Fair decomposition of group obligations

Natasha Alechina
University of Nottingham,
Nottingham, UK,
nza@cs.nott.ac.uk

Wiebe van der Hoek
University of Liverpool,
Liverpool, UK
Wiebe.Van-Der-Hoek@liverpool.ac.uk

Brian Logan
University of Nottingham,
Nottingham, UK,
bsl@cs.nott.ac.uk

Abstract

We consider the problem of decomposing a group norm into a set of individual obligations for the agents comprising the group, such that if the individual obligations are fulfilled, the group obligation is fulfilled. Such an assignment of tasks to agents is often subject to additional social or organisational norms that specify permissible ways in which tasks can be assigned. An important role of social norms is that they can be used to impose 'fairness constraints', which seek to distribute individual responsibility for discharging the group norm in a 'fair' or 'equitable' way. We propose a simple language for this kind of fairness constraints and analyse the problem of computing a fair decomposition of a group obligation, both for non-repeating and for repeating group obligations.

Keywords: Multi-Agent Systems, group obligations, group norms, effectiveness, fairness, minimality of norms.

1 Introduction

Norms have been widely proposed as a means of achieving coordination and guaranteeing desirable system-level properties in multi-agent systems (MAS). Much of the literature on normative MAS has focussed on obligations and prohibitions associated with roles in an organisational structure or directed to individual agents (see, for example, [17]). However, many norms apply to *groups* of agents rather than to an agent enacting a role, or a particular agent in a MAS. For example, the members of the

programme committee for a workshop may have a collective obligation to review the papers submitted to the workshop, or the occupants of a shared apartment may have an obligation to keep the apartment clean (e.g., as part of the rental agreement). Such *group norms* specify a sequence of actions that should be performed by members of the group, leaving the details of how the norm is to be complied with to the members of the group themselves. In general, there will be many possible ways to comply with a group norm, i.e., several different assignments of agents to particular tasks. Each assignment gives rise to a set of individual obligations that specify what each agent should do in order to discharge the group obligation. (Note that this paper will focus on ought-to-do norms, rather norms for ought-to-be. The two are closely related, see e.g., [16]).

The assignment of agents to tasks specified by a group norm is often subject to additional social or organisational norms that specify permissible ways in which tasks can be assigned. An important type of social norms are 'fairness constraints', that seek to distribute individual responsibility for discharging the group norm in a 'fair' or 'equitable' way. For example, there may be a constraint that no single agent should be required to do all the work necessary to discharge the group norm, or that no agent should have to do a particular task more than once a week, etc. The social norms codifying what counts as 'fair' vary from organisation to organisation. For example, in some computer science departments, all members of academic staff may be assigned teaching duties, while in other departments, more senior academics are not obliged to teach. A key problem in normative MAS with group norms is determining whether a particular task allocation is both *effective* (i.e., it discharges the group norm) and *fair*, in the sense of respecting the social norms or fairness constraints in force within the organisation of which the group is a part.

This paper builds upon and extends [4], in which we made a first step towards defining the notion of a fair decomposition of a group obligation into individual obligations for agents in the group. In [4], a group obligation is considered to be a sequential or parallel composition of actions that have to be performed by the agents in the group, either once or repeated indefinitely (for example, the obligation to keep the household running involves repeated execution of the same sequence of cleaning, cooking etc. actions). In this paper, we generalise this by relaxing the requirement of a fixed sequence of actions to allow disjunctions over different orders of executing actions. We generalise results from [4] to this new setting, and prove new results (Theorem 3). We also give additional examples and expand related work. As in [4], we show how to specify agents' individual offers to contribute to a group norm, and analyse the problem of producing a set of individual obligations for the agents in the group, such that if those individual obligations are fulfilled, the group obligation is fulfilled (i.e., an implementation of the group norm). We propose a simple language for fairness constraints and analyse the problem of computing a fair implementation of a group obligation, for both non-repeating and repeating group obligations. We also address the notion of *minimality*: an implementation should not unnecessarily demand contributions from agents.

The structure of the paper is as follows. In Section 2 we introduce the formal preliminaries, including the formal language we use to talk about group obligations and the structures used to interpret the language. In Section 3 we introduce the basic setting of non-repeating group obligations and prove that the problem of whether an

implementation of a group norm exists is NP-complete. We also analyse the problem of the existence of minimal and fair implementations. In Section 4 we analyse similar problems for repeating group obligations. We place our work in the context of existing research in Section 5 and discuss future work in Section 6.

2 Formal setting

Several approaches to norms have been proposed in the literature, including state-based norms (where norms are defined in terms of states that should or should not occur), e.g., [25], and event or action-based norms (where norms are defined in terms of what agents should or should not do), e.g., [20, 14]. In this paper we take an action-based view of norms, in which norms are interpreted as specifying a sequence of actions (possibly containing gaps) that should occur, either once or repeatedly.[1]

We work in a propositional language of linear-time temporal logic. We assume that we have a set of propositional variables $Prop$ that, in addition to 'normal' propositional variables such as c for 'the room is clean', contain a special kind of variables of the form $\mathbf{done}(a, i)$, where a is a type of action from a set of actions Ac (which includes the no-op action `skip`) and i is the name of an agent in the set of agent names $Ag = \{1, \ldots n\}$. Intuitively, $\mathbf{done}(a, i)$ is true in a state if immediately before that state, agent i has performed action a.

The syntax of Linear Time Temporal Logic (LTL), see, e.g., [30], is defined as follows:

$$\phi, \psi := \ p \mid \neg\phi \mid \phi \wedge \psi \mid \ \bigcirc \phi \mid \phi \mathcal{U} \psi$$

where $p \in Prop$, $\bigcirc$ means next state, and $\mathcal{U}$ means until.

Definition 1. *A transition system for a set Ag of n agents and a set Ac of actions is a tuple $\langle S, R, V, s_I \rangle$, where*

- *S is a non-empty set of states;*
- *$R \subseteq Ac^n \times S \times S$. Intuitively, $(\langle a_1, \ldots, a_n \rangle, s, s') \in R$ means that there is a transition from s to s' which corresponds to actions $a_1, \ldots, a_n$ executed by agents $1, \ldots, n$ in parallel. (For $\vec{a} = \langle a_1, \ldots, a_n \rangle \in Ac^n$, we will write $(s, s') \in R_{\vec{a}}$ instead of $(\vec{a}, s, s') \in R$);*
- *$V : S \times Prop \to \{true, false\}$ assigns a truth value to each proposition in each state;*
- *$s_I \in S$ is the initial state.*

In addition, the following conditions are satisfied:

[1] State-based norms require a state of affairs to be achieved rather than particular actions to be executed. Grossi et al [23] argue that a complex action or plan may be seen as equivalent to an action of the form `achieve`(α) where α is a state of affairs. This means that action- and state-based norms can be considered equivalent on the assumption that there is a single agreed action or sequence of actions that achieves the desired state.

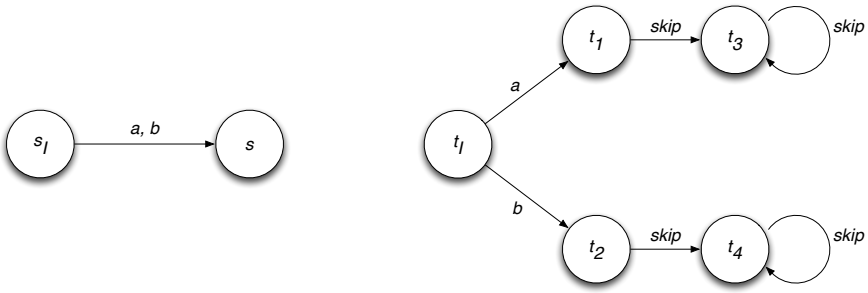

Figure 1: A transition system that does not satisfy (1) and (2) (left) and the corresponding system that satisfies (1) and (2) (right).

1. existence of successor*: for each state there exists a tuple of actions $\vec{a}$ such that* $\exists s'((s, s') \in R_{\vec{a}})$*;*
2. individual determinacy*: if* $(s', s) \in R_{\vec{a}}$ *and* $(s'', s) \in R_{\vec{b}}$ *then for all* i*,* $a_i = b_i$ *(this means that* s *is the result of a unique combination of actions executed in parallel by the agents);*
3. meaning of action propositions*:* $V(s, \mathbf{done}(a_i, i)) = true$ *iff* $\exists s'((s', s) \in R_{\langle a_1,\ldots,a_i,\ldots,a_n\rangle})$.

