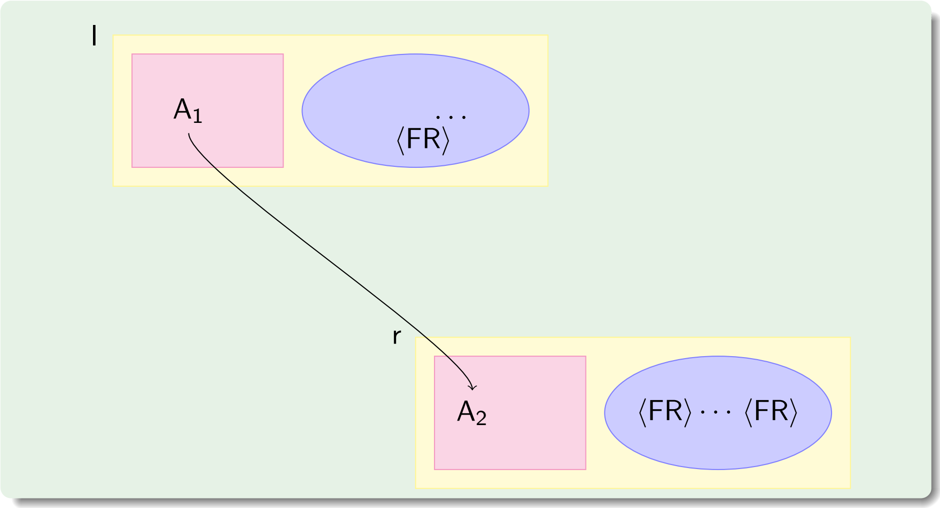
DMS in StoKlaim



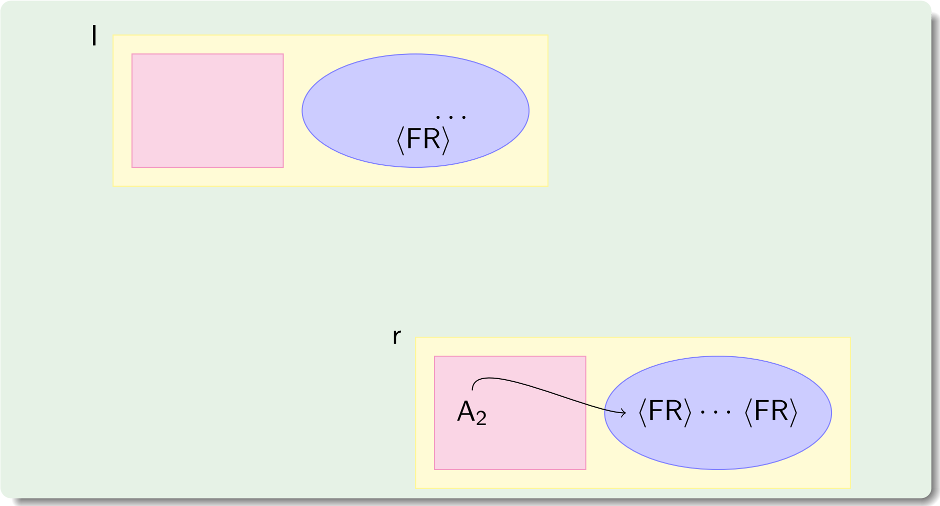
DMS in StoKlaim



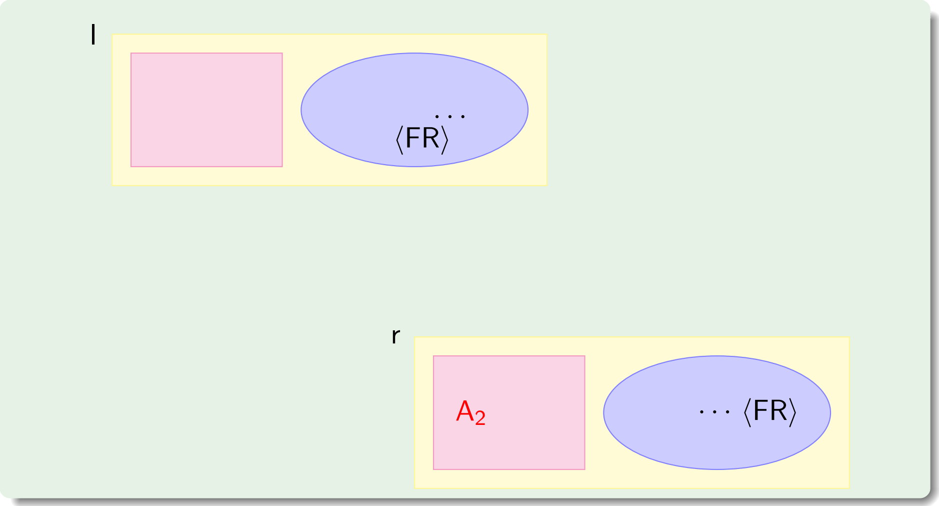
DMS in StoKlaim



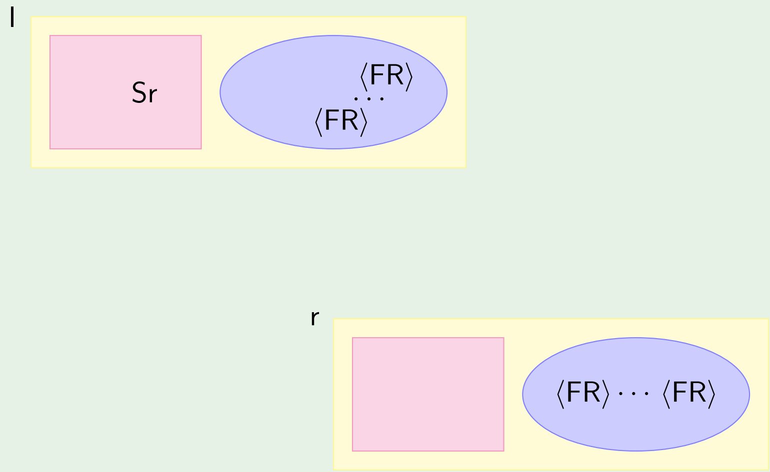
DMS in StoKlaim



DMS in StoKlaim



DMS in StoKlaim



MoSL: Mobile Stochastic Logic

MoSL: Mobile Stochastic Logic

Features:

- a temporal logic (dynamic evolution);
- both action- and state-based;
- a real-time logic (real-time bounds);
- a probabilistic logic (performance and dependability aspects);
- a spatial logic (spatial structure of the network).

MoSL: Mobile Stochastic Logic

Features:

- a temporal logic (dynamic evolution);
- both action- and state-based;
- a real-time logic (real-time bounds);
- a probabilistic logic (performance and dependability aspects);
- a spatial logic (spatial structure of the network).

MoSL extends aCSL with specific operators (*production* and *consumption*) to reason about distribution of resources (*processes* and *tuples*) over the net and to specify the ability to react to *external* stimuli:

MoSL: Mobile Stochastic Logic

Features:

- a temporal logic (dynamic evolution);
- both action- and state-based;
- a real-time logic (real-time bounds);
- a probabilistic logic (performance and dependability aspects);
- a spatial logic (spatial structure of the network).

