Toward Approximate Stochastic Model Checking of Computational Fields for Pervasive Computing Systems

Matteo Casadei, Mirko Viroli
{m.casadei,mirko.viroli}@unibo.it

Alma Mater Studiorum—Università di Bologna

ASENSIS, 10/9/2012

Outline

Preview

Problem

$\Rightarrow$ tackling verification in field-based self-organising systems

Goal

$\Rightarrow$ exploiting approximate stochastic model-checking and Prism

Strategy

$\Rightarrow$ developing a high-level language translating to Prism

Use

$\Rightarrow$ showing few example applications and results

Motivating Setting

An abstract network model for pervasive computing

- A set of interconnected nodes situated in space
- Each node asynchronously interacts with a small neighbourhood
- Topology can be very dynamic due to mobility and faults

Example problem

- Node n advertises an event in a large locality $L(n)$
- An "annotation" (tuple, data) in $m \in L(n)$ then moves towards n

General application scenarios – many rooted in SAPERE

- Steering people in pervasive computing scenarios [6]
- Message routing in wireless sensor networks [2]
- Mobile robot applications [1]

Motivating Setting

An abstract network model for pervasive computing

- A set of interconnected nodes situated in space
- Each node asynchronously interacts with a small neighbourhood
- Topology can be very dynamic due to mobility and faults

Example problem

- Node n advertises an event in a large locality $L(n)$
- An "annotation" (tuple, data) in $m \in L(n)$ then moves towards n

General application scenarios – many rooted in SAPERE

- Steering people in pervasive computing scenarios [6]
- Message routing in wireless sensor networks [2]
- Mobile robot applications [1]

Motivating Setting

An abstract network model for pervasive computing

- A set of interconnected nodes situated in space
- Each node asynchronously interacts with a small neighbourhood
- Topology can be very dynamic due to mobility and faults

Example problem

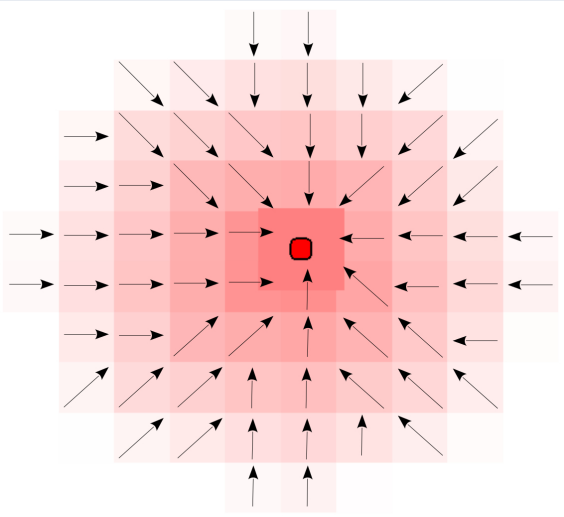
- Node n advertises an event in a large locality $L(n)$
- An "annotation" (tuple, data) in $m \in L(n)$ then moves towards n

General application scenarios – many rooted in SAPERE

- Steering people in pervasive computing scenarios [6]
- Message routing in wireless sensor networks [2]
- Mobile robot applications [1]

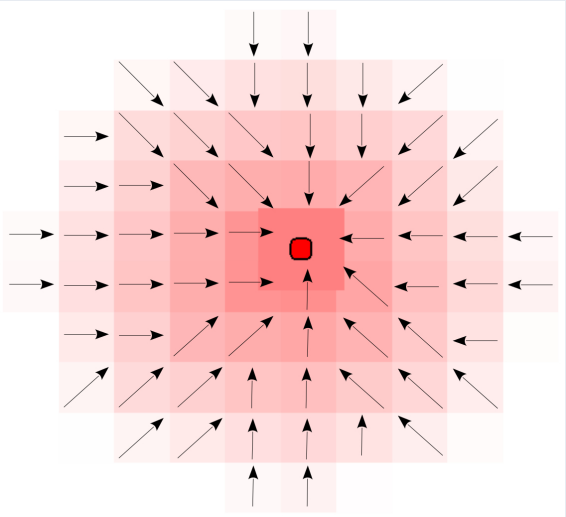
A solution by so-called "Computational Fields" [4]