The first condition is a standard simplifying condition for temporal logics [30]. A transition system that does not satisfy it can easily be transformed into one where all states with no outgoing transitions have a self-loop that can be interpreted as a no-op `skip` action performed by each agent. (To be precise, to satisfy Condition (1), from the terminal state we add a `skip` link to a new state which has a `skip` link to itself.) Condition (2) requires that for each state there should be a unique tuple of actions by all agents that produces it (note this is not the same as requiring that each state has a unique predecessor state; it is possible that $(s_1, s) \in R_{\vec{a}}$ and $(s_2, s) \in R_{\vec{a}}$, $s_1 \neq s_2$, however $\vec{a}$ is the unique tuple of actions generating s). Conditions (2) and (3) are related and are imposed in order to be able to correctly interpret propositions of the form $\mathbf{done}(a_i, i)$ which mean that agent i has just executed action a_i. Together, (2) and (3) imply that for each state s and agent i, there is at most one action a such that $\mathbf{done}(a, i)$ holds in s. This is also a standard feature in agent logics, for example [15], that need to be able to express which action or event causes the current state. Again, it is easy to transform any transition system into a system that satisfies condition (2), by unravelling it [7].

Note that our transitions between states is general, in the sense that we assume each agent to contribute to the transition (more precisely, in R, each agents chooses an action). Here again, the no-op `skip` action can be of use, this time to model non-parallel actions, or actions that are performed by only a subset of the agents (the others perform `skip`, i.e., they wait.

For example, consider the transition system on the left in Figure 1 (with a single agent 1). There are no actions executable in s, so condition (1) is violated. If the agent executes action a in s_I, the resulting state is s, and the same holds if the agent

executes action b. So condition (2) is also violated. We can transform the system to the system on the right in Figure 1 that encodes the same information but has a successor for every state and allows us to make an assignment to action propositions $\mathbf{done}(a, 1)$ and $\mathbf{done}(b, 1)$ in a meaningful way. In the new system, t_1 and t_2 have the same propositional assignment as s apart from the action propositions, and t_3 and t_4 have the same assignment (apart from satisfying the action proposition $\mathbf{done}(\texttt{skip}, 1)$ where `skip` stands for the no-op action).

Given a transition system $M = (S, R, V, s_I)$, a *path* through M is a sequence $s_0, s_1, s_2, \ldots$ of states such that $(s_i, s_{i+1}) \in R_{\vec{a}}$ for $i = 0, 1, 2, \ldots$. A *fullpath* is a maximal path (where every element in the sequence has a successor) and a *run* of M is a fullpath which starts from a state $s_I \in S$. We denote runs by $\rho, \rho', \ldots$, and the state at position i on ρ by $\rho[i]$.

The truth definition for formulas is given relative to a model, a run ρ and the state at position i on ρ:

$M, \rho, i \models p$ iff $V(\rho[i], p) = true$

$M, \rho, i \models \neg\phi$ iff $M, \rho, i \not\models \phi$

$M, \rho, i \models \phi \wedge \psi$ iff $M, \rho, i \models \phi$ and $M, \rho, i \models \psi$

$M, \rho, i \models \bigcirc\phi$ iff $M, \rho, i + 1 \models \phi$

$M, \rho, i \models \phi \mathcal{U} \psi$ iff $\exists j \geq i$ such that $M, \rho, j \models \psi$ and $\forall k : i \leq k < j$, $M, \rho, k \models \phi$

Other boolean connectives are defined as usual, for example $\phi \rightarrow \psi := \neg(\phi \wedge \neg\psi)$. $\Diamond\phi$ (some time in the future) is defined as $\top \mathcal{U} \phi$, $\Box\phi$ (always in the future) is defined as $\neg\Diamond\neg\phi$. We use $\bigcirc^m$, $m \in \mathbb{N}$, to denote a sequence of $\bigcirc$ modalities of length m.

We say that a run ρ in a transition system $M = (S, R, V, s_I)$ satisfies ϕ ($M, \rho \models \phi$) if $M, \rho, 0 \models \phi$. We say that M satisfies ϕ ($M \models \phi$) if for all runs ρ in M, $M, \rho \models \phi$. A formula ϕ is valid if for all transition systems M, $M \models \phi$. A set of formulas Γ logically entails ϕ ($\Gamma \models \phi$) if for every M and ρ, if M and ρ satisfy all formulas in Γ, then $M, \rho \models \phi$.

3 Non-repeating norms

In this setting, a *group obligation* specifies a sequence of actions that should be performed collectively by a group of agents. Each step in the sequence specifies some actions that must be performed in parallel by the agents in the group. We allow actions that must be performed by more than one agent simultaneously, e.g., if two agents are necessary to move a table. The obligation specifies what must be done, and in which order; however it does not specify which actions should be performed by each agent in the group. For example a group of agents may be required to clean a room, where 'cleaning the room' is interpreted as "some agent has to vacuum the room and some agent has to do the dusting". We also consider group obligations where the order of actions is not specified, but the number of times or frequency of some action is; for example, someone has to water the plants twice a week (at least two days apart). This

can be easily expressed as a disjunction of group obligations of the first kind (the plants have to be watered on Monday and Thursday, or on Monday and Friday, or on Monday and Saturday, or on Monday and Sunday, or on Tuesday and Friday, etc.).

We assume that each agent in the group proposes one or more *individual contributions* to implementing the group norm. Each contribution specifies a set of actions the agent is prepared to perform in order to discharge the group norm. For example, an agent may specify that it is prepared to vacuum but not to dust. Where the group obligation specifies that the same action must be performed several times, we allow an agent's individual contribution to specify the maximum number of times the agent is prepared to perform the action. For example, if a group obligation for a week in a shared house involves cooking dinner each evening, an agent may specify that it is prepared to cook dinner at most twice during the week.

Before giving formal definitions of group norms and individual contribution schemes, we need some abbreviations. Let $\mathbf{hapd}(a_1 \| \ldots \| a_m)$ (where $\{a_1, \ldots, a_m\}$ is a multiset of actions) stand for actions '$a_1, \ldots, a_m$ were executed in parallel'. This is definable as

$$\mathbf{hapd}(a_1 \| \ldots \| a_m) = \bigvee_{i_1 \neq \cdots \neq i_m} (\mathbf{done}(a_1, i_1) \wedge \ldots \wedge \mathbf{done}(a_m, i_m))$$

If $A = \{a\}$, we write $\mathbf{hapd}(a)$ for $\mathbf{hapd}(A)$. Moreover, $\mathbf{hapd}(\emptyset)$ is defined as *true*.

Let $\mathbf{haps}(A_1; \ldots; A_N)$ where each A_j is a multiset of actions connected by $\|$, stand for a sequence of parallel executions of actions in multisets A_j. This is definable as

$$\mathbf{haps}(A_1; \ldots; A_N) = \bigcirc(\mathbf{hapd}(A_1) \wedge \bigcirc(\mathbf{hapd}(A_2) \wedge \bigcirc(\ldots \bigcirc \mathbf{hapd}(A_N)) \ldots))$$

where each A_i is in the scope of i nested $\bigcirc$ operators. In particular,

$$\mathbf{haps}(A_1; A_2) = \bigcirc(\mathbf{hapd}(A_1) \wedge \bigcirc \mathbf{hapd}(A_2))$$

Note that in this definition, the actions start 'tomorrow' rather than 'now', which is more or less an arbitrary decision, made for convenience. Finally, to allow expression of additional constraints such as 'watering plants should happen twice during the week, at least two days apart', we allow disjunctions of $\mathbf{haps}(A_{p^{-1}(1)}; \ldots; A_{p^{-1}(N)})$ for all possible permutations p of multisets of actions $A_1, \ldots, A_N$ which satisfy some simple condition. A permutation is a bijection $p : \{1, \ldots, N\} \to \{1, \ldots, N\}$ which shuffles the multisets of actions in the sequence, that is, it changes their position, but does not alter them internally, or their multiplicity in the sequence. For a permutation p, we will abuse notation and write $p(A_1, \ldots, A_N)$ to denote $A_{p^{-1}(1)}; \ldots; A_{p^{-1}(N)}$. Consider the plant watering example. One sequence of executions that satisfies this norm is $\mathbf{haps}(w; \emptyset; \emptyset; w; \emptyset; \emptyset; \emptyset)$. A possible permutation p of $\{1, \ldots, 7\}$ sends 1 to 2, 2 to 3, $\ldots$, 7 to 1; thus, $p(w; \emptyset; \emptyset; w; \emptyset; \emptyset; \emptyset) = \emptyset; w; \emptyset; \emptyset; w; \emptyset; \emptyset$. This satisfies the constraint that the distance between the two occurrences of w is at least 2: $|p(1) - p(4)| > 2$ (since $p(1)$ is 2 and $p(4)$ is 5). Another permutation sends 2 to 4 and 4 to 2, and leaves all other indices the same. It results in $w; w; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$, which does not satisfy the constraint. To sum up, the norm can be expressed as

$$\bigvee_{p:|p(1)-p(4)|>2} \mathbf{haps}(p(w; \emptyset; \emptyset; w; \emptyset; \emptyset; \emptyset))$$

