Generalising the Dining Philosophers problem: Competitive dynamic resource allocation in multi-agent systems

Valentin Goranko
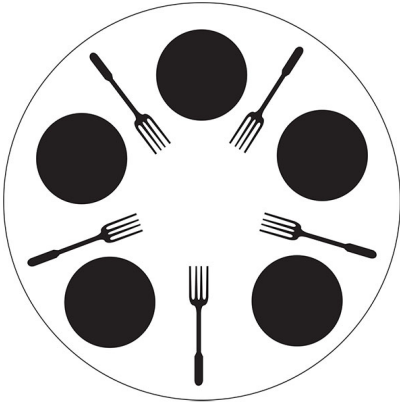
Department of Philosophy, Stockholm University

Joint work with Riccardo De Masellis, Stefan Gruner, and Nils Timm

FMAI 2019
Rennes, May 2, 2019

Dining philosophers problem

Generalising:
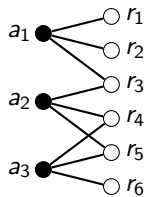
- Philosophers are **agents**
- Forks are **resources**

Generic dining philosophers games

A **GDP game** is a tuple

$$\mathcal{G} = (Agt, Res, d, Acc, Act, Rules) \text{ where:}$$

- *Agt* is a set of **agents**;

- *Res* is a set of **resource units**;

- $d : Agt \rightarrow \mathbb{N}^+$ is a **demand function**;

- $Acc \subseteq Agt \times Res$ is a **resource accessibility relation**.

- *Act* is a set of **possible actions**;

- *Rules* is a set of **transition rules**;

Example

$$d(a_i) = 2$$
for $i \in \{1, 2, 3\}$

The intended goal for each agent a_i is **to acquire** $d(a_i)$ **resource units** (needed to carry out its task).

The actions and rules will be specified later.

Objective of this work

To develop a formal framework for specifying and verifying relevant individual and collective strategic abilities of agents in GDP games, such as "no deadlocks", or "no starvation", or e.g.:

"a_1 and a_2 can act collaboratively so as to guarantee that a_2 reaches its goal (collects $d(a_2)$ resource units) infinitely often?"

Actions

Actions:

- req_r^a agent a requests resource r;

- rel_r^a agent a releases resource r;

- rel_{all}^a agent a releases all resources it holds;

- $idle^a$ agent a does nothing.

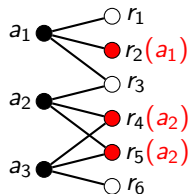An **action profile** is a mapping $ap : Agt \rightarrow Act$.

Configurations

Example

Given $\mathcal{G}$ as before the figure

A possible *state* of the game is called a **configuration**

$$c : Res \to Agt^+$$

graphically represents configuration where r_2 is held by a_1, r_4 is held by a_2 and r_5 by a_2.

Remark

The number of configurations in a GDP game is, in general, exponential in the number of resources.

Transition rules and system dynamics

Given a configuration c and an action profile ap, (c, ap, c') is a **step** if:

1. ap **can be executed** in c, meaning:
   - agents can request only resources available in c;
   - if an agents a holds number $d(a)$ resources, it must perform rel_{all}^a;

2. and the **resulting configuration** c' is such that:
   - the released resources become available in c';
   - if a resource is requested by one agent only, than that agent acquires it, otherwise no agent gets it.
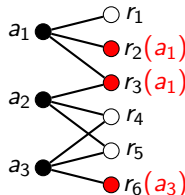
Example
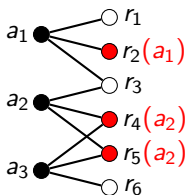
$$ap(a_1) = \text{req}_{r_3}^{a_1}$$

$$ap(a_2) = \text{rel}_{all}^{a_2}$$

$$ap(a_3) = \text{req}_{r_6}^{a_3}$$

$$\Longrightarrow$$

Configuration graph

- **Transition function** of $\mathcal{G}$ is the set $\rho(\mathcal{G})$ of all game steps;

- $\mathfrak{G} = (\textit{Conf}, \rho(\mathcal{G}))$ is the **configuration graph** of $\mathcal{G}$

- a **play** is an infinite sequence of configurations in $\mathfrak{G}$.

Competition and cooperation in GDP games

A GDP game is a *both competitive and cooperative* scenario,
where agents may, but need not to, cooperate in pursuing their goal.

- On the one hand, each agent is interested in reaching their individual goal.

- However, that may become impossible if each agents acts selfishly (follows a greedy strategy), as that may lead to blocking resources.

