Implementation of Normative Practical Reasoning with Durative Actions

Zohreh Shams[1(✉)], Marina De Vos[1], Julian Padget[1], and Wamberto Vasconcelos[2]

[1] Department of Computer Science, University of Bath, Bath, UK
{z.shams,m.d.vos,j.a.padget}@bath.ac.uk
[2] Department of Computing Science, University of Aberdeen, Aberdeen, UK
wvasconcelos@acm.org

Abstract. Autonomous agents operating in a dynamic environment need constantly to reason about actions in pursuit of their goals, while taking into consideration possible norms imposed on those actions. Normative practical reasoning supports agents decision making about what is best for an agent to do in a given situation. What makes practical reasoning challenging is the conflict between goals that the agent is pursuing and the norms that the agent is trying to uphold. We offer a formal model that allows the agents to plan for conflicting goals and norms in presence of *durative* actions that can be executed *concurrently*. We compare plans based on decision-theoretic notions (i.e. utility) such that the utility gain of goals and utility loss of norm violations are the basis of this comparison. The set of *optimal* plans consists of plans that maximise the overall utility, each of which can be chosen by the agent to execute. The formal model is implemented computationally using answer set programming, which in turns permits the statement of the problem in terms of a logic program that can be queried for solutions with specific properties. We demonstrate how a normative practical reasoning problem can be mapped into an answer set program such that the optimal plans of the former can be obtained as the answer sets of the latter.

1 Introduction

Norms define an ideal behaviour for an autonomous agent in an open environment. However, having individual goals to pursue, self-interested agents might not want or be able always to adhere to the norms imposed on them. Depending on the way they are given a computational interpretation, norms can be regarded as *soft* or *hard constraints*. When modelled as hard constrains, norms are regarded as *regimented*, in which case the agent has no choice but blindly to follow the norms [12]. Although regimentation guarantees norm compliance, it greatly restricts agent autonomy. Conversely, *enforcement* approaches in which norms are modelled as soft constraints, leave the choice of obeying or disobeying the norms to the agent. However, in order to encourage norm compliance, there are consequences introduced in terms of punishment in case the agent violates the norm [25,29]. Moreover, in some enforcement approaches [1] the agent

V. Dignum et al. (Eds.): COIN 2015, LNAI 9628, pp. 335–353, 2016.
DOI: 10.1007/978-3-319-42691-4_19

is rewarded for complying with a norm. The enforcement approaches can be broadly divided in two categories. In the utility-based approaches [1,2,26] there is a utility gain/loss associated with respecting norm or not, whereas in the *pressured norm compliance* approaches [25], violating a norm or not is determined by the interference of the norm in satisfying or hindering the agent goals. Gaining better utility or not losing utility is the basis of normative reasoning in the former category, while in the latter it is the potential conflicts between norms and agent goals. If there is no such conflict, the agent only complies with a norm if there are goals that are hindered by punishment of violation, and violates the norms otherwise. On the other hand, if there is a conflict, the agent does not comply unless the goals hindered by punishment are more important than goals facilitated by compliance.

Existing work on normative practical reasoning using enforcement have considered different phases of the practical reasoning process, such as plan generation and plan selection. In [27] norms are taken into account in the agent's plan generation phase, whereas [26] takes norms into consideration when deciding how to execute a pre-generated plan with respect to the norms triggered by that plan. There is also a substantial body of work on integration of norms into the BDI architecture [30]. The BOID architecture [7] extends BDI with the concept of obligation and uses agent types such as social, selfish, etc. to handle the conflicts between beliefs, desires, intentions and obligations. Another extended BDI architecture is proposed in [9], which focusses on norm recognition and considering them in agent decision making processes. More recently, [2] proposed a novel way of utilising permission norms in a BDI agent when the agent does not have complete information about the environment it operates in.

In this paper we define an approach for practical reasoning that considers norms in both plan generation and plan selection. We extend the current work on normative plan generation such that the agent attempts to satisfy a set of potentially conflicting goals in the presence of norms, as opposed to conventional planning problems that generate plans for a single goal [26,27]. Additionally, since in reality the actions are often non-atomic, the model allows for planning with durative actions that can be executed concurrently. Durative actions reflects the real time that a machine takes to execute certain actions, which is also known as "real-time duration" of actions [6]. More importantly, another contribution of this paper is introducing an enforcement approach that is a combination of utility-based and pressure-based compliance methods mentioned earlier. In order to do so, we first extend the notion of conflict defined in [25] by allowing conflict between norms as well as between norms and goals. We then define a penalty cost for norm violation, regardless of the existence of conflict. Whenever a norm is triggered, both outcomes of norm compliance and violation and their impacts on hindering or facilitating other goals and norms, are generated and compared according to their utility. Moreover, in those cases that there are no conflicts and no goals or norms hindered by the punishment of violation: loss of utility drives the agent toward compliance. Regarding plan selection, generated plans are compared based on the utility of the goals satisfied and cost of norms violated in the entire plan. Both plan generation and plan selection mechanisms proposed here are implemented using Answer Set Programming (ASP) [15].

ASP is a declarative programming paradigm using logic programs under the answer set semantics. In this paradigm the user provides a description of a problem and ASP works out how to solve the problem by returning answer sets corresponding to problem solutions. The existence of efficient solvers to generate the answers to the provided problems has increased the application of ASP in different domains of autonomous agents and multi-agent systems such as planning [24] and normative reasoning [8,28]. Several action languages (e.g. event calculus [21], $\mathcal{A}$ [14] and its descendants (e.g. $\mathcal{B}, \mathcal{C}$ [14]), Temporal Action Logics (TAL) [11]) have been implemented in ASP [22,23], which indicates that ASP is an appropriate tool for reasoning about actions. We therefore, propose an implementation of STRIPS [13] as an action language in ASP.

This paper is organised as follows. The formal model and its semantics are proposed in Sect. 2, followed by the computational implementation of the model in Sect. 3. Section 4 provides an example that illustrates the main features of the model in action. Finally, after the discussion of related work in Sect. 5, we conclude in Sect. 6.

2 A Model for Normative Practical Reasoning

This section introduces a formal model and its semantics for normative practical reasoning in the presence of durative actions. The foundation of this model is classical planning in which an agent is presented with a set of actions and a goal. Any sequence of actions that satisfies the goal is a solution to the planning problem. In Sect. 2.1 we extend the classical planning problem by substituting a single goal with a set of potentially inconsistent goals G and a set of norms N. A solution for such a problem is any sequence of actions that satisfies at least one goal. The agent has the choice of violating or complying with triggered norms, while satisfying its goals.

