SimGrid MC 101

Getting Started with the SimGrid Model-Checker

Da SimGrid Team

April 11, 2017

About this Presentation

Goals and Contents

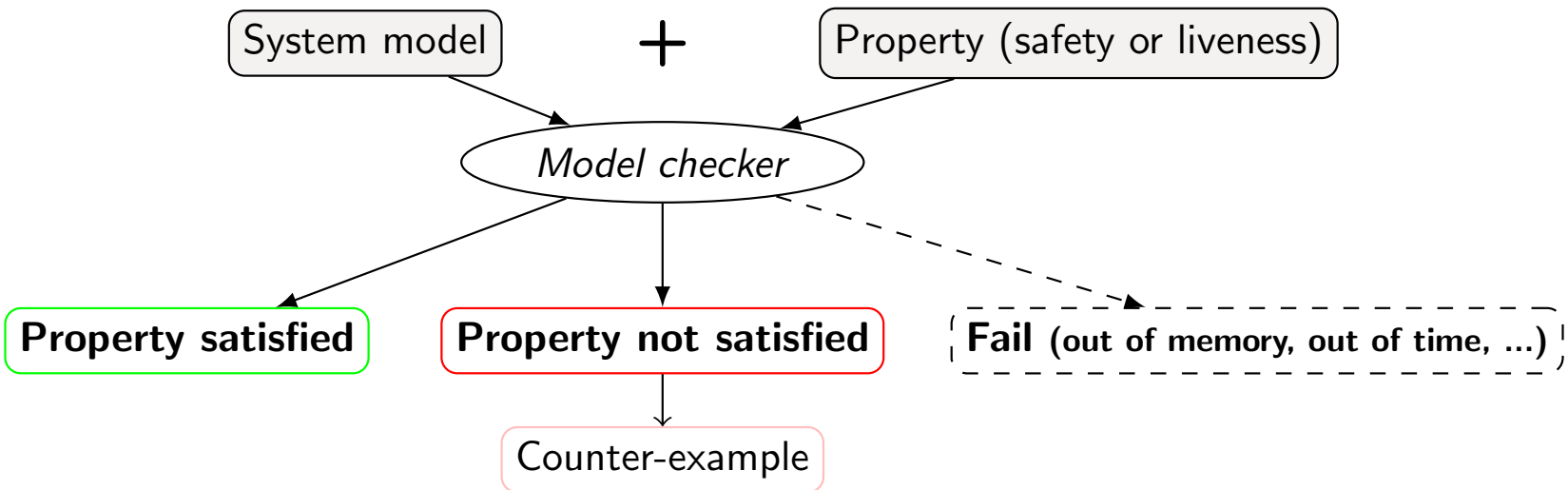
- Understanding the basics of Model checking
- Running SimGrid as a Model Checker
- Analysing the counter-example traces produced

The SimGrid 101 serie

- This is part of a serie of presentations introducing various aspects of SimGrid
- SimGrid 101. Introduction to the SimGrid Scientific Project
- SimGrid User 101. Practical introduction to SimGrid and MSG
- SimGrid User::Platform 101. Defining platforms and experiments in SimGrid
- SimGrid User::SimDag 101. Practical introduction to the use of SimDag
- SimGrid User::Visualization 101. Visualization of SimGrid simulation results
- SimGrid User::SMPI 101. Simulation MPI applications in practice
- SimGrid User::Model-checking 101. Formal Verification of SimGrid programs
- SimGrid Internal::Models. The Platform Models underlying SimGrid
- SimGrid Internal::Kernel. Under the Hood of SimGrid
- Get them from `http://simgrid.gforge.inria.fr/documentation.html`

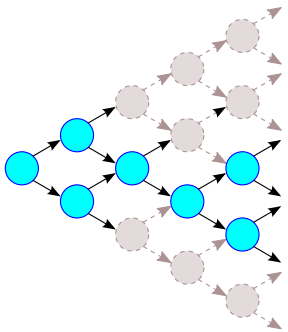
Model checking

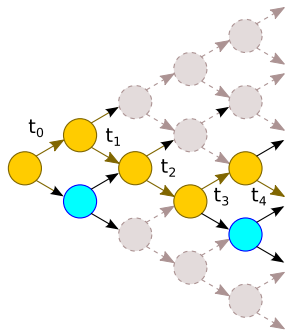
- Automated verification method (hardware or software)
- Checks whether a given model of a system satisfies a property
- Gives a counter-example in case of violation of the property



