On Model Checking Mechanisms

Federico Mari Igor Melatti Enrico Tronci

University of Rome "La Sapienza"

FuDiCo III
Bertinoro – June 2007

Motivations

Model checking has been very successful in:

- digital hardware verification (SMV, NuSMV, VIS)
- protocol verification (SPIN, Murphi)
- software verification (CBMC, SLAM)
- hybrid systems (Uppaal, HyTech, CMurphi)

Motivations

Model checking has been very successful in:

- digital hardware verification (SMV, NuSMV, VIS)
- protocol verification (SPIN, Murphi)
- software verification (CBMC, SLAM)
- hybrid systems (Uppaal, HyTech, CMurphi)

Here we investigate if model checking techniques can be used to verify *mechanism* designs.

Mechanism Design

Definition

Mechanism Design is a sub-field of *Game Theory*.

- It is the art of designing rules of a game to achieve a specific outcome.

Mechanism Design

Definition

Mechanism Design is a sub-field of *Game Theory*.

- It is the art of designing rules of a game to achieve a specific outcome.
- This is done by setting up a structure in which each player has an incentive to behave as the designer intends.

Mechanism Design

Definition

Mechanism Design is a sub-field of *Game Theory*.

- It is the art of designing rules of a game to achieve a specific outcome.
- This is done by setting up a structure in which each player has an incentive to behave as the designer intends.

We are interested in *Byzantine Altruistic Rational* (BAR) systems.

Scenario

Input

- A MAD protocol played by *nodes* (or *agents*)
- A set of properties to be verified

Agents are classified as:

Scenario

Input

- A MAD protocol played by *nodes* (or *agents*)
- A set of properties to be verified

Agents are classified as:

- *Byzantine*. They behave arbitrarily.

Scenario

Input

- A MAD protocol played by *nodes* (or *agents*)
- A set of properties to be verified

Agents are classified as:

- *Byzantine*. They behave arbitrarily.
- *Altruistic*. They obey to the given protocol.

Scenario

Input

- A MAD protocol played by *nodes* (or *agents*)
- A set of properties to be verified

Agents are classified as:

- *Byzantine*. They behave arbitrarily.
- *Altruistic*. They obey to the given protocol.
- *Rational*. They behave in such a way as to maximize their gain.

System History

- Model Checking technology rests on a notion of *state*.
- A state just represents the *system past history*.
- On nonterminating systems, history is infinite, thus we have an infinite number of states.

System History

- Model Checking technology rests on a notion of *state*.
- A state just represents the *system past history*.
- On nonterminating systems, history is infinite, thus we have an infinite number of states.

Since model checking typically works well for finite state systems, we restrict ourselves to histories of finite length.

Observability

- If an agent *a* knows all past actions of all agents, then *a* knows the state of all other agents.
- In other words, the system state is *observable* for each agent.
- This in general may not be true at least for two reasons:
  - an agent may not be able to observe other agents actions;
  - our finite length histories may not be *long enough* to reconstruct the state of each agent.

Parallelism: Synchronous or Asynchronous?

We can model agents behavior in two ways:

- *Synchronous*. All nodes move together (as in synchronous digital hardware).
- *Asynchronous*. Exactly one node move at each turn (as for UNIX processes).

Parallelism: Synchronous or Asynchronous?

We can model agents behavior in two ways:

- *Synchronous*. All nodes move together (as in synchronous digital hardware).
- *Asynchronous*. Exactly one node move at each turn (as for UNIX processes).

When a node moves it does not know what other nodes will do in the same round...

Parallelism: Synchronous or Asynchronous?

We can model agents behavior in two ways:

- *Synchronous*. All nodes move together (as in synchronous digital hardware).
- *Asynchronous*. Exactly one node move at each turn (as for UNIX processes).

When a node moves it does not know what other nodes will do in the same round...

$\Longrightarrow$ Synchronous!

Communication

We have a synchronous model...

Communication

We have a synchronous model...

Nodes communicate using shared variables!

About Rationality

Each rational agent will select one (or more) actions on the basis of some definition of *rationality*, as

- Nash equilibrium.
- Pareto optimality.

About Rationality

- One notion of rationality may be better suited than others.
- A *Mechanism Model Checker* should be parametric w.r.t. a (hopefully large) class of definitions of rationality.

Preliminary Experimental Results

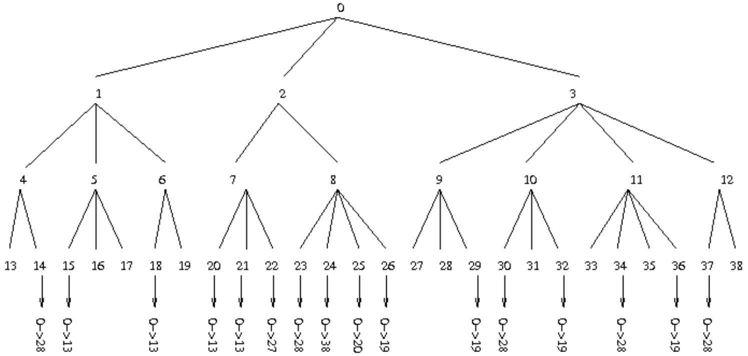
- We consider the *Terminating Reliable Broadcast* (TRB) protocol.
- We apply the assumptions seen till now:
  - Each node may be altruistic, rational or byzantine;
  - All nodes move *simultaneously*.
  - Communication between TRB nodes is implemented via shared variables (mailboxes).