2.1 Syntax

A normative planning system is a tuple $P = (FL, \Delta, A, G, N)$ where FL is a set of fluents, Δ is the initial state, A is a set of durative STRIPS-like [13] actions, G denotes the set of agent goals and N denotes a set of norms imposed on the agent that define what an agent is obliged or forbidden to do under certain conditions. We now describe each of these in more details.

Fluents: FL is a set of domain fluents that accounts for the description of the domain the agent operates in. A literal l is a fluent or its negation i.e. $l = fl$ or $l = \neg fl$ for some $fl \in FL$. For a set of literals L, we define $L^+ = \{fl | fl \in L\}$ and $L^- = \{fl | \neg fl \in L\}$ to denote the set of positive and negative fluents in L respectively. L is well-defined if there exists no fluent $fl \in FL$ such that $fl \in L$ and $\neg fl \in L$, i.e. if $L^+ \cap L^- = \emptyset$.

The semantics of the model are defined over a set of states Σ. A state $s \subseteq FL$ is determined by set of fluents that hold *true* at a given time, while the other

fluents (those that are not present) are considered to be false. A state $s \in \Sigma$ satisfies fluent $fl \in FL$, denoted $s \models fl$, if $fl \in s$. It satisfies its negation $\neg fl$ if $fl \notin s$. This notation can be extended to a set of literals as follows, set X is satisfied in state s, $s \models X$, when $\forall x \in X \cdot s \models x$.

Initial State: The set of fluents that hold at the initial state is denoted by $\Delta \subseteq FL$.

Actions: A is a set of durative STRIPS-like actions, that is actions with preconditions and postconditions that take a non-zero duration of time to have their effects in terms of their postconditions. A durative action $a = \langle pr, ps, d \rangle$ is composed of well-defined sets of literals $pr(a), ps(a) \subseteq FL$ to represents a's preconditions and postconditions and a positive number $d(a) \in \mathbb{N}$ for its duration. Postconditions are further divided into a set of add postconditions $ps(a)^+$ and a set of delete postconditions $ps(a)^-$. An action a can be executed in a state s if its preconditions hold in s (i.e. $s \models pr(a)$. The postconditions of a durative action are applied in the state s at which the action ends (i.e. $s \models ps(a)^+$ and $s \not\models ps(a)^-$).

The model does not allow parallel actions, since it is not realistic to assume that a single agent initiates several actions at the exact same point in time. Concurrency however, is allowed unless there is a concurrency conflict between actions, which prevents them from being executed in an overlapping period of time. The definition of concurrency conflict is adopted from [4] as follows: two actions a_1 and a_2 are in a concurrency conflict, if the preconditions or postconditions of a_1 contradict the preconditions or postconditions of a_2.

Goals: G denotes a set of (possibly inconsistent) goals. Goals identify the state of affairs that an agent wants to satisfy. Each goal $g = \langle r, v \rangle$ is defined as a set of well-defined literals r, that are requirements that should hold in order to satisfy the goal and a positive integer $v \in \mathbb{N}$ that shows the value or utility gain of the agent upon satisfying this goal. Goal g's requirements and value are denoted $r(g)$ and $v(g)$, respectively. Goal g is satisfied in the state s when $s \models r(g)$.

Norms: N denotes a set of event-based norms to which the agent is subject. Each norm is a tuple of the form $n = \langle d\_o, a_1, a_2, dl, c \rangle$, where

- $d\_o \in \{o, f\}$ is the deontic operator determining the type of norm, which can be an obligation or a prohibition. The agent is assumed to be operating in a permissible society, hence what is not prohibited is permitted.
- $a_1 \in A$ is the action that counts as the norm activation condition.
- $a_2 \in A$ is the action that is the subject of the obligation or prohibition.
- $dl \in \mathbb{N}$ is the norm deadline relative to the activation condition, which is the completion of execution of a_1.
- $c \in \mathbb{N}$ is the penalty cost that will be applied if the norm is violated.

An obligation expresses that taking action a_1 obliges the agent to take action a_2 within dl time units of the end of execution of a_1. Such an obligation is complied

with if the agent starts executing a_2 before the deadline and is violated otherwise. A prohibition expresses that taking action a_1 prohibits the agent from taking action a_2 within dl time units of the end of execution of a_1. Such a prohibition is complied with if the agent does not start executing a_2 before the deadline and is violated otherwise.

2.2 Semantics

Let $P = (FL, \Delta, A, G, N)$ be a normative planning problem. A plan is represented by a sequence of actions taken at certain times, denoted as: $\pi = \langle (a_0, t_0), \cdots, (a_n, t_n) \rangle$. (a_i, t_i) means that action a_i is executed at time $t_i \in \mathbb{Z}^+$ s.t. $\forall i < j$ we have $t_i < t_j$. The total duration of a plan, $Makespan(\pi)$, is calculated by the relation: $Makespan(\pi) = max(t_i + d(a_i))$. The evolution of a sequence of actions for a given starting state $s_0 = \Delta$ is a sequence of states $\langle s_0, \cdots s_m \rangle$ for every discrete time interval from t_0 to m, where $m = Makespan(\pi)$. The transition relation between two states is defined by Eq. 1 below. If an action a_j ends at time t_i, state s_i results from removing all delete postconditions and adding all add postconditions of action a_j to state s_{i-1}. If there is no action ending at s_i, it remains the same as s_{i-1}.

$$\forall i > 0 : \; s_i = \begin{cases} (s_{i-1} \setminus ps(a_j)^-) \cup ps(a_j)^+ & i = t_j + d(a_j) \\ s_{i-1} & \text{otherwise} \end{cases} \tag{1}$$

A sequence of actions π satisfies a goal, $\pi \models g$, if there is at least one state s_i in the sequence of states caused by the sequence of actions such that $s_i \models g$. An obligation $n_1 = \langle o, a_i, a_j, dl, c \rangle$ is complied with in plan π (i.e. $\pi \models n_1$), if the action that is the norm activation condition has occurred ($(a_i, t_i) \in \pi$), and the action that is the subject of the obligation occurs ($(a_j, t_j) \in \pi$) between when the condition holds and when the deadline expires ($t_j \in [t_i + d(a_i), dl + t_i) + d(a_i)$). If a_i has occurred but a_j does not occur at all or occurs in a period other than the one specified, the obligation is violated (i.e. $\pi \not\models n_1$). In the case of prohibition $n_2 = \langle f, a_i, a_j, dl, c \rangle$, compliance happens if the action that is the norm activation condition has occurred ($(a_i, t_i) \in \pi$) and the action that is the subject of the prohibition does not occur in the period between when the condition holds and when the deadline expires ($\nexists (a_j, t_j) \in \pi$ s.t. $t_j \in [t_i + d(a_i), dl + t_i + d(a_i))$. If a_i has occurred and a_j occurs in the specified period, the prohibition is violated (i.e. $\pi \not\models n_2$). The set of satisfied goals, norms complied with and norms violated in plan π are denoted as G_π, $N_{cmp(\pi)}$ and $N_{vol(\pi)}$, respectively.