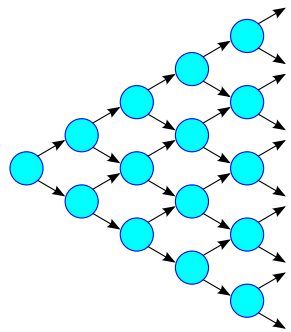
Simulation vs. Model Checking

- Simulation explores **one possible execution** of the program according to the features/limitations of the platform
- Model checking explores **all possible executions** of the program





State space with simulation



State space with model checking

- Simulation and model checking are complementary :
  - Simulation for performance evaluation
  - Model Checking for the **verification of execution properties**
  - Both run automatically

Model checking implementation with SimGrid

Step 1: Express the property that you want to assess

Step 2: Instrument your code with MC primitives

Step 3: Compile and run with the proper MC configuration options

Step 4: Analyze the produced traces (and replay the traces outside the model-checker)

Properties

Safety Property

- *"A given bad behavior never occurs"*
- Ex : no deadlock
- Work on **all states** separately
- **Assertion** on each state of the execution

Liveness property

- *"An expected behavior will happen in all cases"*
- Ex : Any process that asks a resource will obtain it eventually
- Work on **execution path**
- **Temporal logic formula** (LTL, CTL, ...)

Other properties

Non-termination, message determinism (not covered here).

1. Add the safety property

"A given bad behavior never occurs"

- Include header file of model checker:

```
#include <simgrid/modelchecker.h>
```

- Add verification of safety property with assertion in source code:

```
void MC_assert(<boolean expression of the property>)
```

- Beside of that, you keep your MSG, SMPI code inchanged
- Deadlocks are automatically detected

3. Configure, compile and execute

At configuration/compile time

- Set the `enable_model-checking` option of cmake:
- Either in ccmake, or on the command line:

```
cmake -Denable_model-checking=ON ./
```

Warning: Currently, MC builds of SimGrid are slower than standard builds.

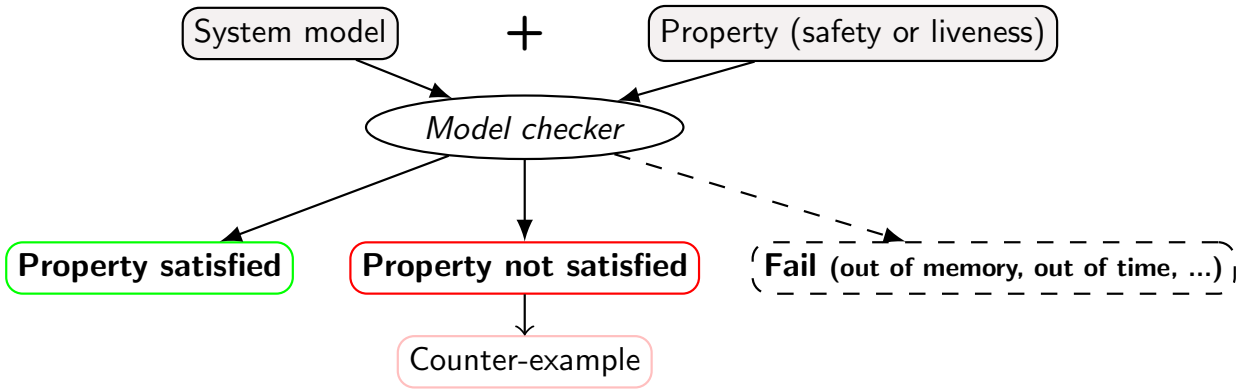
Run your code

Enabling model checking: `simgrid-mc ./<app>`

With SMPI:

```
smpi-run -wrapper "simgrid-mc" -hostfile $hostfile -platform
$platform ./<app>
```

4. Analyze the produced traces



Execution results

- If the property is satisfied, normal exit
- If the property gets violated, produces **counter-example** (execution trace)

Model Checking Statistics

- (produced in any case; you want to keep an eye on them)
- Expanded states: number of states created
- Visited states: number of states created and checked
- Executed transitions: number of enabled transitions executed

5. Replay it outside of the model-checker

It is possible to replay the execution trace outside of the model-checker:

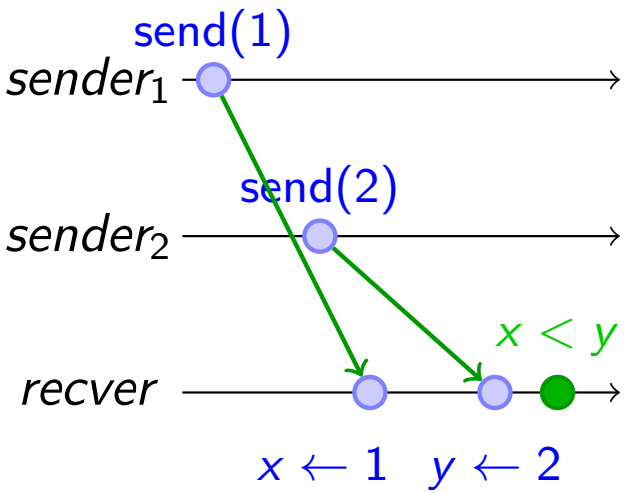
- easier to analyse (logs);
- easier to debug (GDB, valgrind, etc.).