MoSL extends aCSL with specific operators (*production* and *consumption*) to reason about distribution of resources (*processes* and *tuples*) over the net and to specify the ability to react to *external* stimuli:

$$N \models \langle t \rangle @ l \rightarrow \phi \Leftrightarrow N \equiv N' \parallel l :: \langle t \rangle \text{ and } N' \models \phi$$

MoSL: Mobile Stochastic Logic

Features:

- a temporal logic (dynamic evolution);
- both action- and state-based;
- a real-time logic (real-time bounds);
- a probabilistic logic (performance and dependability aspects);
- a spatial logic (spatial structure of the network).

MoSL extends aCSL with specific operators (*production* and *consumption*) to reason about distribution of resources (*processes* and *tuples*) over the net and to specify the ability to react to *external* stimuli:

$$N \models \langle t \rangle @ l \rightarrow \phi \Leftrightarrow N \equiv N' \parallel l :: \langle t \rangle \text{ and } N' \models \phi$$

$$N \models \langle t \rangle @ l \leftarrow \phi \Leftrightarrow N \parallel l :: \langle t \rangle \models \phi$$

MoSL: Mobile Stochastic Logic

Features:

- a temporal logic (dynamic evolution);
- both action- and state-based;
- a real-time logic (real-time bounds);
- a probabilistic logic (performance and dependability aspects);
- a spatial logic (spatial structure of the network).