In classical planning, any sequence of actions that satisfies the goal is a solution to the planning problem. Extending a planning problem to cater for conflicting goals and norms requires considering different types of conflicts as follows:

Conflicting Actions. Actions a_i and a_j have a concurrency conflict iff the preconditions or postconditions of a_i contradict the preconditions or postconditions of a_j.

$$cf_{action} = \{(a_i, a_j) \text{ s.t. } \exists r \in pr(a_i) \cup ps(a_i), \neg r \in pr(a_j) \cup ps(a_j)\} \quad (2)$$

Conflicting Goals. Goal g_i and g_j are in conflict iff satisfying one requires bringing about a state of affairs that is in conflict with the state of affairs required for satisfying the other.

$$cf_{goal} = \{(g_i, g_j) \text{ s.t. } \exists r \in g_i, \neg r \in g_j\} \quad (3)$$

Conflicting Norms. Obligations $n_1 = \langle o, a_1, a_2, dl, c\rangle$ and $n_2 = \langle o, b_1, b_2, dl', c'\rangle$ are in conflict in the context of plan π iff: (i) their activation conditions hold, (ii) the obliged actions a_2 and b_2 have a concurrency conflict and (iii) a_2 is in progress during the entire period over which the agent is obliged to take action b_2. The set of conflicting obligations is formulated as:

$$\begin{aligned} cf^{\pi}_{oblobl} = \{(n_1, n_2) \text{ s.t. } & (a_1, t_{a_1}), (b_1, t_{b_1}) \in \pi; (a_2, b_2) \in cf_{action}; \\ & t_{a_2} \in [t_{a_1} + d(a_1), t_{a_1} + d(a_1) + dl); \\ & [t_{b_1} + d(b_1), t_{b_1} + d(b_1) + dl') \subseteq [t_{a_2}, t_{a_2} + d(a_2))\} \end{aligned} \quad (4)$$

On the other hand, an obligation $n_1 = \langle o, a_1, a_2, dl, c\rangle$ and a prohibition $n_2 = \langle f, b_1, a_2, dl', c'\rangle$ are in conflict in the context of plan π iff: (i) their activation conditions hold and (ii) n_2 forbids the agent to take action a_2 during the entire period over which n_1 obliges the agent to take a_2. The set cf^{π}_{oblpro} denotes the set of conflicting obligations and prohibitions as below:

$$\begin{aligned} cf^{\pi}_{oblpro} = \{(n_1, n_2) \text{ s.t. } & (a_1, t_{a_1}), (b_2, t_{b_2}) \in \pi; \\ & [t_{a_1} + d(a_1), t_{a_1} + d(a_1) + dl) \subseteq \\ & [t_{b_2} + d(b_2), t_{b_2} + d(b_2) + dl')\} \end{aligned} \quad (5)$$

The entire set of conflicting goals and norms is defined as:

$$cf^{\pi}_{norm} = cf^{\pi}_{oblobl} \cup cf^{\pi}_{oblpro} \quad (6)$$

Conflicting Goals and Norms. An obligation $n = \langle o, a_1, a_2, dl, c\rangle$ and a goal g are in conflict, if taking action a_2 that is the subject of the obligation, brings about postconditions that are in conflict with the requirements of goal g. The set of conflicting goals and obligations is formulated as:

$$cf_{goalobl} = \{(g, n) \text{ s.t. } \exists r \in r(g), \neg r \in ps(a_2)\} \quad (7)$$

In addition, a prohibition $n = \langle f, a_1, a_2, dl, c'\rangle$ and a goal g are in conflict, if the postconditions of a_2 contribute to satisfying g, but taking action a_2 is prohibited by norm n.

$$cf_{goalpro} = \{(g, n) \text{ s.t. } \exists r \in r(g), r \in ps(a_2)\} \quad (8)$$

The entire set of conflicting goals and norms is defined as:

$$cf_{goalnorm} = cf_{goalobl} \cup cf_{goalpro} \quad (9)$$

A sequence of actions π is a plan for P, if all the fluents in Δ hold at time t_0 and for each i, the preconditions of action a_i hold at time t_i, as well as through the execution of a_i, and a non-empty subset of goals is satisfied in the path from initial state s_0 to the state holding at time t_m, where $m = Makespan(\pi)$. Furthermore, extending the conventional planning problem by multiple potentially conflicting goals and norms requires defining extra conditions that makes a plan a valid plan and a solution for P. Plan π is a valid plan for P iff:

1. all the fluents and only those fluents in Δ hold in the initial state: $s_0 = \Delta$
2. the preconditions of action a_1 holds at time t_{a_1} and throughout the execution of a_1:
$$\forall k \in [t_{a_1}, t_{a_1} + d(a_1)), s_k \models pr(a_1)$$
3. the set of goals satisfied by plan π is a non-empty consistent subset of goals:
$$G_\pi \subseteq G \text{ and } G_\pi \neq \emptyset \text{ and } \nexists g_i, g_j \in G_\pi \text{ s.t. } (g_i, g_j) \in cf_{goal}$$
4. there is no concurrency conflict between actions that are executed concurrently:
$$\nexists (a_i, t_{a_i}), (a_j, t_{a_j}) \in \pi \text{ s.t. } t_{a_i} \leq t_{a_j} < t_{a_i} + d(a_i), (a_i, a_j) \in cf_{action}$$
5. there is no conflict between norms complied with.
$$\nexists n_i, n_j \in N_{cmp(\pi)} \text{ s.t. } (n_i, n_j) \in cf^{\pi}_{norm}$$
6. there is no conflict between goals satisfied and norms complied with:
$$\nexists g \in G_\pi \text{ and } n \in N_{cmp(\pi)} \text{ s.t. } (g, n) \in cf_{goalnorm}$$