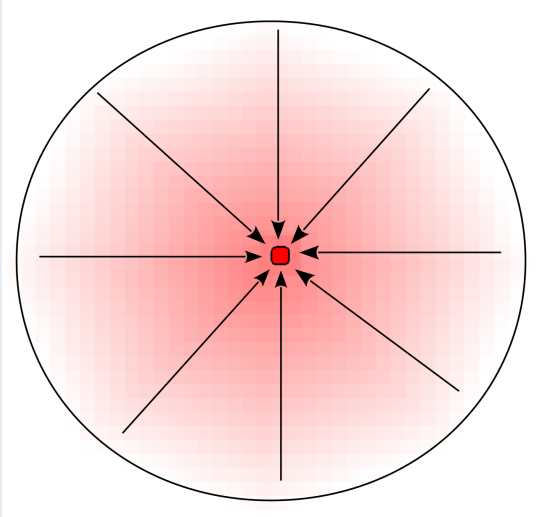
Mapping nodes to values (suggests a continuum space-time viewpoint)



A solution by so-called "Computational Fields" [4]

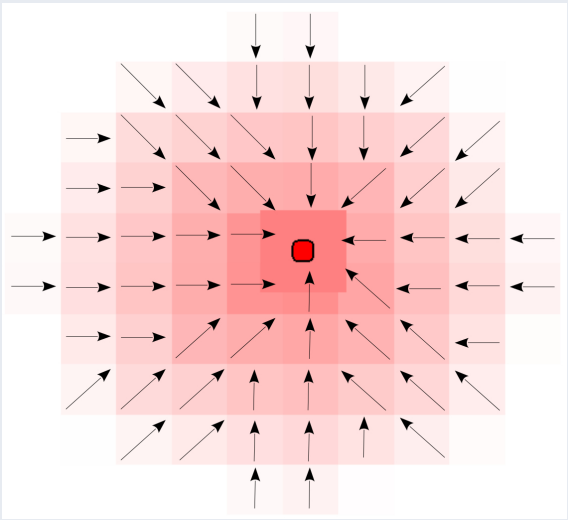
Mapping nodes to values (suggests a continuum space-time viewpoint)

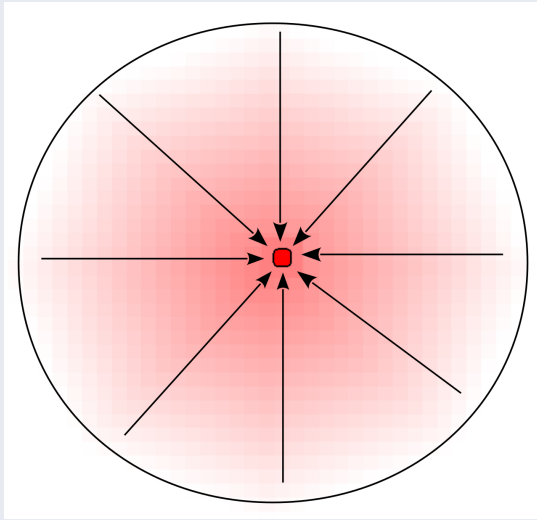


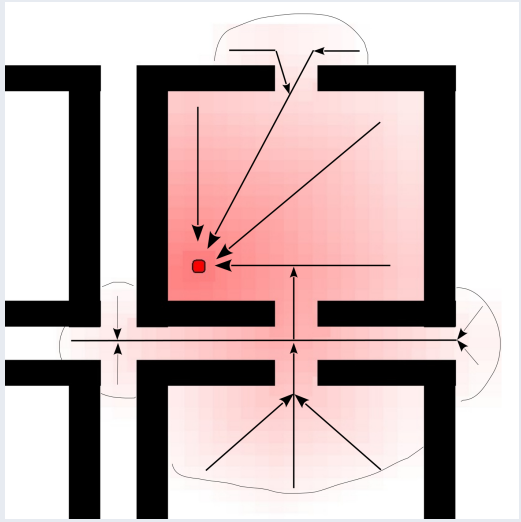


A solution by so-called "Computational Fields" [4]

Mapping nodes to values (suggests a continuum space-time viewpoint)