- Which is the probability that within t time units both local and remote resources are unavailable?

$$\mathcal{P}_{=?}(\mathsf{true}\ \mathcal{U}^{\leq t}\ \neg(\langle \mathsf{FR} \rangle @\mathsf{l} \rightarrow \mathsf{true} \vee \langle \mathsf{FR} \rangle @\mathsf{r} \rightarrow \mathsf{true})$$

Model-checking MoSL

- Model-checking of MoSL formulae is performed by relying on a CSL model checker.
- The proposed model-checking algorithm manipulates the Rate Transition System (RTS) obtained from a StoKlaim specification:
  - the RTS to be model-checked is translated into an equivalent state-labelled CTMC
  - obtained CTMC is then analysed by making use of existing (state-based) CSL model checkers.

Model-checking MoSL

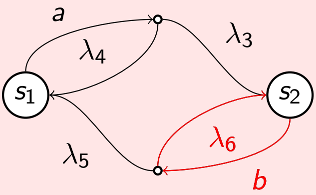
- Model-checking of MoSL formulae is performed by relying on a CSL model checker.
- The proposed model-checking algorithm manipulates the Rate Transition System (RTS) obtained from a StoKlaim specification:
  - the RTS to be model-checked is translated into an equivalent state-labelled CTMC
  - obtained CTMC is then analysed by making use of existing (state-based) CSL model checkers.

Warning:

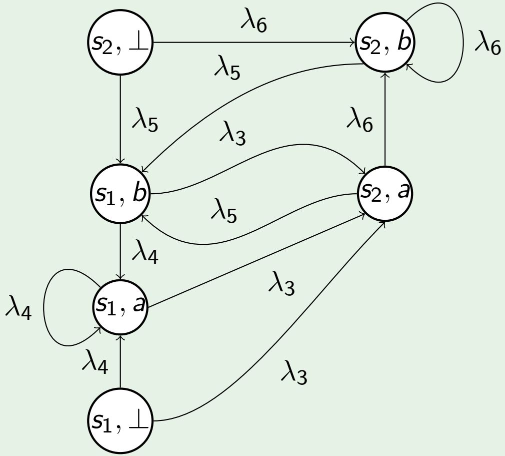
State-space explotion problem!

Translation. . .

From RTS. . .



. . . to CTMC



DMS: State space

Clients	States	Transitions
1	15	16
5	2254	7960
10	42207	199870
15	264860	1402480
20	1012088	5683790

DMS: State space

Clients	States	Transitions
1	15	16
5	2254	7960
10	42207	199870
15	264860	1402480
20	1012088	5683790

Numerical model checking is not practicable!

Statistical model checking

To overcome the state explosion problem, a statistical model-checker can be used.

- this approach has been successfully applied to existing model checkers (YMER, sCOWS,. . .)

Statistical model checking

To overcome the state explosion problem, a statistical model-checker can be used.

- this approach has been successfully applied to existing model checkers (YMER, sCOWS,. . .)

In a numerical model-checker, the exact probability to satisfy a path-formula is computed up to a precision ε.

Statistical model checking

To overcome the state explosion problem, a statistical model-checker can be used.

- this approach has been successfully applied to existing model checkers (YMER, sCOWS,. . .)

In a numerical model-checker, the exact probability to satisfy a path-formula is computed up to a precision ε.

A statistical model-checker is parametrised with respect to a given tollerance ε and error probability δ. The algorithm guarantees that the difference between the computed values and the exact ones are greater than ε with a probability that is less than δ.