Let $satisfied(\pi)$ and $violated(\pi)$ be the set of satisfied goals and violated norms in plan π. The utility of a plan π is defined by Eq. 10 where *Value* is a function that returns the value of goals being satisfied and *Cost* returns the penalty cost of norms being violated in that plan. The set of optimal plans, *Opt*, are those plans that maximise the utility.

$$Utility(\pi) = \sum_{g_i \in satisfied(\pi)} Value(g_i) - \sum_{n_j \in violated(\pi)} Cost(n_j) \tag{10}$$

3 An Answer Set Programming Implementation

Encoding a practical reasoning problem as a declarative specification makes it possible to reason computationally about agent actions, goals and norms. This enables an agent to keep track of actions taken, goals satisfied and norms complied with or violated at each state of its evolution. More importantly, it provides the possibility of querying traces that fulfil certain requirements such as satisfying some specific goals. Consequently, instead of generating all possible

traces and looking for those ones that satisfy at least one goal, only those ones that *do* satisfy at least one goal are generated.

ASP programs consist of a finite set of rules formed from atoms. Atoms are the basic components of the language that can be assigned a truth value (*true* or *false*). Literals are atoms or negated atoms. Atoms are negated using classical negation ($\neg$) or *negation as failure* (*not*). The former states that something is false, whereas the latter states something is assumed false since it cannot be proven true. The general rule syntax in ASP is: $l_0 \leftarrow l_1, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n.$, in which l_i is an atom (e.g. a) or its negation (e.g. $\neg a$). l_0 is the rule head and $l_1, \cdots, l_m$, $not\ l_{m+1}, \cdots, not\ l_n$ are the body of the rule. The above rule is read as: l_0 is known/true, if $l_1, \cdots, l_m$ are known/true and none of l_{m+1}, l_n are known. If a rule body is empty, that rule is called a fact and if the head is empty, it is called a constraint indicating that none of the answers should satisfy the body.

3.1 Translating the Model into ASP

In this section, we demonstrate how a planning problem $P = (FL, \Delta, A, G, N)$ can be mapped into an answer set program such that there is a one to one correspondence between solutions for the planning problem and the answers of the program. The mapping uses the following atoms: `state`(s) for denoting the states; `time`(t, s) to indicate the time at state s; `holdsat`(x, s) to express fluent x is true in state s; `occurred(a,s)` to encode action a occurs at state s. There are additional atoms used in Figs. 2, 3, 4, 5 and 6, that will be discussed in their respective sections. Please note that the variables begin with capital letters in ASP.

Time and Initial State (Fig. 1). The facts produced by Line 1 provide the program with all available states, while Line 2 defines the order of states. The maximum number of states, q, results from sum of duration of all actions: $q = \sum_{i=1}^{n} d(a_i)$. The final state is therefore stated as sq in Line 3. Line 4 illustrates the initial time that increases by one unit from one state to the state next to it (Line 5). Finally, Line 6 encodes the fluents that hold at initial state $s0$.

Actions (Fig. 2). Each durative action is encoded as `action`(a, d) (Line 7), where a is the name of the action and d is its duration. Recalling from Sect. 2, the preconditions $pr(a)$ of action a hold in state s if $s \models pr(a)$. This is expressed in Line 8, where $pr(a)^+$ and $pr(a)^-$ are positive and negative literals in $pr(a)$. In order to make the coding more readable we introduce the shorthand `EX(X,S)` where `X` is a set of fluents that should hold at state `S`. For all $x \in$ `X`, `EX(X,S)` is translated into `holdsat(x,S)` and for all $\neg x \in$ `X`, `EX(`$\neg$`X,S)` is translated into `not EX(x,S)` using negation as failure. The agent has the choice to take any of its actions in any state (Line 9), however, the preconditions of a durative action should be preserved when it is in progress. A durative action is in progress, `inprog`(A, S), from the state in which it begins to the state in which it ends at (Lines 10 to 11). Then, Line 12 rules out the execution of an action, when

$\forall\, k \in [0, q]$

```
1  state(sk) .
```

$\forall\, k \in [0, q-1]$

```
2  next(sk, s(k+1)) .

3  final(sq) .

4  time(t, s0) .
5  time(T+1,S2) :- time(T,S1), next(S1,S2), state(S1;S2).
```

$\forall\, x \in \Delta$

```
6  holdsat(x, s0) .
```

Fig. 1. Rules for time component (Lines 1–5) and initial state (Line 6)

the preconditions of the action do not hold during its execution. In addition there should not be any action in progress in the final state (Line 13). Another assumption made in Sect. 2, is the prevention of parallel actions, which prevents the agent from starting two actions at the same time (Lines 14 to 15). Once an action starts in one state, the result of its execution is reflected in the state where the action ends. This is expressed through (i) Lines 16 to 17 that allow the add postconditions of the action to hold when the action ends, and (ii) Lines 18 to 19 that allow the termination of the delete postconditions. The termination happens in the state before the end state of the action. The reason for this is that all the fluents that hold in a state, hold in the next state unless they are terminated (Lines 20 to 21). Since the delete postconditions of an action are terminated in the state before the end state of the action, they will not hold in the following state, in which the action ends (i.e. they are deleted from the state).

Goals (Fig. 3). Line 22 encodes goal g with value of v. From Sect. 2, we have goal g is satisfied in state s if $s \models r(g)$. This is expressed in Line 23, where $r(g)^+$ and $r(g)^-$ are the positive and negative literals in $r(g)$.

