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

Riccardo De Masellis[1], Valentin Goranko[1,2], Stefan Gruner[3], and Nils Timm[3]

[1] Stockholm University, Sweden
{riccardo.demasellis, valentin.goranko}@philosophy.su.se
[2] University of Johannesburg, South Africa (visiting professorship)
[3] University of Pretoria, South Africa
{sgruner, ntimm}@cs.up.ac.za

Abstract. We consider a new generalisation of the Dining Philosophers problem with a set of agents and a set of resource units which can be accessed by them according to a fixed graph of accessibility between agents and resources. Each agent needs to accumulate a certain (fixed for the agent) number of accessible resource units to accomplish its task, and once it is accomplished the agent releases all resources and starts accumulating them again. All this happens in succession of discrete 'rounds' and yields a concurrent game model of 'dynamic resource allocation'. The Alternating time Temporal Logic (ATL) can thus be used to specify and verify important properties, such as goal achievability, fairness, deadlock, starvation, etc. However, the sizes of the resulting explicit models are generally exponential both in the number of resources and the number of agents, which makes model checking generally intractable. In this paper we develop an abstract representation of the dynamic resource allocation models and present a symbolic model checking procedure for ATL specifications working in such abstractions. This reduces the time complexity of model checking to polynomial in the number of resources.

Keywords: Generalized dining philosophers · ATL

1 Introduction and Related Work

The *dining philosophers problem* [11] is a well-established example for illustrating the problems of resource allocation in distributed computing [7]. In its original version it involved 5 'philosophers' sitting at a round table, where 1 fork is placed between each pair of neighboured philosophers and they share it. The description of this scenario is well-known [11] and we do not need to repeat it here. However we should point out that we assume in this paper that philosophers' actions, thinking or eating, are instantaneous. The problem is to design a distributed protocol for picking up forks that ensures that each philosopher will get to eat repeatedly. The relevant properties here are *liveness*, as well as *deadlock-* and *starvation-freedom*. Technically the 'philosophers' represent processes of a distributed system, 'forks' are shared resources, and 'eating' is performing a computational task.

Since the introduction of the original problem, several generalisations have been published, for example the *drinking philosophers* [6], where each resource is still shared between two philosophers, but the resource distribution can now be arbitrary. Accordingly, a philosopher may have access to more than two resources and may also have more than two neighbours. The solution of [8] also uses communication between neighbours in order to determine on what each philosopher can do next. The *generalised* dining philosophers problem [12] permits that one resource may be shared between more than two philosophers. Still each philosopher has access to exactly two resources and needs these two resources in order to eat. The solution of [12] implements each philosopher as a random-based algorithm whereby all these algorithms run asynchronously. The randomised solution guarantees deadlock-freedom. The original problem has also been generalised by allowing mobility. *Mobile philosophers* [14] are able to move around the table, which results in a resource accessibility relation that changes over time. In [9] a solution to the mobile philosophers problem is presented that ensures mutual exclusion, liveness and self-stabilisation. The solution requires that the philosophers follow a certain access pattern that determines the orders of requests and the direction of moving around. All these problems fall under the broader category of *resource allocation problems*.

In this paper we present a *new generalisation* of the dining philosophers problem, involving a set of agents and a set of resource units which can be accessed by them according to a fixed bipartite graph of accessibility between agents and resources. Each agent needs to accumulate a certain (fixed for the agent) number of accessible resource units to accomplish its task. Once it is accomplished the agent releases all resources and starts accumulating them again. Thus, all agents compete for resources and attempt simultaneously to acquire them in a distributed way, in a discrete succession of rounds. In contrast to the drinking philosophers problem and the so far proposed solutions for it, we assume in our approach that the total resource demand of each philosopher remains the same in each round. However, we do not assume the restriction that a resource may be shared only between two philosophers. Moreover, our agent-based scenario does not involve communication between philosophers (other than possibly coordinating on their joint strategy when acting as a team). Thus, our problem formulation is a further generalisation of the one in [12] in the sense that we allow for arbitrary demands and arbitrary access topologies.

The scenario described above can be naturally modelled as a *concurrent game model* [4], [10, Ch. 9] of 'dynamic resource allocation'. For such models we use a version of the *alternating time temporal logic* ATL [4] to specify and verify their important properties, such as goal achievability, fairness, etc. However, the sizes of the resulting explicit models are generally exponential both in the number of resources and the number of agents, which makes explicit model checking generally intractable. To avoid such 'blow-up' we use an *abstract representation* of the dynamic resource allocation models and develop a symbolic model checking procedure for ATL. Working in such abstractions reduces the time complexity of model checking to polynomial in the number of resources.

In addition to these technical results, to our best knowledge our work is the first that presents an agent-based solution to a *generalisation* of the dining philosophers

problem, even though a multi-agent approach to the *classical* problem was used in [5]. Besides modelling each philosopher as an agent, the solution in [5] uses an additional 'manager' agent (scheduler) who grants permission to acquire and release resources. With the scheduler as a central agent this approach can result in reduced parallelism in comparison to a decentralised solution. By contrast, our solution is fully distributed, without any central authority.

A similar definition of the generalised dining philosophers problem can be found in [15]. There, philosophers and resources are nodes of a bipartite graph that characterises the accessibility of resources. In contrast to our approach, however, a philosopher that requests a resource is blocked until he is eventually able to acquire it. The solution of [15] is based on (NP-complete) graph-colouring and guarantees robustness, deadlock- and starvation-freedom.

Lastly, some works on *resource-bounded reasoning* [1, 3, 2] are conceptually related to the approach presented here, although they are quite different in the framework and proposed solution. Indeed, while our reasoning tasks focus on how to obtain resources, resource-bounded reasoning abstracts this aspect away and is about which properties can be guaranteed given a set of resources and assuming that actions have costs.

2 Generalisation of the Dining Philosophers Problem

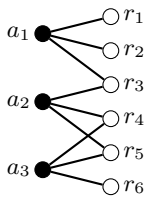
Definition 1 (Generalised Dining Philosophers Game (GDP)). *A GDP game is a tuple:*

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

- $Agt = \{a_1, \ldots, a_n\}$ *is a non-empty set of* ***agents****;*
- $Res = \{r_1, \ldots, r_m\}$ *is a non-empty set of* ***resource units*** *(of the same type);*
- $d : Agt \to \mathbb{N}^+$ *is a* ***demand function*** *defining the number of resources that each agent needs in order to carry out its tasks;*
- $Acc \subseteq Agt \times Res$ *is an* ***accessibility relation*** *denoting which resources agents can access. The set of resources that are accessible to an agent a is $Acc(a) = \{r \in Res \mid Acc(a, r)\}$, and we always assume:*
  - $|Acc(a)| \geq d(a)$ *for each* $a \in Agt$ *and*
  - $\forall r \in Res, \exists a \in Agt. Acc(a, r)$.