Definition 2 (Non-repeating group norms)**.** *Given $N \in \mathbb{N}$, a group norm η is defined as follows:*

$$\eta \;:=\; \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1;\ldots;A_N))$$

where p is a permutation, and $cond(p)$ is a conjunctive condition on p expressed in terms of arithmetic operators, equality, inequality, and natural numbers between 1 and N.

We write group norms of the form

$$\bigvee_{p:\bigwedge_{j\in\{1,\ldots,N\}} p(j)=j} \mathbf{haps}(p(A_1;\ldots;A_N))$$

(where p is an identity function) as

$$\mathbf{haps}(A_1;\ldots;A_N)$$

and refer to them as non-disjunctive norms.

Again, the obligation starts being executed 'tomorrow' rather than 'at some point in the future'. All formal results in the paper would hold if we used $\Diamond \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1;\ldots;A_N))$ instead.

Example 1. *Two flatmates need to decide who contributes in which way to the tasks of dusting (d), doing groceries (g), vacuum cleaning (v) and watering the plants (w) for the next week:*

$$\eta \;=\; \mathbf{haps}(w\|g; d\|v; \emptyset; w; \emptyset; d; \emptyset)$$

That is, on Monday groceries and watering need to be done, on Tuesday, dusting and vacuuming, on Thursday the plants need to be watered again, and on Saturday dusting needs to be done. There are no constraints for Wednesday, Friday, and Sunday.

Note that $\emptyset$ means that no actions are *required* to be performed, so the agents can perform any action at this point in the sequence and still comply with the norm. In this paper, we are only concerned with obligations, and not with prohibitions on executing actions. We can extend the framework to prohibitions by using $\neg\mathbf{done}(a,i)$ expressions.

We will use $\mathbf{do}(a,i)^m$ to indicate that i is prepared to perform a at most m times:

$$\mathbf{do}(a,i)^0 := \Box(\neg\mathbf{done}(a,i))$$

and

$$\mathbf{do}(a,i)^{m+1} := \Box(\mathbf{done}(a,i) \rightarrow \bigcirc\mathbf{do}(a,i)^m)$$

Definition 3 (Individual contribution schemes)**.** *Given an agent i, an individual contribution scheme D_i is defined as $\bigvee C_i^j$ (with j ranging over disjuncts) where*

$$C_i^j \;:=\; \bigwedge_{a_k\in Ac} \mathbf{do}(a_k,i)^{n_k^j}$$

where $n_k^j \in \mathbb{N}$ is the number of times i is prepared to perform a_k as part of the offer C_i^j. We will refer to C_i^j as individual contribution offers *or simply offers.*

Sometimes we will treat D_i as a set and write $C_i^j \in D_i$ to mean that C_i^j is a disjunct in D_i.

Each C_i specifies a possible combination of actions i is prepared to contribute and does not refer to actions by other agents. For example, $\mathbf{do}(a,i)^2 \wedge \mathbf{do}(b,i)^1$ is an offer by agent i to execute action a at most twice and action b at most once.

Example 2 (Example 1 continued)**.** *Consider the following offers by the agents:*

$$\begin{array}{rcll} D_1 & = & \mathbf{do}(d,1)^1 \wedge \mathbf{do}(g,1)^7 \wedge \mathbf{do}(v,1)^1 \wedge \mathbf{do}(w,1)^7 & \\ D_2 & = & \mathbf{do}(d,2)^0 \wedge \mathbf{do}(g,2)^1 \wedge \mathbf{do}(v,2)^0 \wedge \mathbf{do}(w,2)^0 & \vee \\ & & \mathbf{do}(d,2)^1 \wedge \mathbf{do}(g,2)^0 \wedge \mathbf{do}(v,2)^0 \wedge \mathbf{do}(w,2)^2 & \end{array}$$

The offer $C_1^1 = D_1$ expresses that agent 1 does not mind doing the groceries and the watering, but is prepared to do the chores of dusting and vacuuming at most once. Agent 2 (with individual contribution scheme $C_2^1 \vee C_2^2$) is willing to either do the groceries once (C_2^1), or to do the dusting once and watering twice (C_2^2).

Note that there is a gap between a group norm and the offers of the agents, in the sense that although the agents may offer to perform all the actions needed for the group norm, in order for the group norm to be discharged, the agents need to synchronise and commit to performing actions at particular times. An *implementation* of a group obligation is a set of *individual obligations* that particular agents should perform a subset of the actions specified in one of their individual contribution offers (this is called a complete decomposition of the group obligation in [23]). Clearly an implementation should be effective, that is, if the agents discharge their individual obligations, the group norm is also discharged, and minimal, i.e., it should not create individual obligations unnecessarily.

We introduce two types of individual obligation O_i. The first kind of obligation makes sense when an action that needs to be performed by an agent has to be performed in any case, regardless of whether the preceding actions have been performed.

Definition 4 (Unconditional individual obligation)**.** *An unconditional obligation for i is a formula of the form $\bigcirc^j \mathbf{done}(a,i)$, where j is a positive natural number.*

Intuitively, $\bigcirc^j \mathbf{done}(a,i)$ denotes an obligation to perform a at step j.

The second kind of individual obligation is similar to those considered in [23]. They make sense for actions whose preconditions are created by the preceding actions. For example, where an agent is required to decorate a house and the action of decorating can only be carried out if other agents build the house first. In this case, it does not make sense to require the agent assigned to the decorating task to execute it unconditionally.

Definition 5 (Conditional individual obligation)**.** *A conditional obligation for i is a formula of the form*

$$\mathbf{haps}(A_1;\ldots;A_m) \rightarrow \bigcirc^{m+1} \mathbf{done}(a,i)$$

That is, i has an individual obligation to do a if the group obligation $\mathbf{haps}(A_1;\ldots;A_m)$ is discharged.

An individual obligation O_i for agent i is a conjunction of unconditional and conditional individual obligations for i.

Given a tuple of individual obligations by agents in a group G (consisting of k agents), $O_G = \langle O_1, \ldots, O_k \rangle$, we will identify O_G with the conjunction of those O_i's. We say that O_G *respects* the individual offer C_i^j of agent i if $O_G \wedge C_i^j \not\models \bot$. Essentially this means that O_G does not require i to perform each action a more than the maximal number of times specified by C_i^j.

Definition 6 (Implementation of a norm)**.** *Given a group norm $\eta = \bigvee \eta'$, a set of agents $G \subseteq Ag$ and their individual contributions $\{D_i \mid i \in G\}$, an* implementation of η by G *is a conjunction O_G of obligations O_i $(i \in G)$ such that*

$$\exists \eta' \forall i \in G \, \exists C_i^j \in D_i : O_G \textit{ respects } C_i^j \;\&\; \models O_G \rightarrow \eta'$$

Note that the first action A_1 in any implementation of a group obligation $\mathbf{haps}(A_1; \ldots; A_N)$ can be implemented only by unconditional obligations. Note also that if O_G is an implementation of η, then O_G is logically equivalent to a conjunction of unconditional obligations.