- Thus, it is sometimes preferable for agents to cooperate by releasing resources before having reached their individual goals.

- Furthermore, some of them may wish to join forces and act in a coordinated way, as a coalition.
  That, inter alia, makes the analysis of GDP games quite non-trivial.

- Hence, the need for formal specification and algorithmic verification.

Logic for verifying GDP games

Our language $\mathcal{L}_{\mathrm{GDP}}$ is a slight variation of ATL:

$$\varphi ::= g_{a_i} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle\!\langle \mathbf{A} \rangle\!\rangle \, \mathsf{X} \, \varphi \mid \langle\!\langle \mathbf{A} \rangle\!\rangle \, \mathsf{G} \, \varphi \mid \langle\!\langle \mathbf{A} \rangle\!\rangle \, \varphi_1 \, \mathsf{U} \, \varphi_2$$

where $\mathbf{A} \subseteq Agt$

and g_{a_i} means that agent a_i currently holds at least $d(a_i)$ resource units (and has, therefore, reached its goal).

Strategies

For our language it suffices to consider *positional strategies*.

- a **(positional) strategy** for an agent a

$$\sigma_{\mathbf{a}} : Conf \rightarrow Act$$

  which prescribes *executable* actions to the agent.

- a **joint (positional) strategy** for $A = \{a_1, \ldots, a_r\} \subseteq Agt$:

$$\sigma_{\mathbf{A}}(\sigma_{a_i}, \ldots, \sigma_{a_r})$$

 is a tuple of individual strategies σ_{a_i}, for each $a_i \in A$.

Function $out(c, \sigma_A)$ returns the set of all plays in $Conf^{\omega}$ that can occur when agents in A follow the joint strategy σ_A from configuration c on.

Formal semantics

$\mathcal{L}_{\mathrm{GDP}}$ is interpreted in GDP games as follows:

- $\mathfrak{G}, c \models g_{a_i}$ iff the number of resources a_i holds is $\geq d(a_i)$;

- $\wedge$, $\vee$ and $\neg$ are treated as usual;

- $\mathfrak{G}, c \models \langle\!\langle A \rangle\!\rangle \, \mathsf{X} \, \varphi$ iff there is a joint strategy σ_A, such that $\mathfrak{G}, \pi[1] \models \varphi$ for every path $\pi \in out(c, \sigma_A)$;

- $\mathfrak{G}, c \models \langle\!\langle A \rangle\!\rangle \, \mathsf{G} \, \varphi$ iff there is a joint strategy σ_A, such that $\mathfrak{G}, \pi[i] \models \varphi$ for every path $\pi \in out(c, \sigma_A)$ and for every $i \in \mathbb{N}$;

- $\mathfrak{G}, c \models \langle\!\langle A \rangle\!\rangle \, \varphi_1 \, \mathsf{U} \, \varphi_2$ iff there is a joint strategy σ_A, such that for every path $\pi \in out(c, \sigma_A)$: there exists $i \geq 0$ such that $\mathfrak{G}, \pi[i] \models \varphi_2$ and $\mathfrak{G}, \pi[j] \models \varphi_1$ for all j such that $0 \leq j < i$.

Example

$$c_1 = $$
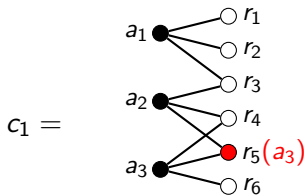
$$\mathfrak{G}, c_1 \models \langle\langle a_1, a_3 \rangle\rangle\, \mathsf{G}\,(\langle\langle a_1 \rangle\rangle\,(\neg g_{a_2})\,\mathsf{U}\,g_{a_1})$$

Model checking

ATL provides an algorithm for solving the global model checking problem:

Inputs:
- formula φ
- a GDP problem $\mathcal{G}$

Output:
- the **state extension** of φ in $\mathfrak{G}$

$$[\![\varphi]\!]_{\mathfrak{G}} = \{c \in \textit{Conf} : \mathfrak{G}, c \models \varphi\}$$

Complexity

The ATL algorithm for global model checking problem applied to $\mathcal{L}_{\mathrm{GDP}}$ has worst-case time complexity exponential in the number of resources.

Since the number of resources can be large, this can be a problem.

Can we be more efficient?

Idea:

- Define a suitable **abstraction**: equivalence relation $\sim$ on configurations, that preserves truth of $\mathcal{L}_{\mathrm{GDP}}$ formulae;

- build the global model checking procedure to use that abstraction.

