Understanding Equilibrium Properties of Multi-Agent Systems

Michael Wooldridge
(and many brilliant co-authors)
Overview

1. Multi-agent dynamics
2. Two approaches to understanding multi-agent dynamics:
   - game theoretic equilibrium analysis
   - agent-based modelling
3. Conclusions and future work
Part I

Multi-agent dynamics
Shift from the view of software as a passive servant to software as a pro-active, cooperative assistant.
Making agents a reality

- First research on agents in late 1980s/early 1990s
- Took 20 years to make it feasible
- Now we all have an agent in our pocket:
  - Siri (Apple)
  - Alexa (Amazon)
  - Cortana (Microsoft)
  - Bixby (Samsung)
Why now?

- Moore’s law: we all have a super computer in our pocket
- Advances in AI make voice understanding possible
- The super computer in your pocket is connected to the internet
When Siri met Siri

• The agent-based interface is the future of computing because **there is no alternative**
• Rich agent interfaces are still some time off
• One aspect of the agent interface has been ignored so far: why shouldn’t **your agent** talk to **my agent**?
Multi-agent systems today

- Multi-agent systems are used today
- High frequency ("algorithmic") traders are exactly that
Unpredictable Dynamics

- Unfortunately, multi-agent systems are prone to instability and have unpredictable dynamics.
- October 1987 Market Crash:
  - the “big bang” led to automated trading systems for the first time
  - simple feedback loops contributed to collapse in the market
- May 2010 Flash Crash:
  - over a 30 minute period, Dow Jones lost over a trillion dollars
  - Accenture briefly traded at a penny a share
  - markets swiftly recovered (ish)

Understanding and managing these dynamics is essential.
Part II

Formal equilibrium analysis
• One approach is to view a flash crash as a **bug**: the system is behaving incorrectly.

• **Verification** is the process of checking that a program is **correct** with respect to its **specification**.

• A long-term goal in computer science is **formal** and **automated** verification:
  - **formal**: specifications are expressed as logical formulae
  - **automated**: get a computer to check correctness

• Most successful approach to this problem has proved to be **model checking**.
Model Checking

- Industry-strength approach to automated verification
- Idea: view the state transition graph of a program $P$ as a model $M_P$ for temporal logic, and express correctness criteria as formula $\varphi$ of temporal logic
- Verification then reduces to a model checking problem:

$$M_P \models \varphi$$

- Can be (reasonably) efficiently automated, leading to many tools (SPIN, SMV, PRISM, MOCHA, MCMAS. . .)
- Most widely used logical specification languages: LTL and CTL.
MODEL

SPECIFICATION
G(req -> F resp)

MODEL CHECKER
"yes, the claim is true of the model"

"no, the claim is not true of the model: here is why"
Propositional Linear Temporal Logic (LTL)

A standard language for talking about **infinite state sequences**.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>truth constant</td>
</tr>
<tr>
<td>( p )</td>
<td>primitive propositions ( \in \Phi )</td>
</tr>
<tr>
<td>( \neg \varphi )</td>
<td>classical negation</td>
</tr>
<tr>
<td>( \varphi \lor \psi )</td>
<td>classical disjunction</td>
</tr>
<tr>
<td>( \bigcirc \varphi )</td>
<td>in the next state . . .</td>
</tr>
<tr>
<td>( \lozenge \varphi )</td>
<td>will eventually be the case that ( \varphi )</td>
</tr>
<tr>
<td>( \Box \varphi )</td>
<td>is always the case that ( \varphi )</td>
</tr>
<tr>
<td>( \varphi U \psi )</td>
<td>( \varphi ) until ( \psi )</td>
</tr>
</tbody>
</table>
Example LTL formulae

◊ ¬jetlag

eventually I will not have jetlag (a liveness property)
Example LTL formulae

\[ \Box \neg crash \]

the plane will never crash (a safety property)
Example LTL formulae

♦ drinkBeer
Example LTL formulae

I will drink beer **infinitely often**
Eventually will come a time at which I am dead forever after.

\( \Diamond \Box \text{dead} \)
Eventually will come a time at which I am dead forever after.
Example LTL formulae

$(\neg \text{friends}) \mathcal{U} \text{youApologise}$
Example LTL formulae

$$\neg \text{friends} U \text{youApologise}$$

we are not friends until you apologise
Basic Model Checking Questions

Is there some computation of the system on which $\varphi$ eventually holds? 

(reachability)

Does $\varphi$ hold on all computations of the system? 

(invariance)
Assumptions in the Standard View of Correctness

• The standard model of verification assumes an **single standard of correctness**
• The specifier is able to say “the system is correct” or “the system is not correct”
• The specifier enjoys a **privileged position**
Let’s turn to multi-agent systems, where we have multiple (semi) autonomous software agents, each acting rationally in pursuit of individual (delegated) preferences.

There is no single “owner” who can dictate a standard of correctness.

Each agent has its own standard of correctness: whether it is acting rationally in pursuit of its preferences.

“Does $\varphi$ hold on some computation of the system” doesn’t make sense if the computation does not correspond to rational choices.