Example 3 (Examples 1 and 2 continued.)**.** *There is no implementation that implements η using the contributions C_1^1 and C_2^1, because on Tuesday both the dusting and the vacuuming would have to be performed by agent 1, which is impossible given Definition 1 (condition 2): $\mathbf{done}(d, 1)$ and $\mathbf{done}(v, 1)$ cannot hold in the same state since d and v are different actions. On the other hand, we can assign all individual actions to agents consistently using C_1^1 and C_2^2: agent 1 is assigned g on Monday, v on Tuesday, w on Thursday, and d on Saturday. This is consistent with its offer C_1^1. Agent 2 is assigned w on Monday and d on Tuesday; this is consistent with its offer C_2^2. The individual obligation for agent 1 is*

$$\bigcirc \mathbf{done}(g, 1) \wedge \bigcirc^2 \mathbf{done}(v, 1) \wedge \bigcirc^4 \mathbf{done}(w, 1) \wedge \bigcirc^6 \mathbf{done}(d, 1)$$

and for agent 2 $\bigcirc \mathbf{done}(w, 2) \wedge \bigcirc^2 \mathbf{done}(d, 2)$. Together both obligations entail η. Call this implementation I_1. There is one other implementation, I_2, which is as I_1 except that it assigns the watering task on Thursday to agent 2.

In order to compute individual obligations, and hence an implementation of a group norm η, we also need an auxiliary notion of an assignment of agents to actions in η. Essentially, given a group norm η and a set of agent offers, an assignment specifies which agent is asked to perform which individual action in η, and which agent offer justifies the assumption that the agent is willing to perform the assigned action.

Definition 7 (Assignment)**.** *An* assignment of agents in $G \subseteq Ag$ to actions in $\eta = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1; \ldots; A_N))$ *is a function f that for some disjunct $\eta' = \mathbf{haps}(p(A_1; \ldots; A_N))$, for every $A_j \neq \emptyset$ in η' assigns an agent $i \in G$ and a contribution C_i to every element a of A_j subject to the following constraints:*[2]

[2]The arguments of f include the position of A_j in η' and the position of a in A_j, since η' may contain several identical A_j and A_j may contain several occurrences of a.

C1 *if $f(A_j, j, a, l) = (i, C_i)$ for k different j (in other words, the agent is assigned to k different occurrences of a in η') then $\mathbf{do}(a, i)^m$ for $m \geq k$ is a conjunct in C_i;*

C2 *if $f(A_j, j, a, l) = (i, C_i)$ and $f(A_j, j, b, l') = (k, C_k)$ and $a \neq b$ or $l \neq l'$, then $i \neq k$ (only one action can be executed by the agent i in a single transition); and*

C3 *if $f(A_j, j, a, l) = (i, C_i)$ and $f(A_k, k, b, l') = (i, C'_i)$, then $C_i = C'_i$ (only one offer by i is used by the assignment throughout).*

Example 4 (Assignment)**.** *As in the previous examples, let the norm be*

$$\eta \;=\; \mathbf{haps}(w \| g; d \| v; \emptyset; w; \emptyset; d; \emptyset)$$

If the agents' offers are specified by

$$\begin{array}{lcl} C_1^1 & = & \mathbf{do}(d,1)^1 \wedge \mathbf{do}(g,1)^7 \wedge \mathbf{do}(v,1)^1 \wedge \mathbf{do}(w,1)^7 \\ C_2^1 & = & \mathbf{do}(d,2)^0 \wedge \mathbf{do}(g,2)^1 \wedge \mathbf{do}(v,2)^0 \wedge \mathbf{do}(w,2)^0 \\ C_2^2 & = & \mathbf{do}(d,2)^1 \wedge \mathbf{do}(g,2)^0 \wedge \mathbf{do}(v,2)^0 \wedge \mathbf{do}(w,2)^2 \end{array}$$

then the following function describes an assignment:

- $f(w \| g, 1, w, 1) = (2, C_2^2)$
- $f(w \| g, 1, g, 1) = (1, C_1^1)$
- $f(d \| v, 2, d, 1) = (2, C_2^2)$
- $f(d \| v, 2, v, 1) = (1, C_1^1)$
- $f(w, 4, w, 1) = (1, C_1^1)$
- $f(d, 6, d, 1) = (1, C_1^1)$

It is clear that f above is a function (it assigns a value to every component of every joint action in η apart from $\emptyset$). Condition **C1** *holds since no agent i is assigned two different actions to perform at the same step (for the same joint action in η). Condition* **C2** *holds: agent 1 is assigned g, v, w and d to perform, which is consistent with its offer to perform each of these actions at least once (more often for g and w), and agent 2 is assigned w and d, which is also consistent with C_2^2. Condition* **C3** *holds because only one offer is used for each agent (C_1^1 for 1 and C_2^2 for 2).*

The following theorem will be useful for analysing the implementation of a norm as a computational problem.

Theorem 1. *Every assignment of agents to actions in η satisfying the conditions of Definition 7 gives rise to an implementation of η, and every implementation gives rise to such an assignment.*

Proof. Assume that we have an assignment f for a group norm η that uses some disjunct $\eta' = \mathbf{haps}(p(A_1; \ldots; A_N))$ of η. Let $\eta' = \mathbf{haps}(A'_1; \ldots; A'_N)$ By **C3**, for each agent i involved in the assignment, there is a single contribution C_i. Given the

assignment, generate O_i as $\bigwedge_{f(A'_j,j,a,l)=(i,C_i)} \bigcirc^j \mathbf{done}(a,i)$. Clearly $\bigwedge O_i$ respects C_i by **C1**. $\bigwedge_{i\in G} O_i$ is satisfiable by **C2**. Finally since f is an assignment, any run that satisfies $\bigwedge_{i\in G} O_i$ also satisfies η'. Hence, $\bigwedge_i O_i$ is an implementation of η.

Now assume that we have an implementation $\bigwedge_{i\in G} O_i$ for $\eta = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1;\ldots;A_N))$. By definition, at least one of the disjuncts $\eta' = \mathbf{haps}(p(A_1;\ldots;A_N))$ is entailed by the implementation. Let $\eta' = \mathbf{haps}(A'_1;\ldots;A'_N)$. We will show how to extract an assignment f for η' from the implementation. First of all, to satisfy **C3**, we assign only one contribution C_i with which O_i is consistent to every $i \in G$. Now we construct f for each of $A'_1,\ldots,A'_N$ in turn. Since $\bigwedge O_i \models \eta'$ and $\eta' \models \mathbf{haps}(A'_1)$, there are enough conjuncts in $\bigwedge O_i$ to make sure $\mathbf{done}(a,i_j)$ holds for every $a \in A'_1$ and there are enough different i_j to for every occurrence of a in A'_1 (no agent is asked to perform more than one action in parallel since $\bigwedge O_i$ is satisfiable). We take some subset of those to assign to $f(A'_1,a)$. Similarly in order for $\bigwedge O_i$ to entail $\mathbf{haps}(A'_1;\ldots;A'_{m+1})$ provided it entails $\mathbf{haps}(A'_1;\ldots;A'_m)$ there must be contributions in $\bigwedge O_i$ of agents promising to execute an action in A'_{m+1} after $\mathbf{haps}(A'_1;\ldots;A'_m)$ (or in $m+1$ time steps unconditionally), and enough of them to entail $\mathbf{haps}(A'_1;\ldots;A'_{m+1})$. Assign some subset of those agents to actions in A'_{m+1}. □

Now we consider the complexity of the problem whether an implementation of η by G exists. In the statement of the theorem below, the size of the input to the problem is measured in the size of the compact representation of η (the size of expression $\bigvee_{p:cond(p)} \mathbf{haps}()$) and D_i (using $\mathbf{do}(a,i)^j$), rather than the size of the underlying LTL expressions. Otherwise, since the LTL expressions are exponential in the size of the compact representation, there exists a trivially polynomial algorithm for solving the problem.