The above definition statically describes a game that is played in turns by agents, as explained later. The scenario is similar in spirit to the dining philosophers problem, where agents are philosophers and resources are forks. However here: *(i)* each resource can be shared by any set of agents (not only by two adjacent philosophers) as specified by Acc relation and *(ii)* each agent needs a generic (fixed for the agent) number of resources (not specifically two) in order to carry out its abstract task as described by the demand function d.

Example 1. The graph describes agents a_1, a_2, a_3, the resources $r_1 - r_6$, and the accessibility relation of a GDP game $\mathcal{G}$.
The game is fully specified once the demand function d is defined, e.g. $d(a_i) = 2$ for each $i = 1, 2, 3$.



Intuitively, the objective of each agent a_i is to acquire, gradually over time, the number of resources it needs by means of 'request' actions. Actually, each agent can perform several types of actions, as formalised below.

Definition 2 (Actions). *Given a GDP game $\mathcal{G}$, the **set of actions** Act is the union of the following types of actions:*

- ***request actions**:* $\{\mathsf{req}^a_r \mid a \in Agt, r \in Acc(a)\}$
- ***release actions**:* $\{\mathsf{rel}^a_r \mid a \in Agt, r \in Acc(a)\}$
- ***release-all actions**:* $\{\mathsf{rel}^a_{all} \mid a \in Agt\}$
- ***idle actions**:* $\{\mathsf{idle}^a \mid a \in Agt\}$

The game is played in rounds, each of which consists of a tuple of (simultaneously executed) actions, one for each agent. Before any round, each agent holds a certain number of resources, and it can: request an accessible resource; release a resource that it holds; carry out its task, and then release all its resources at the same round; or idle.

Disallowing requests for multiple resources at the same time (or, release of multiple resources but not all of them) is a purely conceptual choice to keep the framework simple and essential: however, the results presented in the paper carry over without essential complications when the restrictions above are dropped.

The dynamics of the game is thus given in terms of a system of possible transitions between *configurations* over time. Configurations describe which resources each agent currently possesses.

Definition 3 (Configurations). *Given a GDP game $\mathcal{G}$, a **configuration** in $\mathcal{G}$ is a function $c : Res \to Agt^+$, where $Agt^+ = Agt \cup \{null\}$. If $c(r) = a$ then resource r is assigned in c to agent a. If $c(r) = null$ then r is unassigned in configuration c. We denote by c_0 the **initial configuration**, where $c_0(r) = null$ for each $r \in Res$, and by $Conf$ the set of all possible configurations in $\mathcal{G}$.*

Example 2. Consider $\mathcal{G}$ as in Example 1. Here is a possible configuration c in $\mathcal{G}$: $c(r_1) = null$; $c(r_2) = a_1$; $c(r_3) = null$; $c(r_4) = a_2$; $c(r_5) = a_2$ and $c(r_6) = null$.

In each configuration, only a subset of all actions is executable by each agent.

Definition 4 (Actions' Availability). *The **availability of actions to agents** at configurations is a function $av : Conf \times Agt \to 2^{Act}$ defined component-wise as follows, for each $c \in Conf$ and $a \in Agt$:*

1. *if $|c^{-1}(a)| \geq d(a)$ then $av(c, a) = \{\mathsf{rel}^a_{all}\}$;*
2. *otherwise:*
   (a) $\mathsf{rel}^a_{all} \notin av(c, a)$*;*
   (b) $\mathsf{req}^a_r \in av(c, a)$ *iff* $c(r) = null$*;*
   (c) $\mathsf{rel}^a_r \in av(c, a)$ *iff* $c(r) = a$*;*
   (d) $\mathsf{idle}^a \in av(c, a)$*.*

Intuitively: 1 and 2a say that when, and only when, agent a holds all the number of resources that it needs for achieving its goal, a *must* release them all; 2b says that a can request resource r iff r is accessible by a and is currently available; 2c states that a can release r iff it currently has it; and 2d says that a can always idle, unless it must release its resources.

Example 3. Consider the configuration c defined in Example 2. Then $av(c, a_1) = \{\mathsf{req}^{a_1}_{r_1}, \mathsf{rel}^{a_1}_{r_2}, \mathsf{req}^{a_1}_{r_3}, \mathsf{idle}^{a_1}\}$; $av(c, a_2) = \{\mathsf{rel}^{a_2}_{all}\}$ and $av(c, a_3) = \{\mathsf{req}^{a_3}_{r_6}, \mathsf{idle}^{a_3}\}$.

Note that agents may request resources only if they are *currently* available in the configuration (no waiting queues). This has two implications: *(i)* agents cannot yet request resources that are about to be released by another agent and *(ii)* an agent that has just reached its goal and has just released all of its resources can request again any of these resources in the next turn, i.e., as soon as they are available again to everyone. We assume here full knowledge/observability by all agents of both the game and the current configuration: they know the other agents, their demand function, the accessibility relation as well as the current configuration. However, they cannot observe the actions taken by the others at any given round, until that round is completed.

Definition 5 (Action Profile). *Given a game $\mathcal{G}$ an **action profile** in $\mathcal{G}$ is a mapping $ap : Agt \to Act$. We denote with AP the set of all action profiles. Moreover, we say that ap is **executable** at $c \in Conf$ when for each $a \in Agt$ we have $ap(a) \in av(c, a)$.*

Given a configuration c and an action profile ap, we define the respective successor configuration c' and a **game step** (c, ap, c') component-wise, as follows. Firstly, in order for (c, ap, c') to be a legitimate game step, ap must be *executable* at c. Then, an agent a will keep holding in c' a resource r that it has in c unless it releases it with its action in ap, resulting in r being unassigned in c'. Lastly, an agent a will acquire a requested resource r in that step if and only if r is available and a is the only one requesting r; otherwise, i.e., whenever there is a request conflict for r, it remains *unassigned* for the sake of a *fully deterministic transition function* (see below). This choice is again to keep the framework simple: having nondeterministic evolutions does not affect the abstraction presented later, which only depends on configurations and the logical language.

Definition 6 (Game Steps and Game Transition Function). *Given a GDP game $\mathcal{G}$, a **game step** in $\mathcal{G}$ is a triple (c, ap, c') denoted by $c \xrightarrow{ap} c'$, where $c, c' \in Conf$ and:*

(i) ap is an executable action profile at c in $\mathcal{G}$, and
(ii) c' is such that for each $r \in Res$:

1. *if $c(r) = null$, then:*
   (a) if $((\exists a.ap(a) = \mathsf{req}^a_r \wedge \forall a'.a' \neq a \to ap(a') \neq \mathsf{req}^{a'}_r)$ then $c'(r) = a$);
   (b) otherwise $c'(r) = c(r) = null$;
2. *otherwise, let $c(r) = a$ for some (unique) agent a; then:*
   (a) if $(ap(a) = \mathsf{rel}^a_r \vee ap(a) = \mathsf{rel}^a_{all})$ then $c'(r) = null$;
   (b) otherwise $c'(r) = c(r) = a$.