Norms (Fig. 4). The conditional event-based norms that are the focus of this research are discussed in the previous section. Line 24 encodes norm n with penalty cost of c upon violation. Lines 25–39 deal with obligations and prohibitions of form: $n = \langle d\_o, a_1, a_2, dl, c\rangle$. In order to implement the concepts of norm compliance and violation described in Sect. 2.2, we introduce normative fluents $o(n, a_2, dl')$ and $f(n, a_2, dl')$ that first hold in the state in which action a_1's execution ends. An obligation fluent $o(n, a_2, dl')$ denotes that action a_2 should be brought about before deadline dl'or be subject to violation, whereas prohibition fluent $f(n, a_2, dl')$ denotes that action a_2 should not be brought about before deadline dl' or be subject to violation. If a_1 with duration $d1$ occurs at state S, where time is T, the agent has dl units time starting from end of action

$\forall a \in A$ s.t. $d(a)$

```
7  action(a,d).
8  pre(a,S) :- EX(pr(a)+,S), not EX(pr(a)-,S), state(S).
```

```
9  {occurred(A,S)} :- action(A,D), state(S).
10 inprog(A,S2) :- occurred(A,S1), action(A,D), time(T1,S1),
11                 time(T2,S2), state(S1;S2), T1<=T2, T2<T1+D.
12 :- inprog(Act,S), action(Act,D), state(S), not pre(Act,S).
13 :- inprog(Act,S), action(Act,D), state(S), final(S).
14 :- occurred(A1,S), occurred(A2,S), A1!=A2,
15    action(A1,D1), action(A2,D2), state(S).
```

$ps(a)^+ = X \Leftrightarrow \forall x \in X \cdot$

```
16 holdsat(x,S2) :- occurred(a,S1), action(a,d), state(S1;S2),
17                  time(T1,S1), time(T2,S2), T2=T1+d.
```

$ps(a)^- = X \Leftrightarrow \forall x \in X \cdot$

```
18 terminated(x,S2) :- occurred(a,S1), action(a,d), state(S1;S2),
19                     time(T1,S1), time(T2,S2), T2=T1+d-1.
```

```
20 holdsat(X,S2) :- holdsat(X,S1), not terminated(X,S1),
21                  next(S1,S2),state(S1;S2).
```

Fig. 2. Rules for translating actions

$\forall g \in G$

```
22 goal(g,v).
23 satisfied(g,S) :- EX(r(g)+,S), not EX(r(g)-,S), state(S).
```

Fig. 3. Rules for translating goals

a_1 (`T2=T1+`$d1$) to comply with the norm imposed on it. Lines 25–26 and 32–33 indicate the establishment of obligation and prohibition fluents.

In terms of compliance and violation, the occurrence of an obliged action before the deadline expires, counts as compliance (Lines 27 to 28) and the absence of such an occurrence before the deadline is regarded as violation (Line 30). Atoms `cmp(o|f(`$n, a,$ `DL),S)` and `vol(o|f(`$n, a,$ `DL),S)` are used to indicate compliance or violation of norm n in state `S`. In both cases of compliance and violation, the norm is terminated (Lines 29 and 31). On the other hand, a prohibition is complied with if the forbidden action does not happen before the deadline (Lines 34 to 35) and is violated if it does happen before the deadline (Lines 37 to 38). As with obligations, after being complied with or violated, the prohibitions are terminated (rules 36 and 39).

3.2 Mapping of Answer Sets to Plans

In Sect. 2.2 we defined the criteria for a sequence of actions to be identified as a valid plan and solution for $P = \langle FL, \Delta, A, G, N \rangle$. Figure 5 provides the coding

for the criteria. The rule in Line 41 is responsible for constraining answer sets to those that fulfil at least one goal by excluding answers that do not satisfy any goals. The input for this rule is provided in Line 40. Line 42 prevents satisfying two conflicting goals, hence guaranteeing the consistency of satisfied goals in a plan. Preventing the concurrency of conflicting actions, is implemented using Lines 43–44, by expressing that such two actions cannot be in progress together. Lines 45 and 46 provides the input for Lines 47 and 48, which exclude the possibility of satisfying a goal and complying with a norm that are conflicting. Note that the implementation prevents complying with conflicting norms automatically: (i) since it is not possible to execute two conflicting actions concurrently, if two obligations would require that, one of them has to be violated, while (ii) regarding conflicting obligation and prohibition, by definition, taking the obliged action by the agent and hence complying with the obligation causes the violation of the other norm that enforces the prohibition of taking the very same action, and vice versa.

Theorem 1. *Let program Π_{base} consist of Lines 7 – 48. Given a planning problem $P = (FL, \Delta, A, G, N)$, for every answer set Ans of Π_{base} the set of atoms of the form* `occurred(a,s)`[1] *in Ans encodes a solution to the planning problem P. Conversely, each solution to the problem P corresponds to a single answer set of Π_{base}.*

$\forall n = \langle o | f, a_1, a_2, dl, c \rangle \in N$

```
24 norm(n,c).

25 holdsat(o(n,a2,dl+T2),S) :- occurred(a1,S1), action(a1,d1),
26                 time(T1,S1), T2=T1+d1, time(T2,S2), state(S1;S2).
27 cmp(o(n,a,DL),S) :- holdsat(o(n,a,DL),S),occurred(a,S),action(a,d)
28                   state(S), time(T,S), T!=DL.
29 terminated(o(n,a,DL),S) :- cmp(o(n,a,DL),S), state(S).
30 vol(o(n,a,DL),S) :- holdsat(o(n,a,DL),S), time(DL,S), state(S).
31 terminated(o(n,a,DL),S) :- vol(o(n,a,DL),S), state(S).
32 holdsat(f(n,a2,dl+T2),S) :- occurred(a1,S), action(a1,d1),
33                 time(T1,S1), T2=T1+d1, time(T2,S2), state(S1;S2).
34 cmp(f(n,a,DL),S) :- holdsat(f(n,a,DL),S), action(a,d),
35                    time(DL,S), state(S).
36 terminated(f(n,a,DL),S) :- cmp(f(n,a,DL),S), state(S).
37 vol(f(n,a,DL),S) :- holdsat(f(n,a,DL),S), occurred(a,S),
38                    state(S), time(T,S), T!=DL.
39 terminated(f(n,a,DL),S) :- vol(f(n,a,DL),S), state(S).
```

Fig. 4. Rules for translating norms

[1] In the formal model a plan/solution π for problem P is defined as a set of action, time pairs (e.g. (a_i, t_i)), whereas in the answer sets a plan is expressed by action, state pairs (e.g. `occurred(a,s)`). Action, state pairs can easily be mapped to action, time pairs by replacing the state with the time that holds in that state.

```
40 satisfied(g) :- satisfied(g,S), state(S).
41 :- not  satisfied(g1), ... , not satisfied(gm).
```