Theorem 2. *Given a group norm η, a group of agents G, and agent contributions D_i for $i \in G$, the problem of whether an implementation of η by G exists is NP-complete.*

Proof. By Theorem 1, the problem of finding an implementation can be reduced to the problem of finding an assignment. For membership of NP, observe that an assignment can be guessed in time polynomial in the size of the group norm and checked that it satisfies the conditions **C1-C3** in time polynomial in the group norm and the set of agents' contributions. For NP-hardness, we reduce SAT to the problem of finding an assignment of agents to actions in a simple non-disjunctive group norm. Let ϕ be a propositional formula in CNF containing n variables and k clauses. Without loss of generality, we assume that each clause is unique and none of them contains both p_i and $\neg p_i$ for some variable p_i. The corresponding group norm will be

$$\eta_\phi = c_1;\ldots;c_k$$

where c_j is an action corresponding to making the jth clause in ϕ true. Let G contain n agents, one for each propositional variable p_i in ϕ. Each agent i has two offers. Intuitively, the offer C_i^t corresponds to setting p_i to true and the offer C_i^f corresponds to setting p_i to false.

$$C_i^t = \bigwedge_{p_i\in c_j} \mathbf{done}(c_j,i), \quad C_i^f = \bigwedge_{\neg p_i\in c_j} \mathbf{done}(c_j,i)$$

Since we assume that each clause is unique, the agents offer to make each c_j true at most once. Now assume that we have a function f that assigns to clauses pairs (i, C_i^t) or (i, C_i^f). By **C3**, only one of C_i^t or C_i^f is used for each i in this assignment. Hence for each p_i where $i \in G$ (i was used in the assignment of agents), we can extract a unique assignment of a truth value *true* or *false* to p_i. Because of the way the offers were defined, this assignments of truth values to p_i for $i \in G$ will make all the clauses true. □

However, there is a natural special case for which the problem of whether an implementation of a group obligation exists is tractable.

Theorem 3. *Given a non-disjunctive group norm η, a group of agents G, and agent contributions where*

1. *all agent contributions D_i are identical*
2. *D_i does not contain disjunctions*

the problem of whether an implementation of η by G exists is in PTIME.

Proof. Intuitively, the algorithm to solve the problem would simply allocate arbitrary agents to arbitrary actions in η while ensuring that: (i) the same agent is not assigned to two action executed in parallel, and (ii) the agent's offer is respected. This process continues until all actions have an assigned agent, or no agent can be assigned to an action without violating constraints (i) and (ii). A simple check is sufficient to ensure that it is possible to complete an assignment without violating conditions (i) and (ii). In order to make sure that (i) is not violated, we need to check that the maximal number of actions that must be executed in parallel at any step in η (that is, $max_1^N(|A_j|)$ where $\eta = \mathbf{haps}(A_1; \ldots; A_N)$) is at most $|G|$. To make sure that (ii) is not violated, we need to count, for each action type a occurring in η, how many times it occurs in η, and compare this number with the number of times the action is offered in D_i multiplied by $|G|$. If each action is offered at least as many times as it is required by η, an implementation exists. □

3.1 Minimality

A natural and desirable property of an implementation of a group norm is that the agents are not obliged to do more than the norm requires.

Definition 8 (Minimality)**.** *Let η be a group norm. Let $O_1, \ldots, O_k$ be individual obligations for agents in G, and $I = O_1 \wedge \cdots \wedge O_k$.*

- *I is a minimal implementation of η if it is an implementation of η and there is no implementation $I' = O'_1 \wedge \cdots \wedge O'_k$ of η for which both $\models I \rightarrow I'$ and $\not\models I' \rightarrow I$.*
- *I is an i-minimal implementation of η if there is no obligation O'_i for i such that $(O_1 \wedge \cdots \wedge O'_i \wedge \ldots O_k)$ is an implementation of η for which both $\models O_i \rightarrow O'_i$ and $\not\models O'_i \rightarrow O_i$.*

- *I is an individually minimal implementation of η if it is an i-minimal implementation for every $i \in G$.*

Clearly, a minimal implementation I of η is individually minimal. The opposite also holds:

Theorem 4. *Let $I = O_1 \wedge \cdots \wedge O_k$ be an implementation of a group norm $\eta = \bigvee_{p:con(p)} \mathbf{haps}(p(A_1; \ldots; A_N))$ by G. Then I is a minimal implementation iff I is an individually minimal implementation.*

Proof. The left to right direction is obvious, so consider $I = O_1 \wedge \cdots \wedge O_k$. Since it is an implementation of η, using Theorem 1 we can use an assignment f to write each individual obligation O_i in the following normal form:

$$O_i = \bigcirc(\gamma_{i_1} \wedge \bigcirc(\gamma_{i_2} \wedge \ldots \bigcirc\gamma_{i_N}) \ldots)$$

where each γ_{i_k} is of the form $\mathbf{done}(a, i)$ (i is required to do a at step k) or $\top$ (no requirement for i at step k), and there is a contribution C_i so that the number of times $\mathbf{done}(a, i)$ occurs for every a is consistent with C_i. Now let $\Gamma_j = \bigwedge_{i \leq n} \gamma_{i_k}$. It is not difficult to see that I is equivalent to

$$O = \bigcirc(\Gamma_1 \wedge \bigcirc(\Gamma_2 \wedge \ldots \bigcirc\Gamma_N) \ldots)$$

Now, if I is not minimal, there is a logically weaker implementation $I' = \bigcirc(\Gamma'_1 \wedge \bigcirc(\Gamma'_2 \wedge \ldots \bigcirc\Gamma'_N) \ldots)$. However, since no $\mathbf{done}(a, i)$ entails any $\mathbf{done}(a', i')$ unless $a = a'$ and $i = i'$, the implementation I' can only be weaker than I if there is some Γ_j and Γ'_j for which some $\models \Gamma_j \rightarrow \mathbf{done}(a, i)$ while $\not\models \Gamma'_j \rightarrow \mathbf{done}(b, i)$ for any action b (that is, Γ_j requires i to do a at step j, while Γ'_j does not impose a requirement on i at j). But then, O_i is not minimal, since replacing $\mathbf{done}(a, i)$ by $\top$ in O_i would be a weaker obligation for i, and hence I is not individually minimal. □

Given the result above, it is clear that the problem of computing a minimal implementation is no harder than the problem of computing an implementation, since it is possible to check if an implementation (or rather the corresponding assignment) is individually minimal in polynomial time.

3.2 Fairness

Now we arrive at the main concern of this paper, that is how to define a notion of group norm implementation that agrees with the *social norms* accepted by the agents as a way to regulate the *fairness* of task assignments.

Some implementations of a group norm may be better than others from the point of view of the group's or the wider organisation's notion of fairness as captured in social norms. For example, fairness may require that all agents should contribute equally to the implementation of the group norm, or that agents with less experience are required to contribute less. LTL offers a natural setting to consider *fairness* constraints on implementations. By fairness constraints in this setting we do not mean just the notion of fairness as defined for processes in computer science (e.g., every request will be

eventually granted). Instead we mean some additional constraints on possible implementations that reflect the organisation's view of what is reasonable to require from the agents. For example, it could be that the organisation does not consider it fair that the same agent performs an action a (for example, a work shift) twice in a row:

$$\Box(\bigwedge_{i \in G}(\mathbf{done}(a,i) \rightarrow \bigcirc \neg \mathbf{done}(a,i)))$$

Another example is that each agent gets a rest from all chores every seventh day:

$$\Box(\bigwedge_{i \in G}(\neg\chi_i \vee \bigcirc\neg\chi_i \vee \ldots \vee \bigcirc^6\neg\chi_i))$$

where $\chi_i = \bigvee_{a \in \{d,g,v,w\}} \mathbf{done}(a,i)$.

Definition 9 (Fair implementation)**.** *Let ϕ be an LTL formula expressing a fairness constraint. An implementation I of a group norm is* fair *with respect to ϕ (or ϕ-fair) if $\not\models I \rightarrow \neg\phi$ (in other words, if I is consistent with ϕ).*

Checking fairness of an implementation can be done by checking whether $I \wedge \phi$ is satisfiable.

If group norms are assumed to be fixed length sequences of actions, it arguably does not make sense to consider arbitrary LTL formulas as fairness constraints. In fact, most natural fairness constraints in human work allocation do not have the form 'everyone *eventually* gets a holiday' but 'everyone gets a holiday after working for n months'. We therefore restrict the syntax of fairness constraints to talk about fixed finite patterns of actions.

Definition 10 (Fairness constraint)**.** *An LTL formula ϕ is a fairness constraint if it is of the form $\Box\psi$, where ψ only contains $\bigcirc$ modalities.*

Examples of fairness constraints $\Box\psi$ are as follows, where N is a given number: (1) no agent i performs an action a twice in the next N steps, without another agent i performing it in between those occurrences; (2) agents $i_1, \ldots, i_m$ take perfect turns in all occurrences of action a; (3) if action a happens k times in the next N steps, then at least m different agents should be involved in their execution; (4) agent i is allowed to do something other than any of $a_1, \ldots, a_k$ at least once in every k steps; and (5) if i does a then j does it within k steps.