*The **game transition function** of $\mathcal{G}$ is the set $\rho(\mathcal{G})$ of all game steps in $\mathcal{G}$.*

Example 4. Consider the following action profiles in Example 2:
$ap'(a_1) = \mathsf{idle}^{a_1}$; $ap'(a_2) = \mathsf{rel}^{a_2}_{all}$; $ap'(a_3) = \mathsf{req}^{a_3}_{r_6}$
$ap''(a_1) = \mathsf{req}^{a_1}_{r_3}$; $ap''(a_2) = \mathsf{req}^{a_2}_{r_3}$; $ap''(a_3) = \mathsf{req}^{a_3}_{r_5}$

The respective resulting configurations from performing ap' and then ap'' at configuration c are:
$c'(r_1) = null$; $c'(r_2) = a_1$; $c'(r_3) = null$; $c'(r_4) = null$; $c(r_5) = null$; $c'(r_6) = a_3$.
$c''(r_1) = null$; $c''(r_2) = a_1$; $c'(r_3) = null$; $c'(r_4) = null$; $c(r_5) = a_3$; $c'(r_6) = a_3$.

Plays in a GDP game $\mathcal{G}$ are (infinite) sequences of game steps in $\mathcal{G}$, defined by means of the transition system $\mathfrak{G} = (Conf, \rho(\mathcal{G}))$, which we call **configuration graph of** $\mathcal{G}$. We also define the **local configuration graph of** $\mathcal{G}$ **generated by** c_0, that is the restriction $(\mathfrak{G}, c_0) = (Conf(c_0), \rho(\mathcal{G}))$ of $\mathfrak{G}$, where $Conf(c_0)$ is the set of only those configurations in $Conf$ which are reachable from the initial configuration c_0 by $\rho(\mathcal{G})$.

Proposition 1. *The size of* $(\mathfrak{G}, c_0)$ *(hence, also of* $\mathfrak{G}$*) is, in general, exponential in the number of resources in* $\mathcal{G}$*.*

Proof. Take $\mathcal{G}$ where $\forall a, r.Acc(a, r)$ and $\forall a.d(a) = |Res|$. Then each assignment of resources to agents is a reachable configuration, thus $|Conf| = |Agt|^{|Res|}$.

3 Logic to Specify and Verify GDP Games

Agents in GDP games may, but need not, cooperate in pursuing their goals, which may be *positive*, i.e. eventually acquiring the necessary number of resources to achieve their individual goals, or *negative*, i.e. preventing others from achieve their goals, or combined and more complex. Thus, our choice of language and formalism $\mathcal{L}_{\text{GDP}}$ for specification and verification of GDP games is naturally a (slight) variation of the *alternating time temporal logic* ATL [4], which allows to express temporal properties φ parameterised by a set of agents A in multi-agent games. The main construction in ATL is $\langle\langle A\rangle\rangle \varphi$, the intuitive meaning of which is that the coalition of agents in A has a joint strategy to collaborate in order to achieve φ, no matter how the opponent agents in $Agt \setminus A$ may counter-act. As customary when reasoning at this level of abstraction, we do not focus on how agents in the same coalition should coordinate to achieve the objectives, but we assume such a coordination mechanism is already in place.

Definition 7 (Syntax). *The formulae of* $\mathcal{L}_{\text{GDP}}$ *are defined as follows:*

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

where $a_i \in Agt$*,* $A \subseteq Agt$*. We may also write* $\langle\langle a_1, ..., a_i\rangle\rangle$ *instead of* $\langle\langle\{a_1, ..., a_i\}\rangle\rangle$*.*

The atomic formula g_{a_i} expresses the claim that agent a_i has reached the number of resources it needs (given by $d(a_i)$) to achieve its goal. All other boolean connectives and the standard temporal operators *next* X, *always* G, and *until* U have the usual semantics (cf. e.g. [10]).

Definition 8 (Positional Strategy). *Let* $\mathcal{G}$ *be a game, for each agent* $a \in Agt$*,* a (positional) strategy for a *is a function* $\sigma_a : Conf \to Act$ *such that* $\forall c.\sigma_a(c) \in av(c, a)$*. Given* $A = \{a_1, \ldots, a_r\} \subseteq Agt$*,* a joint strategy for A *is a tuple of strategies* $\sigma_A(\sigma_{a_i}, \ldots, \sigma_{a_r})$ *one for each* $a_i \in A$*.*

The function $out(c, \sigma_A)$ returns the set of all paths in $Conf^\omega$ that can occur when agents in A follow the joint strategy σ_A from configuration c on. We denote by $\pi = c_0, c_1, \ldots$ a path in $Conf^\omega$ and with $\pi[i]$ the i-th configuration of π.

$$out(c, \sigma_A) = \Big\{ \pi = c_0, c_1, \ldots \mid c_0 = c \wedge \\ \forall i \in \mathbb{N}, \exists (act^i_{a_1}, \ldots, act^i_{a_n}), \forall a \in Agt. \\ \big(act^i_a \in av(c_i, a) \wedge \big(a \in A \rightarrow act^i_a \in \sigma_A\big) \wedge q_i \overset{(act^i_{a_1}, \ldots, act^i_{a_n})}{\rightarrow} q_{i+1}\big) \Big\}$$

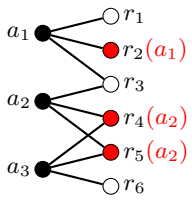
Formulae of $\mathcal{L}_{\text{GDP}}$ are evaluated over a game configuration graph $\mathfrak{G}$. The satisfaction relation $\mathfrak{G}, c \models \varphi$ is defined inductively on the structure of formulae, for all $c \in \mathfrak{G}$, as follows, where $\mathsf{res}(c, a) = |c^{-1}(a)|$, i.e. the number of resources owned by a in configuration c.

- $\mathfrak{G}, c \models g_{a_i}$ iff $\mathsf{res}(c, a_i) \geq d(a_i)$;
- $\wedge$, $\vee$ 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, for every path $\pi \in out(c, \sigma_A)$ we have that $\mathfrak{G}, \pi[1] \models \varphi$;
- $\mathfrak{G}, c \models \langle\langle A \rangle\rangle \mathsf{G}\, \varphi$ iff there is a joint strategy σ_A such that, for every path $\pi \in out(c, \sigma_A)$ and for every $i \in \mathbb{N}$ we have that $\mathfrak{G}, \pi[i] \models \varphi$;
- $\mathfrak{G}, c \models \langle\langle A \rangle\rangle \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there is a joint strategy σ_A such that, for every path $\pi \in out(c, \sigma_A)$ we have that $\exists i \geq 0. \mathfrak{G}, \pi[i] \models \varphi_2$ and $\forall 0 \leq j < i. \mathfrak{G}, \pi[j] \models \varphi_1$.