$\forall\ (g_1, g_2) \in cf_{goal}$

```
42 :- satisfied(g1),satisfied(g2).
```

$\forall\ (a_1, a_2) \in cf_{action}$

```
43 :- inprog(a1,S), inprog(a2,S), action(a1,d1),
44    action(a2,d2), state(S).
```

```
45 complied(n) :- cmp(o(n,a,DL),S), state(S).
46 complied(n) :- cmp(f(n,a,DL),S), state(S).
```

$\forall(g, n_1) \in cf_{goalobl}$

```
47 :- satisfied(g), complied(n1).
```

$\forall(g, n_2) \in cf_{goalpro}$

```
48 :- satisfied(g), complied(n2).
```

Fig. 5. Solutions for problem P

Proof (sketch). The proof can be obtained through structural induction. Line 9 generates all sequences of actions. Line 6 ensures that all fluents in Δ hold at t_0. Line 12 guarantees that the precondition of an action hold all through its execution. Line 41 indicates that a non-empty subset of goals has to be satisfied in a plan, while Line 42 ensures the consistency of the goals satisfied. Preventing the concurrency conflict is provided in Lines 43–44. Finally, Lines 47–48 eliminate the possibility of conflict between goals satisfied and norms complied with. This implies that the sequence of actions that is part of the answer set satisfies the conditions to be a solution to the encoded planning program. Conversely, each solution satisfies all the program's rules in a minimal fashion.

3.3 Optimised Plans

In order to find optimal plans, in Fig. 6 we show how to encode the utility function defined by Eq. 10. The sum of values of goals satisfied in a plan is calculated in Line 49. The sum of costs of norms violated in a plan is calculated in Line 49, by first providing the input for this line in Lines 50 and 51. Having calculated `value(TV)` and `cost(TC)`, the utility of a plan is denoted in Line 53, which is subject to the optimisation statement in the final line.

Theorem 2. *Let program $\Pi = \Pi_{base} \cup \Pi^*$, where Π^* consists of Lines 49 – 54. Given a planning problem $P = (FL, \Delta, A, G, N)$, for every answer set Ans of Π the set of atoms of the form* `occurred(a,s)` *in Ans encodes an optimal solution to the planning problem P. Conversely, each optimal solution for the problem P corresponds to a single answer set of Π.*

```
49 value(TV) :- TV = #sum [satisfied(G) : goal(G,V) = V].
50 violated(n) :- vol(o(n,a,DL), state(S).
51 violated(n) :- vol(f(n,a,DL),S), state(S).
52 cost(TC) :- TC = #sum [violated(N) : norm(N,C) = C].
53 utility(TV-TC) :- value(TV), cost(TC).
54 #maximize [Utility(U)=U].
```

Fig. 6. Optimised solutions for P

Proof (Sketch). Theorem 1 ensures that all solutions are represented by answer sets and vice versa. The optimality of solutions is guaranteed in this program. Line 54 ensures optimal solutions that maximise utility, which is in turn defined in Line 53 as the difference between the cost of violation (Line 52) and goal values (Line 49).

4 Illustrative Example

In this section, we provide a brief example that highlights the most important features of the proposed model. Let us consider an agent with the durative actions presented in Table 1. The agent has three goals presented with their requirements and two different set of values in Table 2. The first goal is to get some $certificate$ that requires the agent to take some test, but in order to be able to attend the test, the agent first needs to pay the fee for the test. The second goal is to make a *submission* of some marking that needs to be done in the *office* and the last goal is to go on *strike*, for which the agent needs to be a member of union, not to go to *office* nor to attend any meeting on behalf of the company. In addition, one of the agent's action, $comp\_funding$, has a normative consequence captured in a norm that states that if company funds are used to pay the fee for the test, the agent is obliged to attend a meeting on behalf of the company within 1 time unit of end of action $comp\_funding$, which results in the payment of the fee for the test. If the agent uses the funding, but does not attend the meeting before the deadline, it is entitled to the penalty cost of 4 units.

$$n = \langle o, comp\_funding, attend\_meeting, 1, 4 \rangle$$

Table 3 shows the corresponding ASP code for this example based on the code in Sect. 3. For spacial reasons, only those rules that need instantiation are provided. For ease of reference, rules instantiated in each part of the code are titled by their corresponding figures in Sect. 3. Moreover, only one action, *drive*, and one goal, $certificate$, are encoded. The rest of the actions and goals can be coded in the same way.

Following Theorem 2, we obtain a one-to-one correspondence between the answer sets of the program in Table 3 and optimal plans for the agent to execute such that the agent utility is maximised. Table 4 illustrates the optimal plans (as translations of the answer sets) based on two different set of values in

Table 2. Plan π_1 satisfies goals *certificate* and *strike*, however due to the conflict between *strike* and norm n, the norm is inevitably violated. Additionally, the conflict between goal *strike* and *submission*, makes it impossible for the agent to satisfy *submission*. Since the sum of utility loss of violating n and not satisfying *submission*, is still less that the utility gain of satisfying *strike*, the agent prefers the former to the latter. On the other hand, in plan π_2 satisfying *submission* is preferred over satisfying *strike*, although they have the same utility gain. However, satisfying *strike* would have implied violating n, and thus incurring the penalty cost of 4. Therefore, in pursuit of maximising the utility, the agent prefers satisfying *submission* and complying with n to satisfying *strike* and violating n, which was the case in plan π_1.

5 Related Work

The interaction between an agent's individual goals and social norms has been discussed in a number of works. Some such as [26,27] use utility measurement to enforce norm compliance. In contrast, in [25] norm compliance relies on the explicit interaction between goals and norms, but if the norm compliance or violation does not hinder any goals there is no connection and hence no computational mechanism in place that enforces the norms. From a planning perspective, norms are taken into account in plan generation [27] and in plan selection [20,26]. In [27] the normative state of the agent is checked by a planner after each individual action is taken, which depending on the number of actions, imposes a high computational cost on the step-by-step generation of plans. It is the utility of individual actions here that determines norm compliance. On the other hand, [20,26] consider norms as part of plan selection, starting from the assumption

Table 1. Agent Actions

Preconditions	(Action, Duration)	Postconditions
$\neg office$	$(drive, 1)$	$office$
$\neg marking\_done, office$	$(marking, 2)$	$marking\_done$
$\neg test\_done, fee\_paid$	$(attend\_test, 1)$	$test\_done$
$\neg fee\_paid$	$(comp\_funding, 1)$	fee_paid
$\neg meeting\_attended, fee\_paid$	$(attend\_meeting, 2)$	$meeting\_attended$
$\neg union\_member$	$(join\_union, 1)$	$union\_member$