Example 5. *Consider the implementation that we gave in Example 3. Implementation I_1 satisfies fairness constraint (1) above, but I_2 does not; likewise for (2). Constraint (3) holds for I_1 in the sense that every action a that occurs twice in the week is assigned to a different agent, again I_2 does not have this property. Regarding (4), note that each agent is allowed to do anything other than one of the duties $\{w, g, d, v\}$ in any three steps; however, none of the implementations allows agent 1 to do anything other than $\{g, v\}$ at least once in every 2 steps, throughout the week. Finally, constraint (5) does not hold for I_2, with $a = w$ and $k = 4$, but does hold for I_1 for those values.*

4 Repeating norms

In the previous section, we looked at group norms that correspond to performing some group task/obligation once. In state-based terms, such norms correspond to achievement goals: a sequence of actions that must be executed in order to achieve a certain desirable state. In this section, we consider the case where a group norm relates, in state-based terms, to a maintenance goal: some condition needs to be maintained in perpetuity. In order to achieve this condition, some group task has to be executed periodically. For example, every week the agents in a household need to execute some combination of cleaning, shopping and cooking tasks: $A_1; \ldots; A_7$.

The norm itself requires them to iterate this sequence forever, which we write as $\mathbf{haps}(A_1; \ldots; A_7)^\infty$. We will refer to the number of sequentially composed actions in the repeated sequence in η as the cycle of η, $c(\eta)$ (in the example above, $c(\eta) = 7$).

An infinite repetition of a sequence $A_1; \ldots; A_N$ can be defined in LTL as follows:

$$\begin{aligned}\mathbf{haps}(A_1; \ldots; A_N)^\infty \quad = \quad & \mathbf{haps}(A_1; \ldots; A_N) \wedge \\ & \Box(\mathbf{haps}(A_1; \ldots; A_N) \rightarrow \bigcirc^N \mathbf{haps}(A_1; \ldots; A_N))\end{aligned}$$

similarly for disjunctive norms.

Definition 11 (Repeating group norm with cycle N)**.** *A repeating norm with cycle N is an obligation to repeat $A_1; \ldots; A_N$ infinitely often: $\eta = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1; \ldots; A_N))^\infty$ where p is a permutation, and $cond(p)$ is a conjunctive condition on p expressed in terms of arithmetic operators, equality, inequality, and natural numbers between 1 and N. As before, we refer to norms where $cond(p)$ is identity as non-disjunctive repeating norms, and write them as $\eta = \mathbf{haps}(A_1; \ldots; A_N)^\infty$*

The syntax of agents' individual contribution schemes is similar to Definition 3, apart from the addition of the norm cycle N: $\mathbf{do}(a, i)^{m,N}$ means that the agent offers to perform a at most m times in every $N = c(\eta)$. The most straightforward way to define this in LTL is to rule out all patterns of length N where the agent performs a more than m times, or, equivalently, to state that in every pattern of length N there are at least $N - m$ steps when the agent is *not* performing a. Let $\mathbf{K}^{m,N} = \{K \subseteq \{1, \ldots N\} \mid |K| = N - m\}$. Intuitively, this defines all possible combinations of a-free steps in a pattern of length N if the agent does a at most m times. Let $\mathbf{not}^K(a, i)$ for $k \in \mathbf{K}^{m,N}$ stand for $\bigwedge_{k \in K} \bigcirc^k \neg \mathbf{done}(a, i)$. This formula says that the agent does not do a on each of the time steps in K. Then $\mathbf{do}(a, i)^{m,N} = \bigvee_{K \in \mathbf{K}^{m,N}} \mathbf{not}^K(a, i)$ says that the agent does a at most m times in N steps. Finally, to make this apply not just to the first N steps but indefinitely, the offer is prefixed with $\Box$.

Definition 12 (Individual contribution schemes for repeating norms)**.** *Given an agent i, and a repeating norm η with cycle N, an individual contribution scheme D_i for η is defined as $\bigvee C_i$ where*

$$C_i^j := \bigwedge_{a_k \in Ac} \Box \mathbf{do}(a_k, i)^{n_k^j, N}$$

where $n_k^j \leq N$ for all k.

Example 6. *The existing examples 1 and 2 can be straightforwardly transformed in the setting of repeating norms. That is,* $\mathbf{haps}(w\|g; d\|v; \emptyset; w; \emptyset; d; \emptyset)^\infty$ *expresses that in all coming weeks, on Monday watering and groceries need to be done, etc. Moreover, the offer from Example 2 for agent 1 becomes*

$$C_1^1 \quad = \quad \Box\mathbf{do}(d,1)^{1,7} \wedge \Box\mathbf{do}(g,1)^{7,7} \wedge \Box\mathbf{do}(v,1)^{1,7} \wedge \Box\mathbf{do}(w,1)^{7,7}$$

When an agent commits to a repeated group obligation, it may need to commit to several different 'shift patterns': for example, one week the agent will be working on Mondays, and in another week on Tuesdays. This means that the definition of an individual obligation for repeated norms becomes a sequence of shift patterns:

Definition 13 (Unconditional individual obligation for repeating norms)**.** *Given an agent* i*, and a repeating norm* η *with cycle* N*, an unconditional individual obligation for* i *with respect to* η *is a formula of the form*

$$\Box(\Gamma_1 \rightarrow \bigcirc(\Gamma_2 \wedge \ldots \wedge \bigcirc^N \Gamma_m \wedge \bigcirc^{N+1}\Gamma_1)\ldots)$$

for some natural number m*, where each* Γ_n*,* $1 \leq n \leq m$*, describes all actions agent* i *is committed to perform in the* n*th* N*-step period, or 'shift'.* Γ_n *is of the form* $\bigcirc\gamma_1 \wedge \bigcirc^2\gamma_2 \wedge \ldots \wedge \bigcirc^N\gamma_n$ *where each* γ_j *is either* $\mathbf{done}(a_j, i)$ *or* $\top$ *if* i *is not committed to performing any action at step* j*.*

An implementation of a repeating norm by a set of agents G is as before a conjunction of obligations $I = \bigwedge O_{i\in G}$ such that $\models I \rightarrow \eta$ and I is consistent with agent offers. A minimal implementation is defined as before, and the same type of fairness constraints as in the previous section can be applied to repeating norms.

A *single cycle assignment* of agents to actions in $\eta = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1; \ldots; A_N))^\infty$ is defined as an assignment for a non-repeating norm $\eta' = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1; \ldots; A_N))$. Clearly any single cycle assignment repeated every N steps gives rise to an implementation for a repeating norm. However for repeating norms it makes sense to consider implementations obtained by 'gluing' several different assignments together and repeating the resulting pattern, as repetition affects fairness in a non-trivial way. Considering the example fairness constraints in the previous section, we can see that 'gluing' together two (even identical) assignments satisfying fairness constraint (2) stated at the end of Section 3 may make the resulting implementation unfair (if the last occurrence of a in the implementation is done by i_k with $k < m$), and also two unfair implementations (not satisfying fairness constraint (5) in Section 3, for instance) may become fair when glued together.

A consequence of this is that when solving the problem of finding a ϕ-fair implementation of a repeating obligation η, it is not sufficient to consider only single cycle assignments. If none of those when repeated correspond to a fair implementation of η, this does not mean that η has no fair implementation. We may need to consider a combination of several assignments. For example, let $\phi = \Box(\bigwedge_{i\in G}(\mathbf{done}(a,i) \rightarrow \bigcirc\neg\mathbf{done}(a,i)))$ and $\eta = \mathbf{haps}(a)^\infty$. The cycle of η is 1. Suppose there are two possible one cycle assignments for η, one where agent 1 does a, and another where agent 2 does a. If either of them alone is repeated, the resulting implementation is not

fair: either all occurrences of action a are done by agent 1, or all of them are done by agent 2. Clearly, if we combine these two one cycle assignments or produce a one cycle assignment to an 'unravelling' of η of length two: $\eta^2 = \mathbf{haps}(a; a)^\infty$, we can produce an assignment that gives rise to a fair implementation: for example, the first a is done by agent 1 and the second by agent 2. However to solve the problem of finding a ϕ-fair implementation of a repeating norm η (if it exists) we need to know how long such an unravelling should get before we give up.

For $\eta = \bigvee_{p:cond(p)} \mathbf{haps}(p(A_1; \ldots; A_N))^\infty$, we will call

$$\eta^m = \mathbf{haps}(A'_1; \ldots; A'_N; \ \ldots \ ; A'_1; \ldots; A'_N)^\infty \quad (m \text{ times})$$

where each $A'_1; \ldots; A'_N$ is a (possibly different) disjunct of η, an m-unravelling of η.