DMS Analysis (50 clients, 3 resources)

DMS Analysis (50 clients, 3 resources)

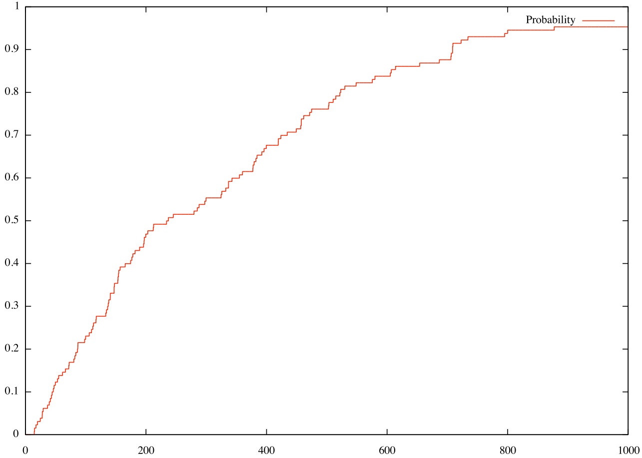


Figure: $\delta = 0.1$, $\varepsilon = 0.2$ (Execution time=13.6 sec)

DMS Analysis (50 clients, 3 resources)

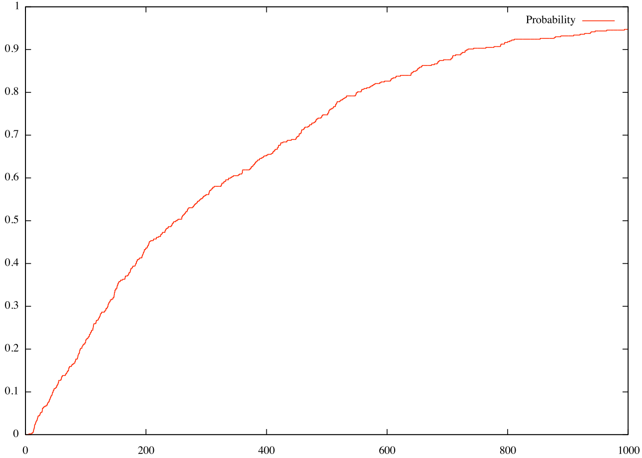


Figure: $\delta = 0.1$, $\varepsilon = 0.1$ (Execution time=56.2 sec)

DMS Analysis (50 clients, 3 resources)

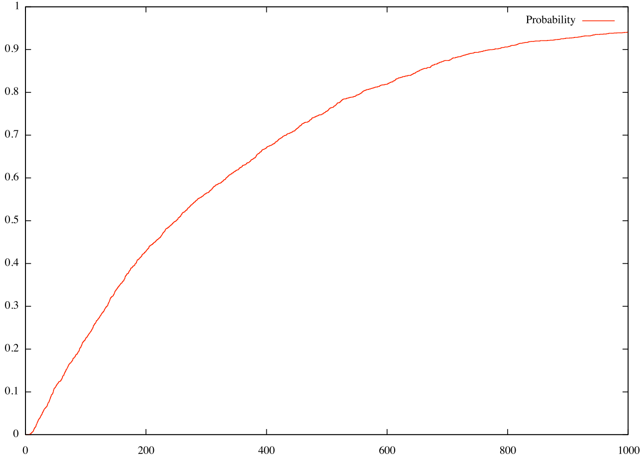


Figure: $\delta = 0.1$, $\varepsilon = 0.05$ (Execution time=233.2 sec)

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

- obtained results are not *precise*

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

- obtained results are not *precise*
- even if highly parallelizable, simulations of *large-scale systems* is time-consuming

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

- obtained results are not *precise*
- even if highly parallelizable, simulations of *large-scale systems* is time-consuming

Following the approach already applied to other Stochastic PA (PEPA) we render behaviour of a concurrent as a family of continuous functions defined in term of Ordinary Differential Equations (ODEs).

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

- obtained results are not *precise*
- even if highly parallelizable, simulations of *large-scale systems* is time-consuming

Following the approach already applied to other Stochastic PA (PEPA) we render behaviour of a concurrent as a family of continuous functions defined in term of Ordinary Differential Equations (ODEs).

We consider a family of continuous random variables of the form:

$\mathcal{F}^{k}_{P@l}(t)=$ probability to have k instances of process P running at locality l at time t

$\mathcal{F}^{k}_{T@l}(t)=$ probability to have k instances of tuple T at locality l at time t

Continuous Semantics of StoKlaim nets

Statistical model checking does not completely resolve the problem:

- obtained results are not *precise*
- even if highly parallelizable, simulations of *large-scale systems* is time-consuming

Following the approach already applied to other Stochastic PA (PEPA) we render behaviour of a concurrent as a family of continuous functions defined in term of Ordinary Differential Equations (ODEs).