For every game configuration graph $\mathfrak{G}$ and a formula $\varphi \in \mathcal{L}_{\text{GDP}}$ we define the **extension of φ in $\mathfrak{G}$** to be the set of all configurations in $\mathfrak{G}$ that satisfy φ:

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

Example 5. The figure shows a graphical representation of c as in Example 2 where red resources are held by the agents in brackets. It is easy to see that the following formulae hold in $\mathfrak{G}, c$:
$\langle\langle a_1 \rangle\rangle \mathsf{G}\, \langle\langle a_1 \rangle\rangle \mathsf{F}\, g_{a_1}$, meaning that there is a strategy for agent a_1 to reach its goal infinitely often (it can always get resources r_1 and r_2), and $\langle\langle a_2, a_3 \rangle\rangle \mathsf{F}\, g_{a_2}$ (there is a strategy for a_2, a_3 to eventually reach the goal of a_2). But, there is no strategy for a_3 alone to eventually reach its goal, i.e. $\mathfrak{G}, c \not\models \langle\langle a_3 \rangle\rangle \mathsf{F}\, g_{a_3}$ as there exists a strategy for a_2 which prevents a_3 to acquire r_4 or r_5. Such a strategy by a_2 simply amounts to mimic a_3 requests (notice that the semantics of $\mathcal{L}_{\text{GDP}}$ only require such a counter-strategy to exists, even if in practice that would mean a_2 to guess a_3 requests).



The **global model checking problem** for $\mathcal{L}_{\text{GDP}}$ is a computational problem which amounts to compute $[\![\varphi]\!]_{\mathfrak{G}}$ given $\mathcal{G}$ and $\varphi \in \mathcal{L}_{\text{GDP}}$.

Lemma 1. *The global model checking problem for $\mathcal{L}_{\text{GDP}}$ has worst-case time complexity exponential in the number of resources.*

Proof. Follows immediately from Proposition 1 and the linear time complexity of the global model checking algorithm for ATL.

Since the number of resources in a GDP game scenarios is generally large, this exponential time bound is bad, and it is clear that the standard model checking algorithm for ATL [4], [10, Ch. 9] would be generally intractable if applied to explicitly generated configuration graphs of GDP games. That is why we develop here an abstract symbolic representation that will eventually keep the worst-case complexity of the model checking problem *polynomial* in the number of resources.

4 Symbolic Specification of $\mathcal{L}_{\mathrm{GDP}}$ Formulae

Our solution is based on an abstraction which represents sets of configurations by means of symbolic expressions. Essentially, we group together configurations which cannot be distinguished by $\mathcal{L}_{\mathrm{GDP}}$-formulae and that behave "similarly" with respect to the $\mathfrak{G}$ transition function. In this way we solve the global model checking problem of $\mathcal{L}_{\mathrm{GDP}}$-formulae without explicitly computing the configuration graph $\mathfrak{G}$, but rather by computing the corresponding symbolic expressions.

In other words, configurations contains more information than needed to answer $\mathcal{L}_{\mathrm{GDP}}$-formulas: indeed, configurations describe which resources each agents holds, while $\mathcal{L}_{\mathrm{GDP}}$-formulas cannot distinguish configurations where agents hold the same number of resources. Therefore, intuition suggests that abstracting from specific resources each agent has and keep only the number of those held by each agent may suffice. Unfortunately this not the case, as resources are in general accessible to subset of agents only (as specified by the *Acc*) thus, given a subset of agents, how many resources *among those accessible to them* are still available is a necessary information to reason on the capability to achieve their goals. To see this, let us consider a situation of $\mathcal{G}$ in Example 1 where a_3 holds one resource only: knowing which one is important to determine the capability of a_3 to reach g_{a_3} in the next state, namely to assess wether $\mathfrak{G}c \models \langle\langle a_3 \rangle\rangle \mathsf{X}\, g_{a_3}$. Indeed, if the resource owned by a_3 is r_4 (or r_5), then the above formula is true (as a_3 can always request and obtain r_6) whilst if it is r_6 is false, as there is no guarantee it can obtain one among r_4 or r_5.

Our symbolic representation is based on the intuition that only resources that are accessed by the same subset of agents can be safely regarded as undistinguishable. Thus, we start by formally describing such sets of resources. With mild notational abuse, we define $Acc^{-1} : Res \to 2^{Agt}$ such that $Acc^{-1}(r) = \{a \mid Acc(a, r)\}$ and $Acc : 2^{Agt} \to 2^{Res}$ such that $Acc(A) = \{r \in Res \mid \forall a \in Agt.a \in A \leftrightarrow Acc(a, r)\}$, namely the set of resources shared by, and only by, the agents in A.

The function Acc^{-1} defined above induces an *equivalence relation* $\sim \subseteq Res \times Res$ among resources as follows:

$$r_i \sim r_j \leftrightarrow (Acc^{-1}(r_i) = Acc^{-1}(r_j))$$

We denote by $\mathcal{R} = \{R_1, \ldots, R_u\}$ the quotient set of *Res* by $\sim$, namely the set of equivalence classes in *Res* induced by $\sim$. Notice that each $R_i \in \mathcal{R}$ uniquely

identifies the set of agents which have access to them. Again with mild notational abuse we denote such a set by $Acc^{-1}(R_i)$.

Our symbolic representation defines for each agent a_i and each equivalence class of resources R_j, the range on the number of resources in R_j that a_i holds.

Definition 9 (Interval Expressions). *Let $\mathcal{G}$ be a game, based of an intuitive understanding of the notion of 'intervals',* interval expressions α *are defined as follows:*

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

Intuitively we call each $(a, R)[l_R^a, l_R^a]$ an **interval**, and it constrains the number of resources held by a and belonging to R to be between l_R^a and L_R^a.

Now we define the *formal semantics* of interval expressions. In every GDP game configuration graph $\mathfrak{G}$ each interval expression α defines a set of configurations $\|\alpha\|_{\mathfrak{G}}$, the **extension** of α, as follows:

- $\|(a, R)[l_R^a, L_R^a]\|_{\mathfrak{G}} = \{c \in Conf : l_R^a \leq |c^{-1}(a) \cap R| \leq L_R^a\}$;
- $\|\bigwedge_{a \in Agt} \bigwedge_{R \in \mathcal{R}} (a, R)[l_R^a, l_R^a]\|_{\mathfrak{G}} = \bigcap_{a \in Agt} \bigcap_{R \in \mathcal{R}} \|(a, R)[l_R^a, L_R^a]\|_{\mathfrak{G}}$;
- $\|\alpha_1 \vee \alpha_2\|_{\mathfrak{G}} = \|\alpha_1\|_{\mathfrak{G}} \cup \|\alpha_2\|_{\mathfrak{G}}$.

Example 6. With reference to $\mathcal{G}$ as in Example 1, we notice that there are four equivalence classes in $\mathcal{R}$: $R_1 = \{r_1, r_2\}$, with $Acc^{-1}(R_1) = \{a_1\}$; $R_2 = \{r_3\}$, with $Acc^{-1}(R_2) = \{a_1, a_2\}$; $R_3 = \{r_4, r_5\}$, with $Acc^{-1}(R_3) = \{a_2, a_3\}$ and $R_4 = \{r_6\}$, with $Acc^{-1}(R_4) = \{a_3\}$. The expression α' :

$$\begin{aligned}&(a_1, R_1)[0, 1] \wedge (a_1, R_2)[0, 1] \wedge \\ &(a_2, R_2)[0, 1] \wedge (a_2, R_3)[1, 2] \wedge \\ &(a_3, R_3)[0, 0] \wedge (a_3, R_4)[0, 1]\end{aligned}$$

is such that the configuration c from Example 2 belongs to $\|\alpha'\|_{\mathfrak{G}}$. For the sake of readability we omit intervals for pairs (a', R') such that $R \cap Acc^{-1}(a') = \emptyset$. Let also c' be as c but with agent a_1 not holding r_2. Then, $c' \in \|\alpha'\|_{\mathfrak{G}}$ as well.