Payoffs

- In order to model rational behavior, we need to define payoffs on agents actions.
- In a global state $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$, let $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$ be the actions choosen by the agents.
- We define $\mathbf{g} = \langle g_1, \ldots, g_n \rangle$, where $g_i \in \mathbb{R}$ is the payoff of agent i (if he chooses action a_i).
    - Note that payoffs are defined only on tuples of actions

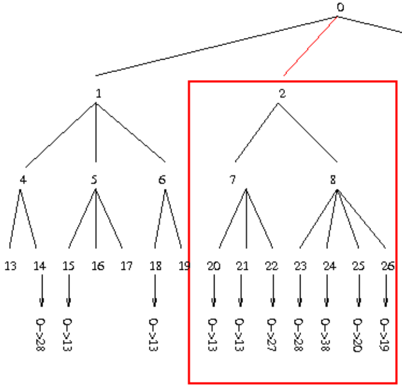
Rationality

An example ($k = 3$)

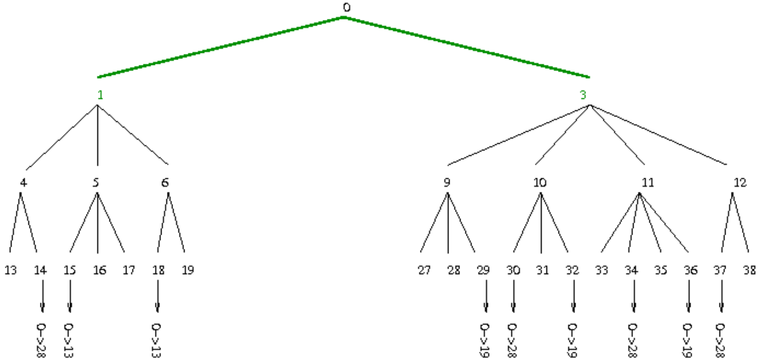


Rationality

What we cut



Rationality
The resulting system



Rationality

How *rational* nodes maximize their own utility?

- We fix a *rational horizon* k.
- Then, each rationale node will compute its set of *profitable* actions as follows.
- Let $\mathbf{s}$ be a system state and a be an allowed action for rational agent i in state $\mathbf{s}$.
- Agent i considers all possible sequences of TRB transitions with length at most k as a response to a.
- If there exists at least one possible outcome that is not worse than any other TRB sequence of at most k transitions, then agent i may *play* action a.

Properties

We intend to verify the following properties.

Agreement If a non-byzantine node delivers a message m, then all non-byzantine nodes eventually deliver m.

Termination Every non-byzantine process eventually delivers exactly one message.

Integrity If a non-byzantine node delivers m, then the sender sent m.

Non-Triviality In periods of synchrony, if the sender is non-byzantine and sends a message m, then the sender eventually delivers m.

Experimental Results

With Byzantine behavior not constrained

Parameters								Properties	
Tot	**A**	**R**	**B**	**Send**	**Lead**	**States**	**Time**	**Exp**	**Obt**
3	1	1	1	A	R	5156	1.96	Any	OK
3	1	1	1	R	B	6660	1.43	Any	NO
3	1	1	1	B	A	1443	1.11	Any	NO
5	2	2	1	A	A	16785	8.02	OK	OK
5	2	2	1	A	R	15588	7.36	OK	OK
5	2	2	1	R	R	14634	6.91	OK	OK
5	2	2	1	B	A	16785	8.07	OK	OK

Experimental Results

With Byzantine behavior not constrained

Parameters								Properties	
Tot	**A**	**R**	**B**	**Send**	**Lead**	**States**	**Time**	**Exp**	**Obt**
2	0	1	1	R	B	730	1.09	Any	OK
2	0	1	1	B	R	5276	1.31	Any	OK
3	0	2	1	R	R	5156	1.60	Any	OK
3	0	2	1	R	B	21642	3.09	Any	NO
3	0	2	1	B	R	3931	1.30	Any	NO
4	0	3	1	R	R	11622	9.05	Any	OK
4	0	3	1	B	R	18273	8.68	Any	NO
5	0	4	1	R	R	16785	93.92	OK	OK

Experimental Results

With Byzantine behavior constrained

Parameters								Properties	
Tot	**A**	**R**	**B**	**Send**	**Lead**	**States**	**Time**	**Exp**	**Obt**
5	2	1	2	A	A	1665	1.14	Any	OK
5	2	1	2	R	B	2148	1.42	Any	OK
5	1	2	2	A	R	1665	1.11	Any	OK
5	1	2	2	R	R	1665	1.12	Any	OK
5	1	2	2	B	B	8975	2.05	Any	NO
5	2	2	1	A	A	47	0.10	OK	OK
5	2	2	1	A	R	47	0.10	OK	OK
5	2	2	1	R	R	47	0.10	OK	OK
5	2	2	1	R	B	47	0.10	OK	OK
5	2	2	1	B	A	15727	5.18	OK	OK