- ODE systems can be resolved by relying on standard iterative ODE resolution methods
- Obtained values can be used to model check a large class of formulae (e.g. reachability properties)

Continuous Semantics of StoKlaim nets

$X = \mathbf{out}(T)@l : \lambda.Y$

$$X \xrightarrow{\mathbf{out}(T)@l\,:\,\lambda} Y$$

Continuous Semantics of StoKlaim nets

$X = \mathbf{out}(T)@l : \lambda.Y$

$$X \xrightarrow{\mathbf{out}(T)@l\,:\,\lambda} Y$$

ODE Functions (Kolmogorov-Chapman): $(k > 0)$

$$\frac{d\mathcal{F}^{k}_{X@l}}{dt} = \lambda \cdot (k+1) \cdot \mathcal{F}^{k+1}_{X@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k}_{X@l}(t)$$

Continuous Semantics of StoKlaim nets

$X = \mathbf{out}(T)@l : \lambda.Y$

$$\textcircled{X} \xrightarrow{\mathbf{out}(T)@l\,:\,\lambda} \textcircled{Y}$$

ODE Functions (Kolmogorov-Chapman): $(k > 0)$

$$\frac{d\mathcal{F}^{k}_{X@l}}{dt} = \lambda \cdot (k+1) \cdot \mathcal{F}^{k+1}_{X@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k}_{X@l}(t)$$

There are $k+1$ instances of X at l one of which performs action $\mathbf{out}(T)@l$, the probability to have k instances of X at l increases.

Continuous Semantics of StoKlaim nets

$X = \mathbf{out}(T)@l : \lambda.Y$

$$X \xrightarrow{\mathbf{out}(T)@l \,:\, \lambda} Y$$

ODE Functions (Kolmogorov-Chapman): $(k > 0)$

$$\frac{d\mathcal{F}^{k}_{X@l}}{dt} = \lambda \cdot (k+1) \cdot \mathcal{F}^{k+1}_{X@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k}_{X@l}(t)$$

There are k instances of X at l one of which performs action $\mathbf{out}(T)@l$, the probability to have k instances of X at l decreases.

Continuous Semantics of StoKlaim nets

$X = \mathbf{out}(T)@l : \lambda.Y$

$$X \xrightarrow{\mathbf{out}(T)@l\,:\,\lambda} Y$$

ODE Functions (Kolmogorov-Chapman): $(k > 0)$

$$\frac{d\mathcal{F}^{k}_{X@l}}{dt} = \lambda \cdot (k+1) \cdot \mathcal{F}^{k+1}_{X@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k}_{X@l}(t)$$

$$\frac{d\mathcal{F}^{k}_{Y@l}}{dt} = \lambda \cdot \mathcal{F}^{k-1}_{Y@l}(t) \cdot \sum_{i} i \cdot \mathcal{F}^{i}_{X@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k1}_{P@l}(t) \cdot \sum_{i} i \cdot \mathcal{F}^{i}_{X@l}(t)$$

$$\frac{d\mathcal{F}^{k}_{T@l}}{dt} = \lambda \cdot \mathcal{F}^{k-1}_{T@l}(t) \cdot \sum_{i} i \cdot \mathcal{F}^{i}_{T@l}(t) - \lambda \cdot k \cdot \mathcal{F}^{k1}_{T@l}(t) \cdot \sum_{i} i \cdot \mathcal{F}^{i}_{T@l}(t)$$

Producer-consumer

Example

We consider a simple system composed of two processes:

- A set of *producers* that continuously emit a tuple T
- A set of *consumers* that continuously retrieve tuple T from the tuple space

Producer-consumer

Example

We consider a simple system composed of two processes:

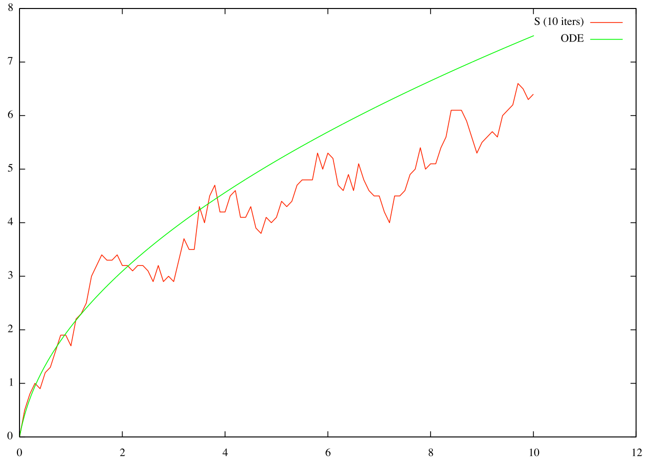
- A set of *producers* that continuously emit a tuple T
- A set of *consumers* that continuously retrieve tuple T from the tuple space

Warning!