1. pass `--cfg=model-check/record:1` to the model-checker in order to a string representation of the execution trace;
2. then pass `--cfg=model-check/replay:MY_TRACE` without the model-checker in order to replay this path outside of the model-checker.

This feature is currently experimental.

Example: Out of order receive (bugged1.c – 1/3)

- Two processes send a message to a third one
- The receiver expects the message to be in order
- This may happen...



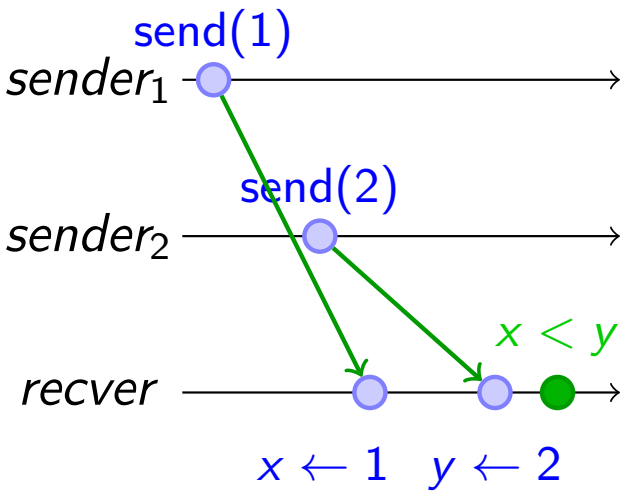
$sender_1$:
    send(1) to *recver*

$sender_2$:
    send(2) to *recver*

recver:
    $x \leftarrow$ RecvAny()
    $y \leftarrow$ RecvAny()
    assert $(x < y)$

Example: Out of order receive (bugged1.c – 1/3)

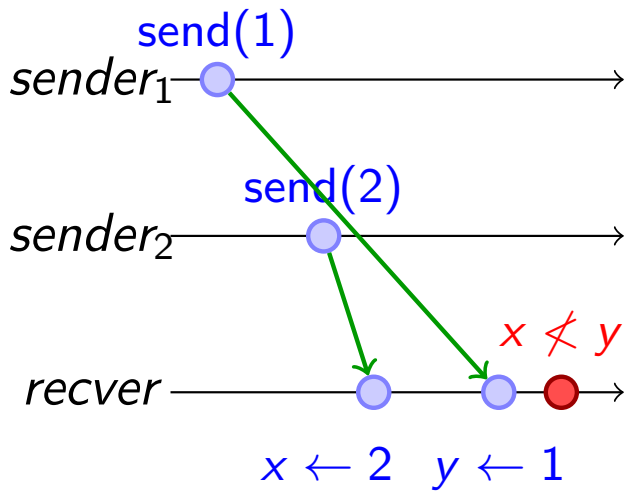
- Two processes send a message to a third one
- The receiver expects the message to be in order
- This may happen... or not



sender$_1$:
send(1) to *recver*

sender$_2$:
send(2) to *recver*

recver:
$x \leftarrow$ RecvAny()
$y \leftarrow$ RecvAny()
assert $(x < y)$



Example: Out of order receive (2/3)

int recver(int argc,char**argv)

```
m_task_t task = NULL;
MSG_task_receive(&task, "mymailbox");
MSG_task_destroy(task); task = NULL;
MSG_task_receive(&task, "mymailbox");

MC_assert(atoi(MSG_task_get_name(task)) == 2);
return 0;
```

int sender(int argc,char**argv)

```
m_task_t t = MSG_task_create(argv[1],0,10,NULL);
MSG_task_send(t, "mymailbox");
return 0;
```

int main(int argc,char**argv)

```
MSG_global_init(&argc, argv);
MSG_create_environment("platform.xml");
MSG_function_register("recver", recver);
MSG_function_register("sender", sender);
MSG_launch_application("deployment.xml");
MSG_main();
MSG_clean();
return 0;
```