Interval expressions, being defined on intervals, are very modular, and they can represent sets of configurations of different sizes. They can be classified from "coarsest" to "finest" depending on the number of configurations in their extensions. The larger the number, the coarser the abstraction. Given a game $\mathcal{G}$ there is a coarsest formula $\alpha_\top$ representing the whole set *Conf* of configurations which is such that for all $a \in Agt$ and $R \in \mathcal{R}$, $(a, R)[0, min(d(a), |R|)]$. Analogously, it is always possible to define a finest expression $\alpha_\bot$ the extension of which is empty (it is enough to define an interval $(a, R)[L, L]$ where $L > |Res|$). The finer expressions are more important for our purposes, as they provide the smallest abstraction level we can manipulate. Those expressions, of the form: $\bigwedge_{a \in Agt} \bigwedge_{R \in \mathcal{R}} (a, R)[\ell_R^a, \ell_R^a]$ will be called **β-expressions**. Clearly, every interval expression can be transformed into a semantically equivalent (i.e., with the same extension) *expanded normal form*, that is, a disjunction of β-expressions.

Definition 10 (Expanded Normal Form). *An interval constraint expression α is in **expanded normal form** if each interval is of the form $(a, R)[\ell^a_R, \ell^a_R]$.*

We recursively define function $\text{EXPD}(\alpha)$ which translates α in its extended normal form as follows:

$$\begin{aligned}\text{EXPD}(\textstyle\bigwedge_{a\in Agt}\bigwedge_{R\in\mathcal{R}}(a,R)[l^a_R, L^a_R]) ::= & \textstyle\bigvee_{l^{a_1}_{R_1}\le\ell^{a_1}_{R_1}\le L^{a_1}_{R_1}}\cdots\bigvee_{l^{a_1}_{R_u}\le\ell^{a_1}_{R_u}\le L^{a_1}_{R_u}}\\ & \vdots\\ & \textstyle\bigvee_{l^{a_n}_{R_1}\le\ell^{a_n}_{R_1}\le L^{a_n}_{R_1}}\cdots\bigvee_{l^{a_n}_{R_u}\le\ell^{a_n}_{R_u}\le L^{a_n}_{R_u}}\\ & (\textstyle\bigwedge_{1\le i\le n}\bigwedge_{1\le j\le u}(a_i,R_j)[l^{a_i}_{R_j}, l^{a_i}_{R_j}])\end{aligned}$$

$$\text{EXPD}(\alpha_1\vee\alpha_2) ::= \text{EXPD}(\alpha_1)\vee\text{EXPD}(\alpha_2)$$

Lemma 2. *Let α be an interval expression. Its expanded normal form $\text{EXPD}(\alpha)$ has, in the worst-case, size which is double exponential in the number of agents and polynomial in the number of resources.*

Proof. Consider $\alpha_\top$ in a game $\mathcal{G}$. Then $\text{EXPD}(\alpha_\top)$ returns all possible β-expressions in $\mathcal{G}$. If we consider $|Res|$ as the complexity parameter, then the worst case is when $d(a_i) = |Res|$ for each $i \in \{1, \ldots, n\}$, which results in a size of $\text{EXPD}(\alpha_\top)$ which is polynomial in the number of resources. If we consider $|Agt|$ as the complexity parameter, then the worst case is given by Acc such that we have all equivalence classes for resources, i.e., $2^{|Agt|}$. Given that each interval in α is $(a, R)[0, |Res|]$, we have size of α double exponential in the number of agents.

5 Symbolic Verification of $\mathcal{L}_{\text{GDP}}$ Formulae

The algorithm for the symbolic verification of $\mathcal{L}_{\text{GDP}}$-formulae has the same structure as that for ATL formulae [4][10, Ch. 9]: for the strategic next-time operator it computes the controllable by a given coalition A pre-image of the extension of a given formula, and then exploits the fixpoint characterisations of the temporal operators G and U. Now, instead of manipulating states of the explicit model $\mathfrak{G}$, i.e. the configuration graph, our algorithm works symbolically, with interval expressions. The basic step of the algorithm is to compute, given a coalition A and an interval expression α', the *controllable by A symbolic pre-image* $\alpha = \text{PRE}(\mathcal{G}, A, \alpha')$ which, intuitively, is the set of all interval expressions that only 'contain' configurations from which coalition A has a joint action that forces the outcome to be in the set of configurations defined by the extension of α'.

Definition 11 (Controllability). *Let $\mathcal{G}$ be a GDP game, $A \subseteq Agt$, and α, α' interval expressions. We say that α **is in the controllable by A pre-image of** α' iff there exists a strategy σ_A such that, for all $c \in \| \alpha \|_{\mathfrak{G}}$ and for all $\pi \in out(c, \sigma_A)$ we have that $\pi[1] \in \|\alpha'\|_{\mathfrak{G}}$.*

In what follows, we show how to compute the controllable pre-image of a interval expression. Given a GDP game $\mathcal{G}$, we do the following:

1. As pre-processing we compute the set AP of all action profiles in $\mathcal{G}$.
2. Then, we consider the expansion $\text{EXPD}(\alpha') = \beta'_1 \vee \ldots \vee \beta'_s$, where $\beta'_1, \ldots, \beta'_s$ are β-expressions. For each $\beta'_i \in \text{EXPD}(\alpha')$ and each $ap_j \in AP$ we compute set $\{\beta_{i,j,1}, \ldots \beta_{i,j,t}\}$ of symbolic β-expressions such that, for each $c \in \text{EXPD}(\beta_{i,j,k})$, performing ap leads to a configuration $c' \in \text{EXPD}(\beta'_i)$.
3. Lastly, for each $\beta_{i,j,k}$ we check if it belongs in the controllable by A pre-image of α', by verifying if there exists a joint action for agents in A such for every possible joint actions for the other agents, the successor is in α'. This requires to check all possible action profiles. If such a joint action exists, $\beta_{i,j,k}$ will be added in disjunction to a formula $\alpha_{\beta'_i}$ which represents (part of) the controllable pre-image of α' that has been computed, roughly speaking, by considering predecessors of β'_i. The whole controllable by A pre-image α of α' is simply the union of such controllable predecessors, namely $\alpha = \bigcup_{1 \leq i \leq s} \alpha_{\beta'_i}$.

We detail steps (2) and (3).

Step (2). The difficulty lies in the fact that the transition function $\rho(\mathfrak{G})$ is not *injective*, thus when computing the pre-images of an interval expression β'_i for a specific ap_j, the result consists, in general, of more than one β-interval expression. In order to see why, let us consider the following example.