Experimental Results

With Byzantine behavior constrained

Parameters								Properties	
Tot	**A**	**R**	**B**	**Send**	**Lead**	**States**	**Time**	**Exp**	**Obt**
4	0	3	1	R	R	47	0.10	Any	OK
4	0	3	1	R	B	47	0.10	Any	NO
4	0	2	2	R	R	15498	3.87	Any	OK
4	0	2	2	R	B	1798	1.23	Any	NO
4	0	2	2	B	R	11848	2.35	Any	NO
5	0	3	2	R	R	1665	2.14	Any	OK
5	0	3	2	R	B	2148	2.47	Any	OK

Conclusions

We have shown some preliminary considerations and experimental results on model checking mechanisms.

- Mechanism model checking can be made viable for small systems and some suitable hypotheses (e.g., *finite memory*, *global observability*).
- The notion of *rationality* to be used during verification has to be an input to the model checker.
- We expect that a model checker for mechanisms will mainly be useful to find errors (*bug hunting*) in a mechanism rather than to prove its correctness.

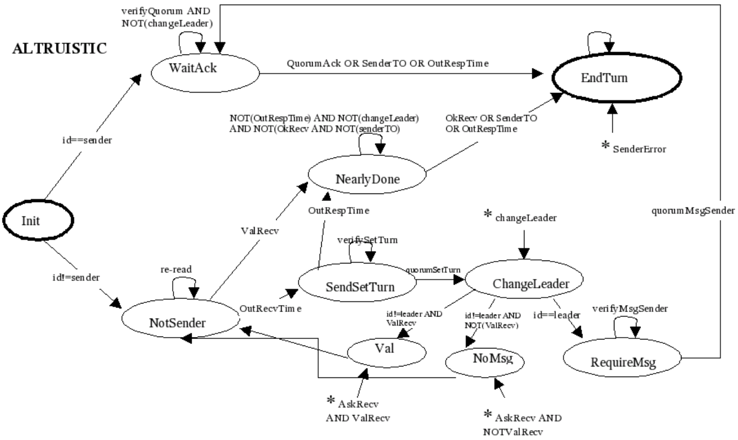
Forthcoming

We are currently working together with Lorenzo Alvisi, Allen Clement and Harry Li towards the realization of an *infinite horizon* mechanism model checker based on a discounting schema for payoffs.

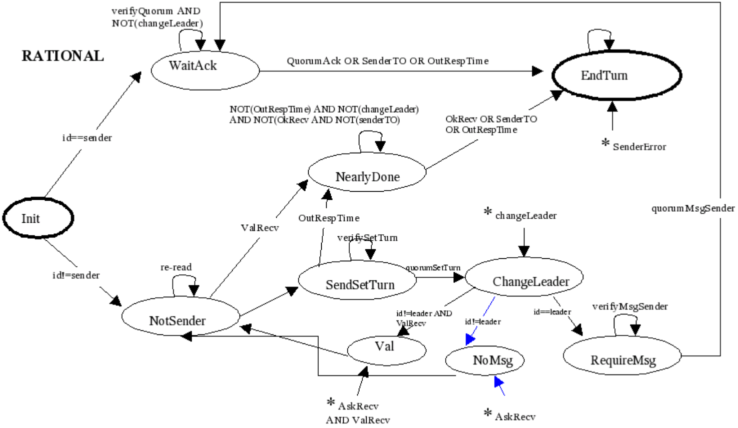
Thanks!

TRB High Level Description

Altruistic Agents

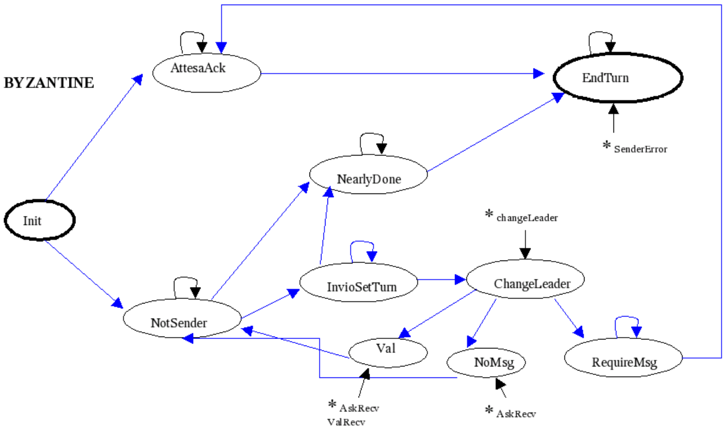


TRB High Level Description
Rational Agents



TRB High Level Description

Byzantine Agents



TRB High Level Description

Constrained Byzantine Agents