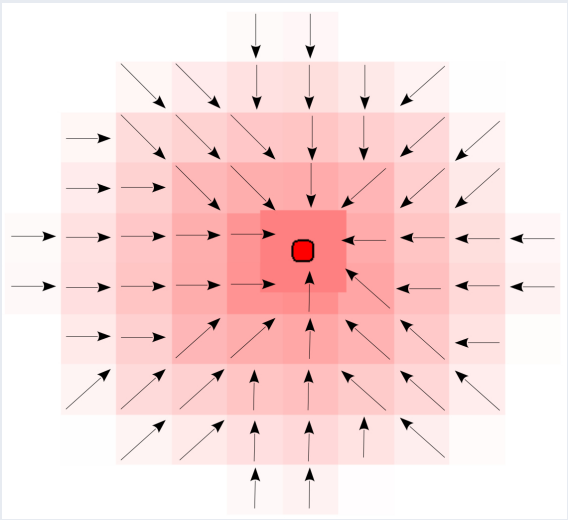


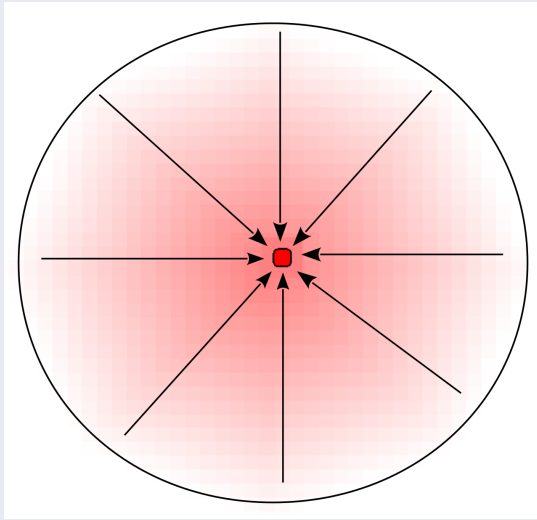


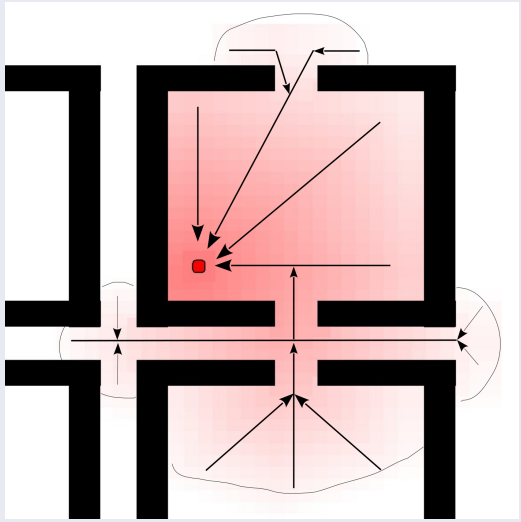


A solution by so-called "Computational Fields" [4]

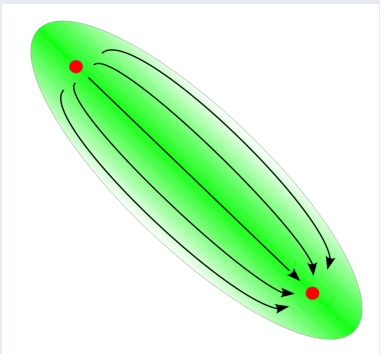
Mapping nodes to values (suggests a continuum space-time viewpoint)

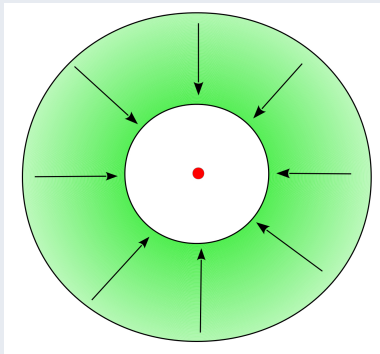


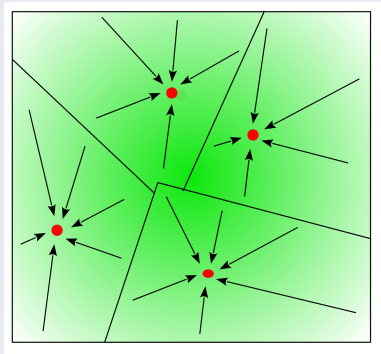


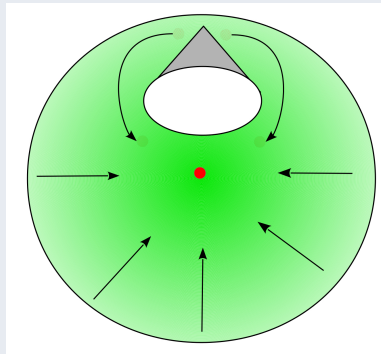


Other structures (channel, shrinking crown, partition, shadow)









Computational Fields and emergence

The predictability/controllability issue

Any guarantee about "appropriateness"?