Table 2. Agent Goals

Goals	Requirements	$Value_1$	$Value_2$
$certificate$	$fee\_paid, test\_done$	8	5
$submission$	$office, marking\_done$	3	7
$strike$	$union\_member, \neg office, \neg meeting$	9	7

Table 3. Instantiated ASP code

Fig. 1.

```
state(s0;s1;s2;s3;s4;s5;s6;s7;s8).
next(s0,s1). next(s1,s2). next(s2,s3). next(s3,s4).
next(s4,s5). next(s5,s6). next(s6,s7). next(s7,s8).
final(s8).
time(0,s0).
```

Fig. 2.

```
action(drive,1).
pre(drive,S) :- not holdsat(office,S), state(S).
holdsat(office,S2) :- occurred(drive,S1),action(drive,1),time(T1,S1),
                      time(T2,S2), state(S1;S2), T2=T1+1.
```

Fig. 3.

```
goal(certificate,8).
satisfied(certificate,S) :- holdsat(fee_paid,S), holdsat(test_done,S),
                            state(S).
```

Fig. 4.

```
norm(n,4).
holdsat(o(n,attend_meeting,1+T2),S2) :- occurred(comp_funding,S1),
                        time(T1,S1),action(comp_funding,1),T2=T1+1,
                                    time(T2,S2),state(S1;S2).
cmp(o(n,attend_meeting,DL),S) :- holdsat(o(n,attend_meeting,DL),S),
             occurred(attend_meeting,S),action(attend_meeting,2),
                                  state(S),time(T,S), T != DL.
terminated(o(n,attend_meeting,DL),S) :- cmp(o(n,attend_meeting,DL),S),
                                        state(S).
vol(o(n,attend_meeting,DL),S) :- holdsat(o(n,attend_meeting,DL),S),
                    time(DL,S),action(attend_meeting,2), state(S).
terminated(o(n,attend_meeting,DL),S) :- vol(o(n,attend_meeting,DL),S),
                                        state(S).
```

Fig. 5.

```
satisfied(certificate) :- satisfied(certificate,S), state(S).
:- not satisfied(submission), not satisfied(certificate),
   not satisfied(strike).
:- satisfied(strike), satisfied(submission).
:- inprog(comp_funding,S), inprog(attend_test,S), state(S)
   action(comp_funding,1), action(attend_test,1).
:- inprog(comp_funding,S), inprog(attend_meeting,S), state(S),
   action(comp_funding,1), action(attend_meeting,2).
complied(n) :- cmp(o(n,attend_meeting,DL),S), state(S).
:- satisfied(strike),complied(n).
```

Fig. 6.

```
violated(n) :- vol(o(n,attend_meeting,DL),S), state(S).
```

that the agent has access to a library of pre-generated plans. In contrast to all of [20,26,27], our work deals with both plan generation and plan selection while taking account of norms, and like [26] we focus on the utility of the entire plan, unlike [27] which only considers the constituent actions in sequence.

Table 4. Optimal plans

Goal Values	Plans	Goals	Norms	*Utility*
$Value_1$	$\pi_1 = \langle(comp\_funding, 0), (union, 1), (attend\_test, 2)\rangle$	$certificate, strike$	$violated(n)$	13
$Value_2$	$\pi_2 = \langle(drive, 0), (marking, 1), (comp\_funding, 2), (attend\_meeting, 3), (attend\_test, 4)\rangle$	$submission, certificate$	$complied(n)$	12

Some works [19,32] focus on interaction between an agent's goal and its commitments, where commitments are made by agents to one another in order to support the realisation of their goals. Our approach is different from these approaches for two main reasons: (i) commitments are deliberately made by the agent, whereas norms are externally imposed to the agent; and (ii) commitments are made to support satisfying goals, while imposed norms might be in conflict with the agent's goals and consequently, hinder some of them.

The Event Calculus (EC) [21] forms the basis for the implementation of some normative reasoning frameworks, such as [2,3]. Our proposed formal model is independent of language and could be translated to EC and hence to a computational model, but the one-step translation to ASP is preferred because the formulation of the problem is very similar to the computational model, thus there are no conceptual gaps to bridge. Furthermore, the EC implementation language is Prolog, which although syntactically similar to ASP, suffers from non-declarative functionality in the form of the cut operator, which results in a loss of completeness. Furthermore, its query-based nature that focusses on one issue at a time, makes it cumbersome to reason about all plans.

A final point is that the norm representation and implementation proposed here is expressive and realistic in respect of time and duration: specifically, since the formal model and ASP implementation handle time explicitly, it is straightforward to represent the norm deadline as a future time instant, rather than a state to be brought about.

6 Conclusions, Discussion and Future Work

An agent performing practical reasoning in an environment regulated by norms, needs constantly to weigh up the importance of goals satisfied and norms complied with against goals not satisfied and norms broken. This comparison is possible when the agent has access to all possible plans, such that the decision of which goals to pursue and which norms to respect is made based on their impact on the entire plan. We show how this impact can be captured in a utility function that permits the agent to execute a plan that maximises the utility.

The focus of plan selection in this paper is on maximising the agent utility by considering the value of goals and penalties for norm violation. While these are sensible criteria, there are others that can be taken into account. Given that actions modelled in this approach are durative, one such criterion is the duration of the entire plan. Since durative actions that do not have concurrency conflicts can be executed concurrently, there might exist some plans with the exact the same utility while one takes longer than another. We intend to extend the plan selection mechanism with additional criteria by using the existing multi-criteria optimisation mechanisms in ASP.

Just like norms, in real scenarios, goals often have a deadline before which they should be satisfied [18]). Temporally extended goals [17] are discussed in detail in agent programming languages such as GOAL [5], however they are not commonly used in practical reasoning frameworks. Substituting achievement goals with temporally extended goals increases the expressiveness of the model. It also allows defining conflict within goals and between goals and norms temporally and which results in enriching the concept of conflict in the model.

Incorporation plan revision is also an avenue for future work. As presented here, a plan once selected is acted out until its conclusion, but it is of course necessary to incorporate plan revision in order to handle the inevitable dynamic environment.

Another area of improvement is to extend the normative reasoning capability of the model by extending it for state based norms in addition to event-based norms. Such an extension would allow the expression of obligations and prohibitions to achieve or avoid some state before some deadline. A combination of event and state based norms [10] enriches the norm representation as well as normative reasoning.