Theorem 5. *Let η be a group obligation with cycle N that has k different one cycle assignments $S_1, \ldots, S_k$ (for different disjuncts and different assignments to each disjunct), and ϕ be a fairness constraint of modal depth d.*[3] *If a ϕ-fair implementation of η exists, then there exists a ϕ-fair implementation of η based on the a single cycle assignment to an m-unravelling of η, where $m \leq max(k, k^{d/N+1})$.*

Proof. Let τ be an assignment corresponding to a fair implementation of η. Without loss of generality, we can assume that τ corresponds to a (possibly infinite) sequence of one cycle assignments for η, $S_{i_1}, \ldots, S_{i_t}, \ldots$. Given τ, we are going to construct a sequence of assignments of length m, that is, some sequence $\tau' = S'_1, \ldots, S'_m$ (a single cycle assignment to η^m) that when repeated infinitely often, gives rise to a ϕ-fair implementation of η.

Note that τ (or any other assignment of agents to actions) corresponds to a description of a run in terms of action propositions. Observe that a run violates ϕ if it has a pattern of d consecutive states $s_1, \ldots, s_d$ that is a counterexample to ϕ. Clearly, τ describes a run that does not contain such a counterexample sequence of states (since it corresponds to a fair implementation). Note also that none of single-cycle implementations of η that occur in τ contain such a sequence of states (otherwise τ would not satisfy ϕ).

Let us first consider a simpler case when $d < N$. Then the only way a sequence of single-cycle assignments $S_{j_1}, \ldots, S_{j_n}$ would violate ϕ is when there is a sequence on the 'joint' between two assignments S_{j_l} and $S_{j_{l+1}}$ that violates it. Let us build a sequence of assignments of length at most k that does not have such a violating joint. For convenience, let us say that S_{j_l} and $S_{j_{l+1}}$ *compose* if their concatenation does not contain a subsequence violating ϕ. To start building our sequence of length at most k,

[3] Modal depth is the depth of nesting of modal operators. Formally, $md(\phi)$ (for modal depth of ϕ) is defined as follows:

- $md(p) = 0$
- $md(\neg\phi) = md(\phi)$
- $md(\phi \wedge \psi) = max(md(\phi), md(\psi))$
- $md(\bigcirc\phi) = 1 + md(\phi)$
- $md(\phi \mathcal{U} \psi) = 1 + max(md(\phi), md(\psi))$.

For the defined connective $\Box$, $md(\Box\phi) = md(\neg\top \mathcal{U} \neg\phi) = 1 + md(\phi)$.

take the first assignments in τ, S_{j_1}. Clearly it composes with some other assignments, since τ does not violate ϕ. If S_{j_1} composes with itself (there is a subsequence in τ that has $S_{j_1}; S_{j_1}$), we are done: $\tau' = S_{j_1}$. Otherwise we consider the first two assignments in τ, $S_{j_1}; S_{j_2}$. If S_{j_2} composes with itself, we are done and $\tau' = S_{j_2}$, or if it composes with S_{j_1}, then $\tau' = S_{j_1}; S_{j_2}$. Otherwise we consider a 3-element prefix of τ. Note that eventually we are going to encounter S_{j_f} which composes with $S_{j_{f+1}}$ that already occurs in the prefix of the sequence (the maximal possible value for f is k, the total number of single-cycle implementations). Then we set τ' to be the subsequence of the current sequence that starts from the first occurrence of $S_{j_{f+1}}$ and continues until S_{j_f}. Clearly, τ' has length at most k and nowhere in the 'joints' of the single cycle implementations in τ' there is a counterexample to ϕ (including the joint of τ' to itself).

Now let $d \geq N$. Then a counterexample sequence $s_1, \ldots, s_d$ can span multiple single cycle assignments. Let $d \leq p \cdot N$ (p iterations of N are required to produce a counterexample to ϕ, so $p \leq (d/N) + 1$). Then we make a set of 'viable multi-cycle assignments' $Z_1, \ldots, Z_{k^p}$ of all p-sequences of single-cycle assignments occurring in τ. We treat them as we treated single cycle assignments S_i before, as the building blocks for τ'. Similarly to the previous construction, we are bound to start to encounter the same 'viable multi-cycle assignments' after k^p steps. So τ' is of length at most $k^{d/N+1}$. □

This means that to construct a ϕ-fair implementation of η, we only need to consider assignments to sequences of actions of length $m \leq max(k, k^{d/N+1})$. This gives us an (exponential) algorithm for finding a ϕ-fair implementation of a repeating norm η (generate all possible one cycle assignments and then check all concatenations of them of length m for consistency with ϕ).

5 Related work

Social laws have long been recognised as an important mechanism to facilitate coordination in multi-agent systems [12], and there exists an extensive literature on formal approaches to social laws and norms, for example, [33, 28, 35, 32, 1, 8, 17, 10, 11, 3]. Logics for social laws often build upon dynamic or temporal logics such as LTL, CTL, ATL and STIT. (A recent paper that questions the suitability of temporal logic as a framework to model norms is [21]; a response is given in [5]). Most of this work specifies norms and their effects on the multi-agent system semantically by labelling certain transitions as forbidden (in the case of prohibitions) or labelling certain states as 'green' (good, or encouraged states) or 'red' (forbidden ones, see, e.g., [27]). In this paper, we model only obligations (and not prohibitions) and specify obligations in the object language.

Group norms have been studied in, for example, [2, 23]. Our definition of non-repeating group norms is a generalisation of that given in [23]. However in [23] the emphasis is on formalising synchronisation, and they abstract from the problem of computing individual obligations necessary to implement a group norm. In [2] group norms are considered at a much more abstract level. In that framework, a group norm is defined as making a state formula ϕ true, and the set of agents responsible for carrying

out (an abstract STIT-like) action to achieve ϕ and the set of agents responsible for the violation are explicitly given as part of the norm. Our approach is closer to [23], in that the notion of agents responsible for the violation of a group norm given a particular implementation is definable in terms of the set of individual obligations. An agent that does not fulfil an unconditional obligation is responsible for a violation, and an agent with a conditional obligation where the condition of which has not been made true, is not responsible.

Team formation and coordination of joint actions has been extensively studied in Artificial Intelligence, for example [18, 24, 34]. However the emphasis of that work is on efficient and flexible team work rather than on fairness. An exception is the work in [6], where the authors consider the problem of repeatedly choosing actions (which could, for example, be actions that assign jobs to people) in a fair way, where fairness has a decision theoretic interpretation in terms of minimising loss for worse-off beneficiaries of actions. The motivation of their work is very similar to our problem of finding a fair implementation of a repeated norm, but they have a specific notion of fairness and reduce the problem of fair selection of actions to an optimisation problem.

Another strand of related work is found in the behaviour composition literature, e.g., [36, 29]. The behaviour composition problem aims at the synthesis of a controller which can implement a desired target specification by controlling a collection of behaviours, running in some environment. Roughly, a controller corresponds to our notion of implementation, the target specification corresponds to our norm, and the behaviours correspond to our individual contribution schemes. However, standard behaviour composition does not consider parallel composition of actions, nor fairness constraints.

The problem we address is also somewhat related to several other topics in AI and Computer Science generally. In the remainder of this section, we briefly summarise the key similarities and differences between our approach and some of this work.

Our setting has some similarities to resource allocation problems discussed in the social choice literature [9] and in particular the fair allocation of indivisible goods [13]. However, there are some important differences. In fair allocation, a set of (indivisible) objects must be allocated to a set of agents. Each agent is assigned a subset of objects, and is assumed to have a preference ordering over sets of objects (bundles). Our notion of an individual contribution scheme has some similarities to logic-based representation languages for preferences over bundles [26]. However in our setting an offer is *maximal*—an agent is satisfied if the set of actions it is assigned is a subset of one of its offers. In fair allocation, the converse holds: agents are satisfied if their goal is a subset of the bundle they are allocated. Moreover, in resource allocation, an allocation which assigns an agent a bundle containing none of its preferred objects may still be complete, in the sense of allocating each object to some agent. However, in our setting, we assume agents will simply refuse to perform any action (or combination of actions) not included in one of its offers. Any allocation that assigns a 'refused' action to an agent is thus incomplete in an important sense (i.e., the action will not be performed). Finally, the notion of fairness we consider (essentially any LTL formula) is much more general than the notion of envy-freeness found in ordinary resource allocation.