A natural abstraction

Observation:

- our logic cannot distinguish on atomic level configurations where agents hold the same number of resources
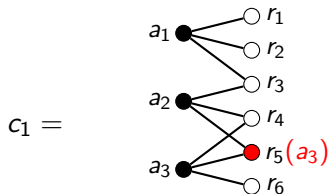
So, can we use

$$c_i \sim_\# c_j$$

iff

for each agent a,
the number of resources a holds in c_i
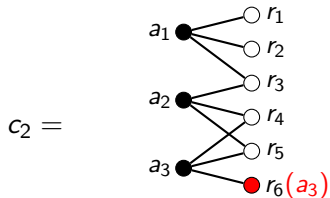is the same it holds in c_j?

No! This is too coarse.

The abstraction $\sim_\#$ is too coarse

Example

$$c_1 \sim_\# c_2$$

$\mathfrak{G}, c_1 \models \langle\!\langle a_3 \rangle\!\rangle \, \mathsf{X} \, g_{a_3}$ True

$\mathfrak{G}, c_2 \models \langle\!\langle a_3 \rangle\!\rangle \, \mathsf{X} \, g_{a_3}$ False

A correct abstraction

A finer abstraction is required.

1. We first define an equivalence relation on resources

$$\mathbf{r_i \approx r_J}$$

iff
r_i and r_j are accessible by the same subset of agents

2. We then define

$$\mathbf{c_1 \sim c_2}$$

iff
for each agent a and
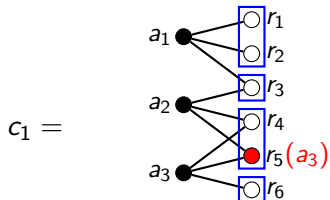for each equivalence class of resource $R \in Res/\approx$
the number of resources from R
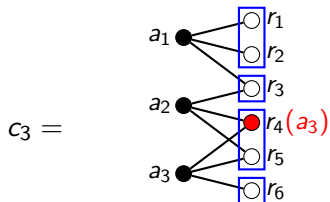that a holds in c_1 is the same as in c_2

A sound and complete abstraction

Example

$$c_1 \approx c_2$$

$c_1 =$

$\mathfrak{G}, c_1 \models \langle\langle a_3 \rangle\rangle \, \mathsf{X} \, g_{a_3}$ True

$c_3 =$

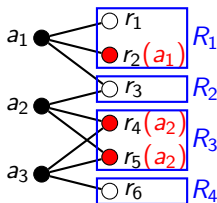$\mathfrak{G}, c_3 \models \langle\langle a_3 \rangle\rangle \, \mathsf{X} \, g_{a_3}$ True

Interval expressions

We symbolically represent sets of configurations with expressions:

$$\alpha ::= \bigwedge_{a \in Agt} \bigwedge_{R \in \mathcal{R}} (a, R)[l_R^a, l_R^a] \mid \alpha_1 \vee \alpha_2$$

and $\|\alpha\|_{\mathfrak{G}}$ denotes the set of configurations "contained" in α

Example

is contained in:

$$(a_1, R_1)[1, 1] \wedge (a_1, R_2)[0, 0] \wedge$$
$$(a_2, R_2)[0, 0] \wedge (a_2, R_3)[2, 2] \wedge$$
$$(a_3, R_3)[0, 0] \wedge (a_3, R_4)[0, 0]$$

A symbolic model checking algorithm for $\mathcal{L}_{\mathrm{GDP}}$

We develop a *symbolic* global model checking algorithm for $\mathcal{L}_{\mathrm{GDP}}$.

Given
- a game $\mathcal{G}$
- a formula φ

it returns
- the **interval constraint expression** $\alpha(\mathcal{G}, \varphi)$

Theorem

For each game $\mathcal{G}$ and formula $\varphi \in \mathcal{L}_{\mathrm{GDP}}$ we have:

$$c \in [\![\varphi]\!]_{\mathfrak{G}} \text{ iff } c \in \alpha(\mathcal{G}, \varphi)$$

Complexity

The symbolic global model checking algorithm runs in time at most *double exponential in the number of agents* but *polynomial in the number of resources*.

Concluding remarks: future work

- To obtain more refined complexity results.
  (The double exponential case seems to never actually happen.)

- Can we do better? Is our algorithm optimal?

- To extend the framework to one where resources are autonomous
  agents themselves. *Clients/Bankers problem*.

- To explore the case with agents' incomplete information.

THE END

Questions?