- Will the computational field stabilise? (or can it diverge?)
- Will the computational field have the proper shape?
- Will people be steered until eventually reaching the POI?

Approaches to assess properties

- Formal proof: difficult to find, typically ad-hoc
- Simulation: the standard-de-facto, often hard to be fully trusted
- Automatic Verification (model-checking): shortly impractical

The predictability/controllability issue

Any guarantee about "appropriateness"?

- Will the computational field stabilise? (or can it diverge?)
- Will the computational field have the proper shape?
- Will people be steered until eventually reaching the POI?

Approaches to assess properties

- Formal proof: difficult to find, typically ad-hoc
- Simulation: the standard-de-facto, often hard to be fully trusted
- Automatic Verification (model-checking): shortly impractical

A solution between Simulation and Automatic Verification

Approximate Stochastic Model Checking [3] (A-SMC)

Tackle the state-space explosion, probabilistically:

- Explore a subset of state-space through a (possibly high) number of stochastic simulations (requires less time and less space than MC)
- Result: probability for the property to hold, with known confidence

Three key parameters

1. Number of independent simulation runs N
2. Approximation ϵ: the desired precision on the obtained probability
3. Confidence factor δ: probability that approximation is not met

$\Rightarrow$ (Definition of ϵ and δ: $Prob[|M_{exact} - M_{approx}| \leq \epsilon] \geq 1 - \delta$)

$\Rightarrow$ Parameters are linked: $N \geq 4log(\frac{2}{\delta})/\epsilon^2$

$\Rightarrow$ Our choice: $\epsilon = 0.01$, $\delta = 0.01$, $N \simeq 90'000$.

A solution between Simulation and Automatic Verification

Approximate Stochastic Model Checking [3] (A-SMC)

Tackle the state-space explosion, probabilistically:

- Explore a subset of state-space through a (possibly high) number of stochastic simulations (requires less time and less space than MC)
- Result: probability for the property to hold, with known confidence

Three key parameters

1. Number of independent simulation runs N
2. Approximation ϵ: the desired precision on the obtained probability
3. Confidence factor δ: probability that approximation is not met

$\Rightarrow$ (Definition of ϵ and δ: $Prob[|M_{exact} - M_{approx}| \leq \epsilon] \geq 1 - \delta$)

$\Rightarrow$ Parameters are linked: $N \geq 4log(\frac{2}{\delta})/\epsilon^2$

$\Rightarrow$ Our choice: $\epsilon = 0.01$, $\delta = 0.01$, $N \simeq 90'000$.

A solution between Simulation and Automatic Verification

Approximate Stochastic Model Checking [3] (A-SMC)

Tackle the state-space explosion, probabilistically:

- Explore a subset of state-space through a (possibly high) number of stochastic simulations (requires less time and less space than MC)
- Result: probability for the property to hold, with known confidence

Three key parameters

1. Number of independent simulation runs N
2. Approximation ϵ: the desired precision on the obtained probability
3. Confidence factor δ: probability that approximation is not met

$\Rightarrow$ (Definition of ϵ and δ: $Prob[|M_{exact} - M_{approx}| \leq \epsilon] \geq 1 - \delta$)

$\Rightarrow$ Parameters are linked: $N \geq 4log(\frac{2}{\delta})/\epsilon^2$

$\Rightarrow$ Our choice: $\epsilon = 0.01$, $\delta = 0.01$, $N \simeq 90'000$.

PRISM (www.prismmodelchecker.org)

The reference tool for A-SMC

- Widely used: biochemistry, games, protocols, coordination
- Support for Continuous Stochastic Logic (CSL) and CTMC

The "module" linguistic construct in PRISM:

- State – A small set of bounded numerical variables
- Behaviour – A small set of condition-action transitions
- Network – Can write many modules, also by clone & rename
- Synchronisation – Can influence other modules via synch. transitions

Limits of PRISM as front-end language to our ends

⇒ No first-class support for true (large, dynamic, ad-hoc) topologies

⇒ No first-class support for node-to-node communications

PRISM (www.prismmodelchecker.org)

The reference tool for A-SMC

- Widely used: biochemistry, games, protocols, coordination
- Support for Continuous Stochastic Logic (CSL) and CTMC

The "module" linguistic construct in PRISM:

- State – A small set of bounded numerical variables
- Behaviour – A small set of condition-action transitions
- Network – Can write many modules, also by clone & rename
- Synchronisation – Can influence other modules via synch. transitions

Limits of PRISM as front-end language to our ends

⇒ No first-class support for true (large, dynamic, ad-hoc) topologies

⇒ No first-class support for node-to-node communications

A PRISM-based framework

Three inputs

- Specification of a node (state + behaviour + interaction)
- Specification of a topology (grid, torus, ad-hoc, and the like)
- Specification of a formula to verify (CSL + node quantification)

Two outputs

- (Big) PRISM specification (basically obtained by expansion)
- PRISM formula to verify

Then..

- PRISM is used as usual to run modelchecking
- Specifying ϵ,δ and N
- Charting probability of truth for different parameters

The hop-count gradient case

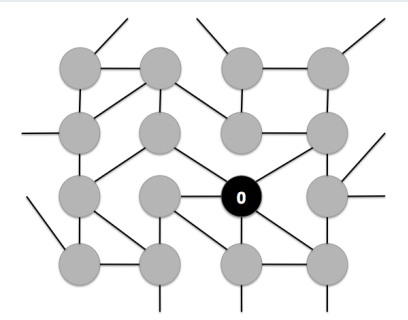
Node specification

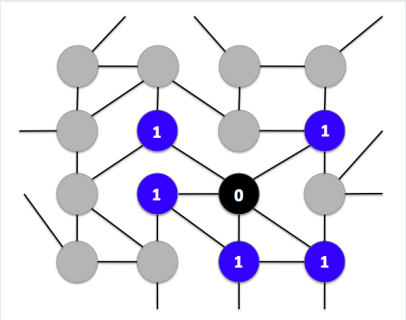
```
pump  : [0..1];  field : [0..MAX];
[]     pump=1 & field>0 -- 1.0 --> field'= 0;
[diff] pump=0           -- 1.0 --> field'= min[@.field]+1;
```

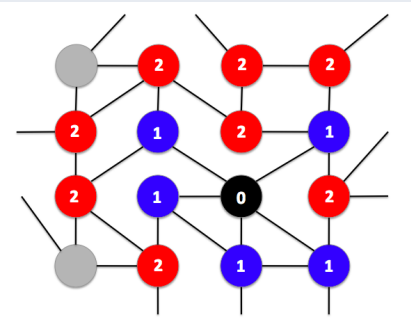
Referencing neighbours

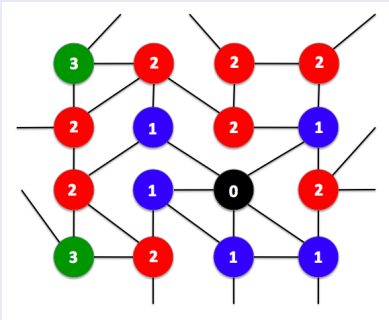
- `min[@.field]`: minimum value of `field` in neighbours

An example on a "random torus"









The hop-count gradient case

Node specification

```
pump  : [0..1];  field : [0..MAX];
[]     pump=1 & field>0 -- 1.0 --> field'= 0;
[diff] pump=0           -- 1.0 --> field'= min[@.field]+1;
```

PRISM specification (grid topology, node 11, having neighbours 13,21,31)