As noted in Section 3.2, our work differs from the notion of fairness used in resource allocation problems in computing. In such settings, a resource allocation policy

or protocol is assumed, and the aim is to verify that the policy satisfies some LTL property (e.g., that a request for a resource is eventually granted) under some fairness assumption, such as strong or weak fairness. In our work, the allocation of agents to actions is computed rather than assumed, and LTL is used to express arbitrary fairness constraints on the allocation, rather than to specify that, e.g., an action will eventually be performed.

The problem of allocating a sequence of parallel tasks to a set of resources has some similarities to production (job shop) scheduling [22]. However there are important differences. In general production scheduling each job consists of a set of operations, where each operation must be performed by a single machine. Each job may have a release, a due date, and jobs may be partially ordered. In contrast our language of norms specifies not just which actions (operations) should be performed, but when they should be performed (the release and due dates are the same), and at each point in time, a set of actions must be performed in parallel. Moreover, we consider repeating patterns of actions. This has no counterpart in production scheduling, where the aim is simply to process a set of jobs once (the jobs do not repeat). Lastly, we consider fair allocations of tasks to agents, where fairness is defined as any pattern LTL-expressible on a fixed length interval. In scheduling the aim is to find a (near) optimal schedule, where the optimality criterion is specified in terms of makespan, tardiness or some other cost function.

Our work is also related to the problem of personnel scheduling or rostering [19]. In personnel scheduling, the aim is to find a (near) optimal allocation staff to 'shifts', while taking into account staff availability, preferences etc. Shifts (and their associated constraints, e.g., tasks that must be performed, levels of seniority of staff that must be present etc.) correspond to the parallel execution of a set of actions in our setting, and staff availability has some similarities to our notion of individual contribution schemes. Similarly to our work, rosters are often repeating (the same shift pattern may be used for several weeks or months), and schedules may be subject to 'fairness' constraints, e.g., legal regulations, organisation personnel policies, etc. However our notion of fairness is more general than the constraints normally considered in personnel scheduling.

There is also work on allocating resources to tasks in a workflow, e.g., [31]. This work considers parallel execution of tasks, and constraints on allocation of the resources to tasks (e.g., if task1 is executed by agent1, then task2 must also be executed by agent 1). However this work does not consider cyclic or repeating workflows, and constraints on resource allocation are restricted to cost constraints and control constraints, rather than any pattern expressible in LTL over a fixed length interval of time as in our approach.

6 Conclusions and Future Work

We describe an approach to expressing and reasoning about implementations of group obligations taking into account fairness constraints. The notion of fairness constraints was introduced in [4]. In this paper, we extend fairness constraints to group norms in which the order in which actions are performed is not fixed. We formalise our approach in Linear Time Temporal Logic, and give some preliminary results. Our approach has

a number of limitations. We model only obligations and do not consider prohibitions. Our notion of group obligation does not allow arbitrary disjunctions of tasks, but only disjunctions over some permutations of the same set of tasks. We also consider only a restricted class of fairness constraints. Moreover, neither the norms nor the offers are state-based, in the sense that one can refer to the state of the environment to say whether some action should be executed, or whether one is able to perform it. Relaxing these limitations, and a more compact syntax for representing, for example, individual offers, are the subject of future work. For example, it would be interesting to explore using automata for the formulation of the problem and transfer of results from the field of behaviour composition. It would also be interesting to investigate relaxation of norms and fairness constraints when a fair implementation does not exist.

Acknowledgements We thank the anonymous referees of CLIMA 2014 for their insightful comments that helped to improve the paper.

References

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Conservative social laws. In *Proc. 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 49–54, 2012.

[2] H. Aldewereld, V. Dignum, and W. Vasconcelos. We ought to; they do; blame the management! – a conceptualisation of group norms. In *Proc. 15th International Workshop on Coordination, Organisations, Institutions and Norms (COIN)*, 2013.

[3] N. Alechina, M. Dastani, and B. Logan. Reasoning about normative update. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[4] N. Alechina, W. van der Hoek, and B. Logan. Fair allocation of group tasks according to social norms. In N. Bulling, L. van der Torre, S. Villata, W. Jamroga, and W. Vasconcelos, editors, *Computational Logic in Multi-Agent Systems, 15th International Workshop (CLIMA XV)*, volume 8624 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2014.

[5] Natasha Alechina, Mehdi Dastani, and Brian Logan. Expressibility of norms in temporal logic. *CoRR*, abs/1608.06787, 2016.

[6] G. Catalin Balan, D. Richards, and S. Luke. Long-term fairness with bounded worst-case losses. *Autonomous Agents and Multi-Agent Systems*, 22(1):43–63, 2011.

[7] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[8] G. Boella and L. van der Torre. Delegation of power in normative multiagent systems. In *Proc. 8th International Workshop on Deontic Logic in Computer Science (DEON)*, 2006.

[9] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A.D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[10] J. Broersen, R. Mastop, J-J. Ch. Meyer, and P. Turrini. A deontic logic for socially optimal norms. In *Proc. 9th International Conference Deontic Logic in Computer Science (DEON)*, volume 5076 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2008.

[11] N. Bulling, M. Dastani, and M. Knobbout. Monitoring norm violations in multi-agent systems. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 491–498. IFAAMAS, 2013.

[12] C. Castelfranchi. Modelling social action for AI agents. *Artificial Intelligence*, 103(1-2):157–182, 1998.

[13] Y. Chevaleyre, P. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[14] O. Cliffe, M. de Vos, and J. Padget. Specifying and reasoning about multiple institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 67–85. Springer, 2007.

[15] P. R. Cohen and H. J. Levesque. Intention is choice with committment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[16] P. D'Altan, J-J. Ch Meyer, and R. J. Wieringa. An integrated framework for ought-to-be and ought-to-do constraints. *Artificial Intelligence and Law*, 4(2):77–111, 1996.

[17] M. Dastani, D. Grossi, J.-J. Ch. Meyer, and N. Tinnemeier. Normative multi-agent programs and their logics. In *Proc. Workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS)*, number 5605 in Lecture Notes in Computer Science, pages 16–31. Springer, 2009.

[18] K. Decker and V. Lesser. Designing a family of coordination algorithms. In *Proc. 1st International Conference on Multiagent Systems (ICMAS)*, pages 73–80, 1995.

[19] A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.

[20] M. Esteva, D. de la Cruz, and C. Sierra. ISLANDER: an electronic institutions editor. In *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1045–1052, 2002.

[21] Guido Governatori. Thou shalt is not you will. In *Proc. 15th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 63–68, New York, NY, USA, 2015. ACM.

[22] R. L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:169–231, 1979.

[23] D. Grossi, F. Dignum, L. M. M. Royakkers, and J-J. Ch. Meyer. Collective obligations and agents: Who gets the blame? In *Deontic Logic in Computer Science*, volume 3065 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2004.

[24] B. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[25] J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multi-agent systems using the $\mathcal{M}$OISE$^+$ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.

[26] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1):37–71, 2004.

[27] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.

[28] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):533–562, 1995.

[29] M. Ramírez, N. Yadav, and S. Sardiña. Behavior composition as fully observable non-deterministic planning. In *Proc. Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI, 2013.

[30] Ph. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic 4*, pages 393–436. King's College Publications, 2003.

[31] P. Senkul, M. Kifer, and I.H. Toroslu. A logical framework for scheduling workflows under resource allocation constraints. In *Proc. of 28th International Conference on Very Large Data Bases (VLDB)*, pages 694–705. Morgan Kaufmann, 2002.

[32] M. Sergot. Action and agency in norm-governed multi-agent systems. In *Proc. Engineering Societies in the Agents World, 8th International Workshop (ESAW)*, volume 4995 of *Lecture Notes in Computer Science*, pages 1–54. Springer, 2008.

[33] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proc. 10th National Conference on Artificial Intelligence*, 1992.

[34] M. Tambe and W. Zhang. Towards flexible teamwork in persistent teams: Extended report. *Autonomous Agents and Multi-Agent Systems*, 3(2):159–183, 2000.

[35] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2007.

[36] N. Yadav and S. Sardina. Decision theoretic behavior composition. In *Proc. 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 575–582. ACM Press, 2011.