Example 7. Let α' be as in Example 6 and let β'_i:

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

be one of its expanded β-expressions. Notice that configuration c' at the end of Example 6 belongs to β'_i.

Now, let $\langle \mathsf{rel}^{a_1}_{all}, \mathsf{req}^{a_2}_{r_4}, \mathsf{rel}^{a_3}_{r_6} \rangle$ be an action profile ap_j. Which interval expressions belong to the pre-image of β'_i given ap_j? In other words, from which interval expressions is β'_i reachable when performing ap_j? Issues arise with the *release-all* action $\mathsf{rel}^{a_1}_{all}$, as we do not know which resources a_1 was holding in the previous configuration, hence to which equivalence classes those resources belonged to. Indeed, since $d(a_1) = 2$, a_1 was holding 2 resources that could have been any of the following: $\{\{r_1, r_2\}, \{r_1, r_3\}, \{r_2, r_3\}\}$. This gives rise to two different interval expressions in the pre-image of β' given ap_j, that are $\beta_{i,j,1}$ in case a_1 was holding $\{r_1, r_2\}$:

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

and $\beta_{i,j,2}$ if a_1 was holding either $\{r_1, r_3\}$ or $\{r_2, r_3\}$:

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

Technically, during Step (2) for each $ap_j \in AP$, we compute the set $\{\overline{ap}_{j,1}, ..., \overline{ap}_{j,q}\}$ of *expanded* action profiles for ap_j, where each $\overline{ap}_{j,k}$ is a set of actions where: *(i)* resources are replaced by their equivalence classes and *(ii)* the *release-all* actions by each agent a are replaced by a set of (single-resource) *release* actions one for each resource to be released by a in an equivalence relation in all possible ways. It is easy to see that the number of possible *expanded* action profiles for ap_j is, in the worst case, double exponential in the number of agents. Indeed, if agent a performs a *release-all* action, such that $d(a) = |Res|$ and a has access to all possible equivalence classes of resources, namely $|2^{Agt}|$, then we get a predecessor for each way of assigning $|Res|$ resources to 2^{Agt} equivalence classes, thus $|Res^{2^{Agt}}|$.

We now show how we actually compute the predecessor. Let

$$\beta' = \bigwedge_{a \in Agt} \bigwedge_{R \in \mathcal{R}} (a, R)[\ell^a_R, \ell^a_R]$$

(primed) be the successor state under consideration and let $(ap_j, \overline{ap}_{j,k})$ as before. We build β (unprimed) such that $\beta_{i,j,k} \overset{(ap_j, \overline{ap}_{j,k})}{\rightarrow} \beta'_i$ by first considering the release actions in $\overline{ap}_{j,k}$ and then the request actions in ap_j:

- for each release action $\mathsf{rel}^{\bar{a}}_{\bar{R}} \in \overline{ap}_{j,k}$ we increase by one the number $\ell^{\bar{a}}_{\bar{R}}$;
- for each request action $\mathsf{req}^{\bar{a}}_{\bar{r}} \in ap$ such that $\neg\exists a'$ such that $\mathsf{req}^{\bar{a}'}_{\bar{r}} \in ap$ we decrease by one the number $\ell^a_{\bar{R}}$ where $\bar{R}$ is the equivalence class of $\bar{r}$.

Step (3). Given a $\beta_{i,j,k}$ obtained in the previous step, for each joint action for A we check if that is the required one-step strategy σ_n by simply verifying if all action profiles extending σ_n lead to α'. Let ap_v be one of such action profiles. We compute $\beta'_{i,j,k,v}$ such that $\beta_{i,j,k} \overset{ap_v}{\rightarrow} \beta'_{i,j,k,v}$. The issue is that $\beta_{i,j,k} \overset{ap_v}{\rightarrow} \beta'_{i,j,k,v}$ may not be a step, for two reasons: *(i)* $\beta_{i,j,k}$ may be inconsistent, i.e., $\|\beta_{i,j,k}\|_{\mathfrak{G}} = \emptyset$ or *(ii)* ap_v may not be executable in $\beta_{i,j,k}$, meaning that there is no $c \in \|\beta_{i,j,k}\|_{\mathfrak{G}}$ such that $c \overset{ap_v}{\rightarrow} c'$ is a step.

We first perform check *(i)* and then *(ii)* separately but with the same technique: by a reduction to the maximal matching problem in a bipartite graph.

Given bipartite graph $G = (V = (X, Y), E)$ where $E \subseteq X \times Y$, the maximum matching problem amounts to find a *maximal* set $M \subseteq E$ such that $\forall (x, y), (x', y') \in M. u \neq u' \wedge v \neq v'$, namely edges in M share no vertices. The Hopcroft-Karp algorithm [13] solves it in worst-time complexity linear in the size of G.

We now show how to build the bipartite problem which solution guarantees that $\beta_{i,j,k}$ is consistent. From $\beta_{i,j,k}$ and Acc we build $bpg(\beta, Acc) = ((X, Y), E)$ as follows:

- for each $a \in Agt$ and $R \in \mathcal{R}$ we have $\{x(a, R, 1), \ldots, x(a, R, \ell^a_R)\} \in X$ iff $\ell_a \neq 0$;
- $Y = Res$;
- for each $a \in Agt$ and $R \in \mathcal{R}$ and $1 \leq t \leq \ell^a_R$, $(x(a, R, t), r) \in E$ iff $(a, r) \in Acc$ and $r \in R$.

In the worst case, the size of G is double exponential in the number of agents, coming from the number of x-nodes. It is easy to prove that solution M of the maximum matching problem is such that each x-node is covered, i.e., $\forall x \in X, \exists y \in Y.(x, y) \in M$ if and only if $\|\beta_{i,j,k}\|_{\mathfrak{G}} \neq \emptyset$. The only if entailment follows from the fact that M is the largest possible.

If $\|\beta_{i,j,k}\|_{\mathfrak{G}} \neq \emptyset$ then we check if ap_u can be executed in $\beta_{i,j,k}$.

Fact 1 *Notice that given $\beta_{i,j,k}$ consistent, and ap_u, then there is a unique possible* expansion *$(ap_u, \overline{ap}_u)$, as $\beta_{i,j,k}$ provides information on the equivalence classes of resources held by agents performing release-all actions. This entails there is only a successor for each action profile, which guarantees that the transition relation between interval expressions is actually a function.*

We build an instance of the maximum bipartite problem $bpg(\beta_{i,j,k}, Acc, (ap_u, \overline{ap}_u))$ starting from $bpg(\beta_{i,j,k}, Acc) = ((X, Y), E)$ as before and modifying it so as to account for $(ap_u, \overline{ap}_u)$:

1. for each $\mathsf{rel}^a_R \in \overline{ap}_u$ we:
   (a) remove one node among $\{x(a, R, 1), \ldots x(a, R, \ell^a_R)\}$ and its corresponding edges. If no such a node exists, then $\beta_{i,j,k}$ is not compatible with $(ap_u, \overline{ap}_u)$ and we discard them both;
   (b) add a node $z(\mathsf{rel}^a_R)$ to X and edges $(z(\mathsf{rel}^a_R), r)$ for all $r \in R$.
2. for each request action $\mathsf{req}^a_r \in ap$:
   (a) add a node $z(\mathsf{req}_r)$ and edge $(z(\mathsf{req}_r), r)$.