Lastly, we intend to build on the current ASP implementation to provide justification for why a certain plan maximises the utility considering the goals and norms it satisfies against those it does not. A potential starting point is [31], where it is possible to explain why certain literals are part of an answer set of a program and why others are not.

References

1. Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Operationalisation of norms for usagein electronic institutions. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8–12, pp. 223–225. ACM (2006)
2. Alrawagfeh, W., Meneguzzi, F.: Utilizing permission norms in BDI practical normative reasoning. In: Ghose, A., et al. (eds.) COIN 2014. LNCS, vol. 9372, pp. 1–18. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25420-3_1
3. Artikis, A., Sergot, M.J., Pitt, J.V.: Specifying norm-governed computational societies. ACM Trans. Comput. Log. **10**(1), 1–42 (2009)
4. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artif. Intell. **90**(1), 281–300 (1997)

5. de Boer, F.S., Hindriks, K.V., van der Hoek, W., Meyer, J.-J.C.: A verification framework for agent programming with declarative goals. J. Appl. Logic **5**(2), 277–302 (2007)
6. Börger, E., Stärk, R.: Asynchronous multi-agent ASMs. In: Börger, E., Stärk, R. (eds.) Abstract State Machines, pp. 207–282. Springer, Heidelberg (2003)
7. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In: Proceedings of the Fifth International Conference on Autonomous Agents. AGENTS 2001, pp. 9–16. ACM, Montreal (2001)
8. Cliffe, O., De Vos, M., Padget, J.: Answer set programming for representing and reasoning about virtual institutions. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA 2006. LNCS (LNAI), vol. 4371, pp. 60–79. Springer, Heidelberg (2007)
9. Criado, N., Argente, E., Botti, V.J.: A BDI architecture for normative decision making. In: van der Hoek, W., Kaminka, G.A., Lespérance, Y., Luck, M., Sen, S. (eds.) 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10–14, vol. 1–3. IFAAMAS, pp. 1383–1384 (2010)
10. De Vos, M., Balke, T., Satoh, K.: Combining event-and state-based norms. In: Gini, M.L. Shehory, O., Ito, T., Jonker, C.M. (eds.) International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013, Saint Paul, MN, USA, May 6–10, IFAAMAS, pp. 1157–1158 (2013)
11. Doherty, P., Gustafsson, J., Karlsson, L., Kvarnström, J.: TAL: temporal action logics language specification and tutorial. Electron. Trans. Artif. Intell. **2**, 273–306 (1998)
12. Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., Arcos, J.-L.: On the formal specification of electronic institutions. In: Sierra, C., Dignum, F.P.M. (eds.) AgentLink 2000. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
13. Fikes, R.E., Nilsson, N.J.: STRIPS: a new approach to the application of theorem proving to problem solving. In: Proceedings of the 2Nd International Joint Conference on Artificial Intelligence. IJCAI 1971, pp. 608–620. Morgan Kaufmann Publishers Inc., San Francisco (1971)
14. Gelfond, M., Lifschitz, V.: Action languages. Electron. Trans. AI **3**, 281–300 (1998)
15. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) ICLP, SLP, pp. 1070–1080. MIT Press (1988)
16. Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.): International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013, Saint Paul, MN, USA, May 6–10, 2013. IFAAMAS (2013)
17. Hindriks, K.V., van der Hoek, W., van Riemsdijk, M.B.: Agent programming with temporally extended goals. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10–15, 2009, vol. 1. IFAAMAS, pp. 137–144 (2009)
18. Hindriks, K.V., van Riemsdijk, M.B.: Satisfying maintenance goals. In: Baldoni, M., Son, T.C., Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2007. LNCS (LNAI), vol. 4897, pp. 86–103. Springer, Heidelberg (2008)
19. Kafali, Ö., Günay, A., Yolum, P.: GOSU: computing Goal Support with commitments in multiagent systems. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) Frontiers in Artificial Intelligence and Applications ECAI 2014–21st European Conference on Artificial Intelligence, 18–22 , Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014), vol. 263, pp. 477–482. IOS Press (2014)

20. Kollingbaum, M.: Norm-governed Practical Reasonig Agents. Ph.D. thesis. University of Aberdeen (2005)
21. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. **4**(1), 67–95 (1986)
22. Lee, J., Palla, R.: Reformulating temporal action logicsin answer set programming. In: Hoffmann, J., Selman, B. (eds.) Proceedings of the Twenty-SixthAAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada. AAAI Press (2012)
23. Lee, J., Palla, R.: Reformulating the situation calculus and theevent calculus in the general theory of stable models and in answer set programming. CoRR abs/1401.4607 (2014)
24. Lifschitz, V.: Answer set programming and plan generation. Artif. Intell. **138**(1–2), 39–54 (2002)
25. y López, F.L., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. In: Normative Multi-Agent Systems (NORMAS), pp. 24–35 (2005)
26. Oren, N., Vasconcelos, W., Meneguzzi, F., Luck, M.: Acting on norm constrained plans. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 347–363. Springer, Heidelberg (2011)
27. Panagiotidi, S., Vázquez-Salceda, J., Dignum, F.: Reasoning over norm compliance via planning. In: Aldewereld, H., Sichman, J.S. (eds.) COIN 2012. LNCS, vol. 7756, pp. 35–52. Springer, Heidelberg (2013)
28. Panagiotidi, S., Vázquez-Salceda, J., Vasconcelos, W.: Contextual norm-based plan evaluation via answer set programming. In: BajoPérez, J., et al. (eds.) Highlights on Practical Applications of Agentsand Multi-Agent Systems. AISC, vol. 156, pp. 197–206. Springer, Heidelberg (2012)
29. Pitt, J., Busquets, D., Riveret, R.: Formal models of social processes: the pursuit of computational justice in self-organising multi-agent systems. In: 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 269–270 (2013)
30. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Proceedings of The First International Conference on Multi-Agent Systems (ICMAS 1995), pp. 312–319 (1995)
31. Schulz, C., Toni, F.: Justifying Answer Sets using Argumentation. CoRR abs/1411.5635 (2014)
32. Telang, P.R., Meneguzzi, F., Singh, M.P.: Hierarchical planning about goals and commitments. In: Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.) International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013, Saint Paul, MN, USA, May 6–10, 2013. IFAAMAS, pp. 877–884 (2013)