Deployment File

```
<platform version="3">
  <process host="H1" function="sender">
    <argument value="1"/>
  </process>
  <process host="H2" function="sender">
    <argument value="2"/>
  </process>
  <process host="H3" function="recver"/>
</platform>
```

Platform File

```
<platform version="3">
  <AS  id="AS0"  routing="Full">
    <host id="H1" power="1"/>
    <host id="H2" power="1"/>
    <host id="H3" power="1"/>
    <!-- links and routes omitted -->
  </AS>
</platform>
```

Example: Out of order receive (3/3)

```
$ simgrid-mc ./bugged1 --cfg=model-check/record:1
[...]
**************************
*** PROPERTY NOT VALID ***
**************************
Counter-example execution trace:
Path = 1;3;1;1;2;1
[(1)server] iRecv(dst=(1)server, buff=(...), size=(...))
[(3)client] iSend(src=(3)client, buff=(...), size=(...))
[(1)server] Wait(comm=(...) [(3)client-> (1)server])
[(1)server] iRecv(dst=(1)server, buff=(...), size=(...))
[(2)client] iSend(src=(2)client, buff=(...), size=(...))
[(1)server] Wait(comm=(...) [(2)client-> (1)server])
Expanded states = 18
Visited states = 36
Executed transitions = 32
```

Example: replay of out of order receive (3/3)

```
$ ./bugged1 --cfg=model-check/replay:'1;3;1;1;2;1'
Configuration change: Set 'model-check/replay' to '1;3;1;1;2;1'
path=1;3;1;1;2;1
** SimGrid: UNCAUGHT EXCEPTION received: category: unknown error; value: 0
** Assertion prop failed
** Thrown by server() in this process
/home/user/simgrid/src/xbt/ex.c:148: [xbt_ex/CRITICAL] Assertion prop failed

**   In MC_assert() at simgrid/src/mc/mc_client_api.cpp:31 (discriminator 1)
**   In server() at simgrid/examples/msg/mc/bugged1.c:28
**   In smx_ctx_raw_wrapper() at simgrid/src/simix/smx_context_raw.c:388
Aborted
```

Liveness Properties

"An expected behavior will happen in all cases"

Modified Steps

- Step 1: express liveness property with LTL formula
  - ex: $G(r \rightarrow Fcs)$ (r = critical section requested, cs = critical section granted)
- Step 2: instrument source code for liveness verification
  - Atomic propositions of LTL formula correspond to global variables
- Step 3: run and compile with configuration options
  - `cmake -Denable_model-checking=ON ./`
  - `--cfg=model-check/property:<filename>` (using the result of `ltl2ba`)

Example, mutual exclusion algorithm.

Expressing the liveness property

1. Write a LTL formula;
2. Convert it into a Promela formula (with `ltl2ba`);
3. Feed the Promela formula to SimGridMC (`--cfg=model-check/property:myproperty.promela`)

LTL formula

```
G(r->F(cs))
```

Corresponding Promela Formula (Büchi automata)

```
never {
  T0_init :    /* init */
    if
    :: (1) -> goto T0_init
    :: (!cs && r) -> goto accept_S2
    fi;
  accept_S2 :    /* 1 */
    if
    :: (!cs) -> goto accept_S2
    fi;
}
```

Defining the propositional variables

Declaration

```
int r = 0;
int cs = 0;

int main(int argc, char *argv[])
{
  MSG_init(&argc, argv);
  MC_automaton_new_propositional_symbol_pointer("r", &r);
  MC_automaton_new_propositional_symbol_pointer("cs", &cs);
  // ...
}
```

Update

```
int client(int argc, char *argv[])
{
  // ...
  if(strcmp(my_mailbox, "1") == 0){
    r = 1;
    cs = 0;
    XBT_INFO("Propositions changed : r=1, cs=0");
  }
  // ...
}
```