The maximum matching problem for the above has a solution covering all the nodes in X iff $(ap_u, \overline{ap}_u)$ is executable in $\beta_{i,j,k}$. The only if entailment is again guaranteed by M being the largest. The (again, unique) successor state $\beta'_{i,j,k,u}$ can be easily computed by modifying the intervals in the intuitive way.

We now present our main result, the soundness and completeness of our symbolic technique for global model checking of $\mathcal{L}_{\mathrm{GDP}}$ formulae. The proof is constructive and provides a model checking algorithm.

Theorem 1. *For every GDP game $\mathcal{G}$ and a formula $\varphi \in \mathcal{L}_{\mathrm{GDP}}$ there exists an interval constraints expression $\alpha(\mathcal{G}, \varphi)$ such that:*

$$[\![\varphi]\!]_{\mathfrak{G}} = \|\alpha(\mathcal{G}, \varphi)\|_{\mathfrak{G}}$$

The proof is in the Appendix.

Theorem 2. *For each $\varphi \in \mathcal{L}_{\mathrm{GDP}}$, computing $\|\alpha(\mathcal{G}, \varphi)\|_{\mathfrak{G}}$ can be done in time double exponential in the number of agents, and polynomial in the number of resources.*

Proof. To be moved here from the appendix.

6 Conclusions and Outlook to Future Work

In this paper we introduced the Generalised Dining Philosophers games, which are a substantial generalisation of the original dining philosophers problem and model distributed multi-agent dynamic resource allocation problems. We have developed a symbolic algorithm for the verification of properties of GDP games specifiable in a version of ATL, built over atomic propositions stating that an agent's goal is achieved. We have showed that this symbolic algorithm works in time which is polynomial in the number of resources, while the standard ATL model checking algorithm, applied on the explicit configuration graph of the game generally works in time exponential in both the number of agents and the number of resources. We therefore claim that the symbolic algorithm is significantly more efficient than the explicit one in cases where the number of resources is much larger than the number of agents.

From the theoretical perspective, GDP games are amenable to various natural modifications of the operational semantics, e.g.: allowing agents to request multiple resources at a time, or assuming basic individual rationality according to which a rational agent will always attempt to acquire more resources rather than idle, until it reaches its goal. These considerations are left to future work, in which we also intend to apply our new techniques to more realistic scenarios in the domain of 'classical' Computer Science (e.g. operating systems). Also, the two opposite paths of narrowing and broadening of our approach are both worth exploring. The former looks for classes of formulas, or classes of models (e.g., those with a specific accessibility graph) that allow for more compact symbolic representations, while the latter investigates to which extent our ideas can be successfully applied to more general settings.

From the practical perspective, a future implementation of the symbolic technique described here will allow to compare the effectiveness of our approach against explicit or BDD-based model checkers for MAS.

Acknowledgements

The work of Valentin Goranko and Riccardo De Masellis was supported by a research grant 2015-04388 of the Swedish Research Council, which also partly funded a working visit of Nils Timm to Stockholm. Valentin Goranko was also partly supported by a visiting professorship grant by the University of Pretoria.

References

1. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Resource-bounded alternating-time temporal logic. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1. pp. 481–488. AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2010)
2. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F.: Symbolic model checking for one-resource RB+-ATL. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 1069–1075. IJCAI'15, AAAI Press (2015)

3. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F., Mostarda, L.: Symbolic model-checking for resource-bounded ATL. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 1809–1810. AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2015)
4. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
5. Bhargava, D., Vyas, S.: Agent based solution for dining philosophers problem. In: 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS). pp. 563–567 (Dec 2017)
6. Chandy, K.M., Misra, J.: The drinking philosophers problem. ACM Trans. Program. Lang. Syst. **6**(4), 632–646 (Oct 1984)
7. Chevaleyre, Y., Dunne, P., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodrguez-Aguilar, J., Sousa, P.: Issues in multiagent resource allocation. Informatica **30**, 3—-31 (2006)
8. Choppella, V., Kasturi, V., Sanjeev, A.: Generalised dining philosophers as feedback control. CoRR **abs/1805.02010** (2018)
9. Datta, A.K., Gradinariu, M., Raynal, M.: Stabilizing mobile philosophers. Information Processing Letters **95**(1), 299 – 306 (2005)
10. Demri, S., Goranko, V., Lange, M.: Temporal Logics in Computer Science. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2016), `http://www.cambridge.org/core_title/gb/434611`
11. Dijkstra, E.W.: Hierarchical ordering of sequential processes. Acta Informatica **1**(2), 115–138 (Jun 1971)
12. Herescu, O.M., Palamidessi, C.: On the generalized dining philosophers problem. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing. pp. 81–89. PODC '01, ACM, New York, NY, USA (2001)
13. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973)
14. Papatriantafilou, M.: On distributed resource handling: Dining, drinking and mobile philosophers. In: In Proceedings of the First International Conference on Principles of Distributed Systems (OPODIS. pp. 293–308 (1997)
15. Sidhu, D.P., Pollack, R.H.: A robust distributed solution to the generalized dining philosophers problem. In: 1984 IEEE First International Conference on Data Engineering. pp. 483–489 (April 1984)

A Technical Proofs

Proof of Theorem 1: By induction on φ.

- $\varphi = g_a$. Then $[\![\varphi]\!]_{\mathfrak{G}} = \{c : \mathsf{res}(c, a) = d(a)\}$. The required formula is $\alpha = \alpha_1 \vee \ldots \vee \alpha_s$ where each α_i is the conjunction of intervals for agents $a' \neq a$ and the conjunction of intervals for a. The conjunction of intervals for $a' \neq a$ is the same for every α_i, and is, for each $a' \neq a$ and for each $R \in \mathcal{R}$, $(a', R)[0, d(a')]$. The conjunction of intervals for a is different for each α_i, has the form $(a, R)[\ell^a_R, \ell^a_R]$ and is a solution of the constraint $\sum_{R\in\mathcal{R}} \ell^a_R = d(a)$.
- $\varphi = \langle\langle A\rangle\rangle \mathsf{X}\, \psi$. Then $[\![\varphi]\!]_{\mathfrak{G}}$ is the set of configurations from which there exists a collective strategy σ_A for A such that for all $\pi \in out(c, \sigma_A)$, $\mathfrak{G}, \pi[1] \models \psi$. We show that $\textsc{pre}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))$ satisfies the claim. We prove separately $[\![\varphi]\!]_{\mathfrak{G}} \supseteq \|\textsc{pre}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))\|_{\mathfrak{G}}$ (soundness) and $[\![\varphi]\!]_{\mathfrak{G}} \subseteq \|\textsc{pre}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))\|_{\mathfrak{G}}$ (completeness).