Instead, we focus on equilibrium checking:
Along with many others, we adopt a \textit{game theoretic} standpoint. Appropriate analytical concepts are then \textit{game theoretic solution concepts}, in particular, \textit{equilibrium properties} such as \textit{Nash equilibrium}. Reachability and invariance are not appropriate in this setting: we are interested in whether properties will obtain under the assumption of rational action. Some computations will not arise because they involve irrational action. Key concepts: “Nash reachability” (E-Nash) and “Nash invariance” (A-Nash).
Equilibrium Checking

MODEL

PREFERENCES
\[ \leq_1, \ldots, \leq_n \]

TEMPORAL LOGIC PROPERTY
\[ \varphi \]

EQUILIBRIUM CHECKER

“yes, property \( \varphi \) is true on some Nash equilibrium”

“no, property \( \varphi \) is not true on any Nash equilibrium”
Reactive Module Games

- Practical model checkers use **high-level** model specification languages.
- **Reactive modules** is such a language:
  - a guarded command language for model specification
  - introduced by Alur & Henzinger in 1999
  - used in MOCHA, PRISM, ...
A multi-agent system is specified by a number of modules (=agents).

\[
\text{module } \text{toggle} \text{ controls } x \\
\text{init} \\
[] \top \rightarrow x' := \top \\
[] \top \rightarrow x' := \bot \\
\text{update} \\
[] x \rightarrow x' := \bot \\
[] (\neg x) \rightarrow x' := \top
\]

Thus, at every decision cycle, an agent has choices, defined by its enabled guarded commands.

A strategy for an agent defines how to make choices.
An arena $A$ is an $(n + 2)$-tuple:

$$A = \langle N, \Phi, m_1, \ldots, m_n \rangle,$$

where:

- $N = \{1, \ldots, n\}$ is a set of agents
- $\Phi$ is a set of Boolean variables
- for each $i \in N$, $m_i = \langle \Phi_i, l_i, U_i \rangle$ is an module over $\Phi$ that defines the choices available to agent $i$. 

A reactive module game is a tuple:

\[ G = \langle A, \gamma_1, \ldots, \gamma_n \rangle \]

where:
- \( A \) is an arena
- for each player \( i \) in \( A \), \( \gamma_i \) is the temporal logic goal of \( i \).

We consider LTL and CTL goals:
- LTL: players choose \textbf{deterministic} FSM strategies
  \textbf{Deterministic} strategies are \textbf{controllers}
- CTL: players choose \textbf{non-deterministic} FSM strategies
  \textbf{Non-deterministic} strategies are \textbf{supervisors}
**Decision problems**

**E-Nash:**
Given: Game $G$, LTL formula $\varphi$.
Question: Does $\varphi$ hold on some NE of $G$?

**A-Nash:**
Given: Game $G$, LTL formula $\varphi$.
Question: Does $\varphi$ hold on all NE of $G$?

**Equilibrium Checking:**
Given: Game $G$, strategy profile $\bar{\sigma}$.
Question: Is it the case that $\bar{\sigma}$ forms a NE of $G$?
Decision Problems

Theorem

For LTL:
1. \text{E-NASH} \text{ and } \text{A-NASH} \text{ are } 2\text{EXPTIME}-complete, while \text{EQUILIBRIUM CHECKING} \text{ is } \text{PSPACE}-complete.
2. \text{For CTL, EQUILIBRIUM CHECKING is } \text{EXPTIME}-complete
An Example

- $N = \{1, 2\}$,
- player 1 controls $\{p\}$
- player 2 controls $\{q\}$
- $\gamma_1 = \Box \Diamond (p \leftrightarrow q)$
- $\gamma_2 = \Box \Diamond \neg (p \leftrightarrow q)$

These strategies form a NE.
We have implemented a tool for equilibrium checking CTL RMGs.

Takes as input:

1. arena $A$ specified in RML
2. goals $\gamma_1, \ldots, \gamma_n$ for each player, specified in LTL

computes E-NASH and A-NASH problems

automata-theoretic approach
Part III

Agent-based Modelling
Agent-based models

- Equilibrium analysis only makes sense for small systems
- For larger systems, our best hope is **simulation**
- Agent-based modelling directly models the actors in a system and individual trades
- Possible now because of:
  - availability of data
  - availability of compute resources
Agent-based modelling challenges

- Getting meaningful simulations
- “Magic numbers” in simulation
- Calibration – when does a $ in the model correspond to a $ in the real world
- Qualitative **versus** quantitative interpretations
Exploring effects of leverage and crowding on flash crashes

- “Leverage management practice by funds creates a contagion channel for flash crash propagation”
- “Speed of flash crash propagation depends on leverage, capital and network topology (non-monotonic)”
- “Crowding can be beneficial to systemic stability, but only if fund-asset allocations are non-uniform”
Conclusions & future work

- The agent paradigm, a 30 year dream for AI, is now a reality, powered by Moore’s law, the Internet, and
- The next step for the agent paradigm is to put agents in touch with other agents
- But multi-agent systems – particularly large ones – have unpredictable dynamics
- We need to be able to model and understand such dynamics
- Formal equilibrium analysis is one approach
- Agent-based modelling is another.