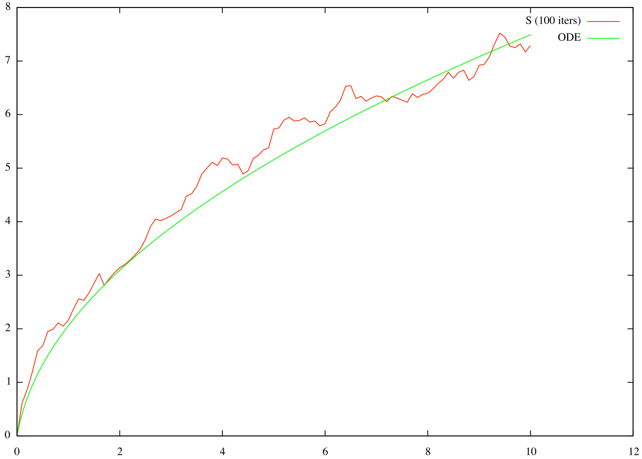
This simple specification generates an infinite state space!

Producer-consumer: ODE vs Simulation

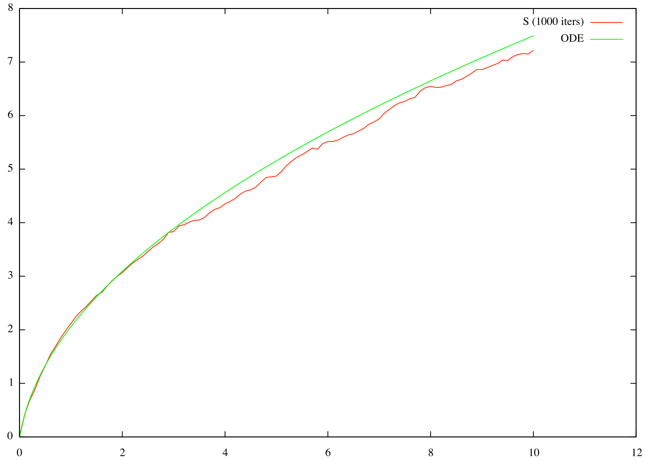
Producer-consumer: ODE vs Simulation



Producer-consumer: ODE vs Simulation



Producer-consumer: ODE vs Simulation



Concluding remarks

Concluding remarks

- Classical analytical tools for quantitative properties are based on CTMC and requires significative efforts to generate the complete CTMC associated to a process
  - the number of states is exponential with respect to the size of the corresponding process
  - the standard numerical model checking techniques cannot always be applied.

Concluding remarks

- Classical analytical tools for quantitative properties are based on CTMC and requires significative efforts to generate the complete CTMC associated to a process
  - the number of states is exponential with respect to the size of the corresponding process
  - the standard numerical model checking techniques cannot always be applied.

- To overcome the problem, we have considered alternative model-checking techniques:
  - statistical model checking
  - continuous approximation

Concluding remarks

- Classical analytical tools for quantitative properties are based on CTMC and requires significative efforts to generate the complete CTMC associated to a process
  - the number of states is exponential with respect to the size of the corresponding process
  - the standard numerical model checking techniques cannot always be applied.

- To overcome the problem, we have considered alternative model-checking techniques:
  - statistical model checking
  - continuous approximation

A tool has been developed for supporting analysis of StoKlaim systems:

- http://rap.dsi.unifi.it/SAM/

On going work

- Complete the integration of ODE semantics of StoKlaim in SAM
- Equip StoKlaim with a language for specifying *rewards* and extends MoSL accordingly
- Use StoKlaim and its tools to specify and verify a larger class of case studies
  - networks (gossip protocols, crowds,. . .)
  - economics
  - evolutionary systems
  - biological systems
  - . . .

Thank you for your attention!