Soundness. We have to prove that for every $c \in \|\text{PRE}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))\|_{\mathfrak{G}}$ there exists one step strategy to reach a state where ψ holds. By the inductive hypothesis, all (and only) configurations in $\alpha(\mathfrak{G}, \psi)$ satisfy ψ, thus it is enough to show that a one step strategy reaches $\alpha(\mathfrak{G}, \psi)$. Let us then consider a generic $c \in \|\text{PRE}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))\|_{\mathfrak{G}}$. If $c \in \|\text{PRE}(\mathcal{G}, A, \alpha(\mathcal{G}, \psi))\|_{\mathfrak{G}}$ then, by the pre-image algorithm, there exists β, β' and $(ap, \overline{ap})$ such that:

- $c \in \|\beta\|_{\mathfrak{G}}$;
- β is in the predecessor of β', thus $\beta \overset{(ap,\overline{ap})}{\rightarrow} \beta'$ from soundness of step (2) in the pre-image computation;
- there exists a one-step strategy σ_A from β which leads to $\alpha(\mathcal{G}, \psi)$.

We have to prove that such a strategy satisfies Definition 11.
Let $(ap, \overline{ap})$ be a generic action profile consistent with strategy σ_A such that $\beta \overset{(ap,\overline{ap})}{\rightarrow} \beta''$, with clearly $\beta'' \in \text{EXPD}(\alpha(\mathcal{G}, \psi))$. We show that for all $c \in \|\beta\|_{\mathfrak{G}}$ there exists $ap' \in AP$ and there exists $c' \in Conf$ such that $c \overset{ap'}{\rightarrow} c'$, $c' \in \|\beta''\|_{\mathfrak{G}}$ and the *expansion* of ap' is $\overline{ap}$.
Constructively, we build ap' agent-wise from $\overline{ap}$. For each $a \in Agt$, its action in $\overline{ap}$ can be one of the following: (1) idle^a; (2) (single release) rel^a_R; (3) (multiple release) $\{\mathsf{rel}^a_{R_1}, \ldots, \mathsf{rel}^a_{R_t}\}$; or (4) req^a_R. If (1) then the $ap'(a) = \mathsf{idle}^a$. If (2) then $ap'(a) = \mathsf{rel}^a_r$ for a random $r \in R$ such that $c(r) = a$. Notice that this is always possible, given that $\beta \overset{(ap,\overline{ap})}{\rightarrow} \beta''$, meaning that a does have the number of resources necessary to perform a release. If (3), then $ap'(a) = \mathsf{rel}^a_{all}$. This is always possible again from the same observations in (2). If (4), then it is not enough to pick a random available $r \in R$ and set $ap'(a) = \mathsf{req}^a_r$, as, depending on the requests actions performed by the other agents, a can own or not r in the next configuration. Thus, first the requested r must be such that $c(r) = null$ (there exists one otherwise $\beta \overset{(ap,\overline{ap})}{\rightarrow} \beta''$ would not be a step) but also the requests actions of all agents have to be considered together when building ap'. It is however enough to look at the interval $(a, R)[\ell'^a_R, \ell'^a_R]$ in β'' in order to determine whether the request of a has been successful or not. If $\ell'^a_R = \ell^a_R$, then it is the former, and there is another agent a' performing a request for the same resource (this is guaranteed by how the intervals of β'' are computing from β by performing $(ap, \overline{ap})$). Otherwise, if $\ell'^a_R = \ell^a_R + 1$, then the request by a of r has been successful and, when building ap' for the other agents, we have to be sure that no other agent requests r (again, this is always guaranteed by how the intervals of β'' are computing from β by performing $(ap, \overline{ap})$). Finally, we have to show that $c' \in \|\beta''\|_{\mathfrak{G}}$. This is immediate as the expansion of ap' is $\overline{ap}$ and from the way intervals are updated when computing $\beta \overset{(ap,\overline{ap})}{\rightarrow} \beta''$.

Completeness. We have to prove that if $c \in Conf$ and there exists a strategy σ_A which in one step reaches a configuration satisfying ψ, then $c \in \|\text{PRE}(\mathcal{G}, A, \alpha(\mathfrak{G}, \psi))\|_{\mathfrak{G}}$. If c satisfies the above, then $\exists ap, c'.c \overset{ap}{\rightarrow} c'$ with $ap \in \sigma_A(c)$ and $c' \in [\![\psi]\!]_{\mathfrak{G}}$. By inductive hypothesis, $c' \in \|\alpha(\mathcal{G}, \psi)\|_{\mathfrak{G}}$, meaning that there exists $\beta' \in \alpha(\mathcal{G}, \psi)$ and $c' \in \|\beta'\|_{\mathfrak{G}}$. The pre-image algorithm tries

all action profiles in AP that could lead to β', thus also ap. We have to show that there exists $\overline{ap}_k$ such that $\beta \overset{(ap,\overline{ap}_k)}{\rightarrow} \beta'$ and $c \in \|\beta\|_{\mathfrak{G}}$. Constructively, the required $\overline{ap}_k$ is built by looking at configuration c, which tells us to which classes the resources of the agents performing the release all-operations belongs. Since it exists, the algorithm finds it as it explores all possible $\overline{ap}_k$ for any ap. Also $c \in \|\beta\|_{\mathfrak{G}}$ by inspecting how we modify the intervals when computing the predecessors. It remains to prove that β is controllable. This is straightforward as the required controllable joint action for A in Definition 11 is easily obtained from ap, which is a one-step strategy by hypothesis.

- $\varphi = \neg\psi$. Recursively, from $\alpha(\mathcal{G}, \psi) = \beta_1 \vee \ldots \beta_s$ we compute the negation $\overline{\beta_i}$ of each β_i complementing its intervals. Such operation produces at most two intervals, thus each $|\overline{\beta_i}|$ is at most $2 \cdot |\beta_i|$. Then we produce the intersection of those β_i by simply "projecting" on intervals common to each β_i.
- $\varphi = \langle\langle A \rangle\rangle \mathsf{G}\, \psi$. Recursively, we start from $\alpha(\mathcal{G}, \psi)$ and compute the conjunction with its pre-image until a fixpoint is reached. Sound and completeness follows from the fixpoint characterisation of the G operator.
- $\varphi = \langle\langle A \rangle\rangle \psi_1 \,\mathsf{U}\, \psi_2$. Similarly to the previous case, but each iteration computes the disjunction of $\alpha(\mathcal{G}, \psi_2)$ with the conjunction of the pre-image of the expression obtained at the previous step with $\alpha(\mathcal{G}, \psi_1)$. Soundness and completeness follow from the fixpoint characterisation of the operator U.

Proof of Theorem 2: Let us start from the complexity of the controllable pre-image subroutine. As pointed out in the description of the algorithm, Step (2) generates all possible distinct β-expressions, which are, as stated by Lemma 2, double exponential in the number of agents and polynomial in the number of resources. Given a formula $\varphi \in \mathcal{L}_{\mathrm{GDP}}$, the number of times the controllable pre-image subroutine is called is linear in the number of possible distinct β-expressions for each fixpoint computation and the number of fixpoint computations is linear in the size of the formula.