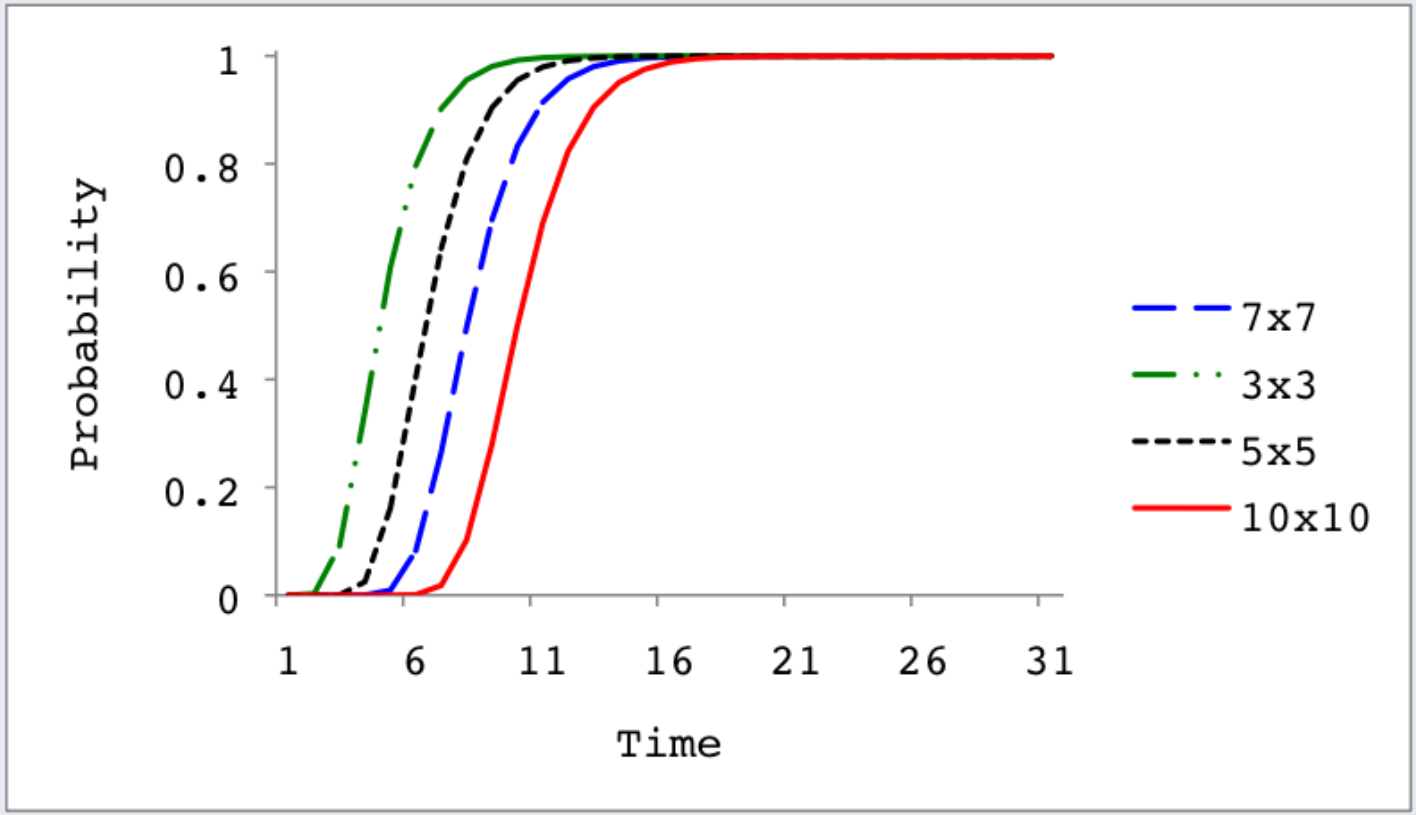
```
module node1_1
  pump1_1 : [0..1] init 1;  field1_1 : [0..MAX] init MAX;
  [] pump1_1>0 & field1_1>0 -> 1.0 : field1_1' = 0;
  [diff_1_1] pump1_1=0 -> 1.0 : field1_1' = min(field1_3,field2_1,field3_1)+1;
endmodule
module node1_2=node1_1 [ diff_1_1=diff_1_2, pump1_1=pump1_2, ..] endmodule
module node2_1=node1_1 [ diff_1_1=diff_2_1, ..] endmodule
...
```

Property to verify and query (stabilisation within "k" time units)

```
property "stab" = forall[(pump=0 & field=min(@.field)+1) | (pump=1 & field=0)];
P=? [F<=k "stab"]   % F is bounded-eventually operator of temporal logics
```

Simulation

Charting probability of convergence within k time units



⇒ Result: stabilisation is reached linearly in the network diameter

⇒ This simulation takes about 2 hours on a 2.66 Ghz Dual-Core PC...