Running the liveness model-checker

Launch the model checker

```
$ simgrid-mc ./bugged1_liveness platform.xml ./deploy_bugged1_liveness.xml \
   --cfg=contexts/factory:ucontext \
   --cfg=model-check/property:promela_bugged1_liveness --cfg=model-check/record:1
[...]
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
|             ACCEPTANCE CYCLE            |
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Counter-example that violates formula :
Path = 1;2;1;1;2;2;3;1;1;3;3;1;1;3;3;1;1;3;3;1;1
[(1)coordinator] iRecv(dst=(1)coordinator, buff=(...), size=(...))
[(2)client] iSend(src=(2)client, buff=(...), size=(...))
[(1)coordinator] Wait(comm=(...) [(2)client-> (1)coordinator])
[(1)coordinator] iRecv(dst=(1)coordinator, buff=(...), size=(...))
[(2)client] Wait(comm=(...) [(2)client-> (1)coordinator])
[(2)client] iRecv(dst=(2)client, buff=(...), size=(...))
[(3)client] iSend(src=(3)client, buff=(...), size=(...))
[...]
[(1)coordinator] Wait(comm=(...) [(3)client-> (1)coordinator])
[(1)coordinator] iSend(src=(1)coordinator, buff=(...), size=(...))
Expanded pairs = 23
Visited pairs = 21
Executed transitions = 21
Counter-example depth : 22
Aborted
```

`--cfg=contexts/factory:ucontext` is necessary for proper snapshot management.

Replay outside of the model-checker

Replay

```
$ ./bugged1_liveness platform.xml deploy_bugged1_liveness.xml \
  --cfg=contexts/factory:ucontext \
  --cfg=model-check/replay:'1;2;1;1;2;2;3;1;1;3;3;1;1;3;3;1;1;3;3;1;1'
(0:@) path=1;2;1;1;2;2;3;1;1;3;3;1;1;3;3;1;1;3;3;1;1
(2:client) Ask the request
(3:client) Ask the request
(2:client) Propositions changed : r=1, cs=0
(1:coordinator) CS idle. Grant immediatly
(3:client) 2 got the answer. Sleep a bit and release it
(1:coordinator) CS release. resource now idle
(3:client) Ask the request
(1:coordinator) CS idle. Grant immediatly
```

State equality detection

`--cfg=model-checking/visited:NNN`

- Saves at most NNN states, and stop exploration when revisiting one of them
- Considers globals, stack and heap (unless `MC_ignore(ptr, size)`)+in-fly msg

Use Cases beyond state space reduction

- Checking liveness properties: counter-examples are infinite acceptance paths
- Checking for non-termination (non-progressive cycle or infinite applications)

Pitfalls

- Compile your code with `-g` and `-O0`.
- Only works with `--cfg=contexts/factory:ucontext` so far

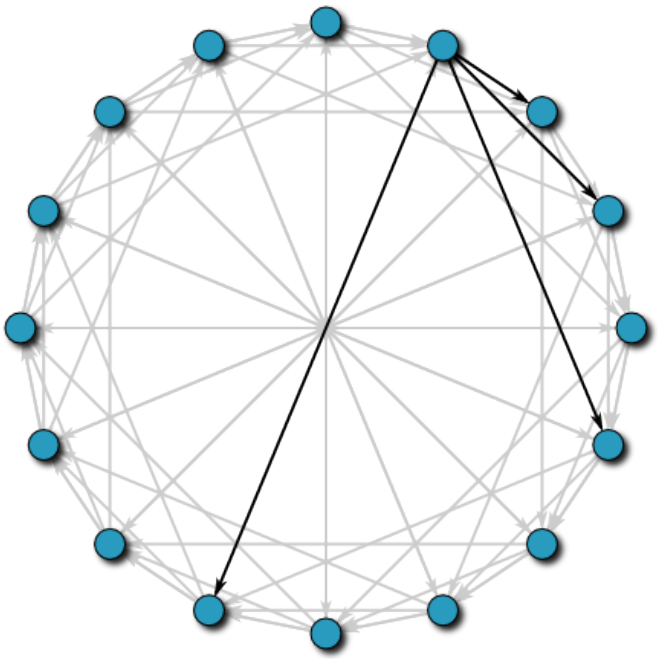
Reducing the memory consumption

- Each stack copied #snap times $\rightsquigarrow$ `--cfg=contexts/stack-size:4`
  - **Warning:** overflowing stacks will lead to unexpected segfaults
- Similarity between snapshots $\rightsquigarrow$ `--model-check/sparse-checkpoint:on`

Cannot be mixed with DPOR yet!

Chord Experiments

Chord P2P DHT protocol



SimGrid Implementation

- 500 lines of C (MSG interface)
- Suffered of bug in big instances
- Unable to spot it precisely

SimGrid MC with two Nodes

- DFS: 15600 states - 24s
- DPOR: 478 states - 1s
- Simple Counter-example!
- One line fix (prevent messages of previous rounds to mixup in the current round)

Conclusion

Model-Checking in SimGrid

- Check safety, liveness properties.
- MSG, SMPI interfaces.
- Bindings are currently not supported (Java, lua).
- This may help you to hunt hard bugs down.
- You should test it! (feedback welcomed)