A random walk – showing node synchronisation

Node specification

```
v : [0..1];
[move] v=1 & N:=&any[@.v=0] -- 1.0 --> v'=0 & N.v'=1;
```

Referencing neighbours

- `any[@.v=0]`: any neighbour having `v` set to 0

PRISM specification (node 1, having neighbours 2,3)

```
module node_1
  v_1 : [0..1] init 1;
 [move_1_2] v_1 = 1 & v_2 = 0 -> 1.0 : (v_1'=0); % one per outgoing neighbour
 [move_1_3] v_1 = 1 & v_3 = 0 -> 1.0 : (v_1'=0);
 [move_2_1] true -> 1.0 : (v_1'=1);              % one per incoming neighbour
 [move_3_1] true -> 1.0 : (v_1'=1);
endmodule
module node_2 .. endmodule
module node_3 .. endmodule
```

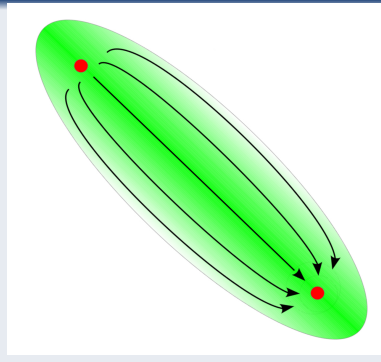
Language Syntax

Module specification

```
S ::= D̄ T̄                                  % Specification
D ::= X : [n_l..n_u];                       % Variable def
T ::= [L] P̄ --e--> Ā;                       % Transition
A ::= V'=e                                  % Assignment
P ::= b | M:=&f[e] | M:=&f[b]               % Precondition
f ::= any | min | max                       % Selection function
e ::= r | V | (e) | e+e | e-e | e*e | -e | f[e]    % exp
b ::= e<=e | e<e | e>=e | e>e | e=e | e!=e          % bool exp
V ::= X | M.X | @.X                                % Variable
r ::= <real-num>                            % (real) Number
n ::= <int-num>                             % (integer) Number
L ::= <literal>                             % Label
X ::= <literal>                             % Variable name
M ::= <literal>                             % Node variable
```

A more involved example – channel structure

Node specification



```
source  : [0..1];   fs : [0..MAX];
target  : [0..1];   ft : [0..MAX];
distance : [0..MAX]; range : [0..MAX];
channel : [0..1];

[]      source=1 & fs>0 -- 100.0 --> fs'= 0 ;
[sdiff] source=0        -- 1.0 --> fs'= min[@.fs]+1;
[]      target=1 & ft>0 -- 100.0 --> ft'= 0 ;
[tdiff] target=0        -- 1.0 --> ft'= min[@.ft]+1;
[dist]  source=1 & ft<MAX -- 1.0 --> distance'=ft;
[goss]  N:=&any[@.distance>distance] -- 1.0 --> N.distance'=N.distance;
[chn]   channel=0 &  fs+ft<distance+range -- 1.0 --> channel'=1
```

Conclusions

Open issues

- Very hard to deal with network mobility, can simulate by:
  - $\Rightarrow$ translating links into modules
  - $\Rightarrow$ such modules activate/disactivate stochastically
- PRISM itself does not scale very well with size of the specification
- A-SMC is becoming popular in academia, but it is not yet a standard
- Can analyse topologies of few hundreds nodes

Future works

- Improve the specification language – still very constrained by PRISM
- Integrating A-SMC in ad-hoc simulators (e.g. ALCHEMIST [5])
- Find proof methodologies for certain classes of fields
- Incorporate a development methodology based on A-SMC in SAPERE

References I

[1] Jonathan Bachrach, Jacob Beal, and James McLurkin.
Composable continuous-space programs for robotic swarms.
Neural Computing and Applications, 19(6):825–847, 2010.

[2] Matteo Casadei, Mirko Viroli, and Luca Gardelli.
On the collective sort problem for distributed tuple spaces.
Sci. of Computer Programming, 74(9):702–722, 2009.

[3] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet.
Approximate probabilistic model checking.
In Bernhard Steffen and Giorgio Levi, editors, *Proc. 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2004.

[4] Marco Mamei and Franco Zambonelli.
Programming pervasive and mobile computing applications: The tota approach.
ACM Trans. Softw. Eng. Methodol., 18(4):1–56, 2009.

[5] Danilo Pianini, Sara Montagna, and Mirko Viroli.
A chemical inspired simulation framework for pervasive services ecosystems.
In Maria Ganzha, Leszek Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 675–682, Szczecin, Poland, 18-21 September 2011. IEEE Computer Society Press.

[6] Mirko Viroli, Danilo Pianini, Sara Montagna, and Graeme Stevenson.
Pervasive ecosystems: a coordination model based on semantic chemistry.
In Sascha Ossowski, Paola Lecca, Chih-Cheng Hung, and Jiman Hong, editors, *27th Annual ACM Symposium on Applied Computing (SAC 2012)*, Riva del Garda, TN, Italy, 26-30 March 2012. ACM.