Too Cool for School — Adding Social Constraints in Human Aware Planning [1]

Stevan Tomic and **Federico Pecora** and **Alessandro Saffiotti** [2]

Abstract. Robots operating in the presence of humans should adapt their plans and behavior accordingly. The recent area of human-aware planning (HAP) addresses this problem. In this paper, we propose a method for extending HAP to so-called Socially Aware planning. We build on known principles for human context recognition, extending them to support different social situations. In this new paradigm, we are able to define social norms and rules which are taken into account by the planning mechanism to obtain plans consisting of socially adjusted behaviors.

1 INTRODUCTION

Imagine the following scenario. There is a hospital with children, sometimes they are with a doctor, sometimes they play, sometimes they learn new things in combination with playing. One of the children, Tom, somehow has forgotten about school classes, and while the other children are in the classroom, Tom is playing in his room. The hospital in this story is very advanced, and it has robots. One of the robots notices that Tom is absent from the school room. It notifies the teacher, who confirms that the robot should inform Tom that school is in. Specifically, the robot's task is to navigate to the room where Tom is located, inform him verbally about the classes, and escort him to the school room. Imagine now a slightly different scenario, where Tom's roommate John is sleeping in the same room where Tom is playing. It would not be appropriate for our robot to use its voice to interact with Tom in this social situation, since this could wake up John. It would be much more (socially) acceptable to interact with Tom in a silent way, perhaps by beeping to capture Tom's attention and then showing a message on its display.

The above hospital is real and the story is part of a set of scenarios that is being realized in the European project MOnarCH [15]. The project focuses on developing robots to help children and staff in the hospital with various tasks, such as playing and learning. The project brings many specific scientific challenges, one of which is addressed in this paper.

Recently, planning was extended to support human activities, so that robots could adapt their own behaviors according to human needs. This is known as Human Aware Planning (HAP). There are different approaches to HAP, e.g., based on forward search [5], on constraint based planning [18], and on hierarchical tasks networks (HTN) [16]. We are interested in the problem of using the right behavior in the right social context. Pure HAP wouldn't suffice in this case, rather, it must be extended to the level where it is able to reason about the social context in which humans operate. Social norms, which relate social context with robot behavior, must be used by the planner to achieve appropriate human aware plans. We call this problem *Social Aware Planning*. In the above story, the robot should (1) recognize (infer) the social context in the room where Tom is playing; and (2) create plans depending on social norms defined by the recognized social context.

Since HAP has not, so far, addressed the issue of accommodating social norms, our work consists of developing a planner that is able to use social norms in order to support planning on a social level. We can differentiate between several problems that need to be solved. Beyond the necessary planning capability and dispatching mechanism for execution, we also need a human-aware component that is able to recognize human behavior and plan/act upon it. It is also necessary to put in place a mechanism for handling the social aspects required by the social environment. To do so, we introduce social norms and rules, which should be applied in appropriate social contexts. Taking into account that we are dealing with children, who can be characterized as having highly stochastic proprieties, we also need to be able to react as unexpected events occur on-line, and update plans or re-plan as soon as possible.

In this paper we propose a preliminary formalization of the Social Aware Planing problem, and we study how well a particular HAP solution lends itself to the social aware context. This solution is used to realize an example scenario inspired by the story above with a real robot.

2 CONSTRAINT BASED PLANNING

Our approach builds upon existing Constraint Based Planning (CBP) techniques (see [18, 8]). CBP retains many of the properties necessary for Social Aware Planning. First, CBP approaches generate temporally flexible plans and explicitly account for action timings. CBP approaches are appropriate for representing both qualitative and quantitative temporal requirements through temporal constraints. This type of planning also supports human awareness, by defining context variables which can represent states of the human. The values of context variable (e.g., human states) are inferred from the values of sensors in the environment. Interestingly, CBP allows to cast context inference and planning as by-products of the same algorithm [18]. Regarding fast reactions and on-line plan-based robot control, CBP supports constant feedback on relevant states in the system during execution to make sure that the system is converging towards the desired goals. Depending on actual development of events during execution, it may update the current plan or re-plan and update the execution queue. This is known as "closed-loop" planning and execution [7].

[1] This work was funded by the EC Seventh Framework Programme (FP7/2007-2013) grant agreement no. 601033 MOnarCH.

[2] Center for Applied Autonomous Systems (AASS), Orebro University, Sweden, contact: stevan.tomic@aass.oru.se

2.1 Representation

Variables. The domain representation is grounded on the notion of state variable. State variables have a symbolic and a temporal component, and represent elements in the environment which can be in one of several states. These can represent objects, like a door, which can be open or closed; people, who can be in certain position; or actions, like navigation actions of particular robot. The set of possible values that variables can have, is called a *domain*.

In our planning system, every element of a variable's domain is a pair $(\mathtt{v}, I)$, where $\mathtt{v}$ represents the state of the particular element the variable refers to (e.g., `open` in the case of a door), and I is a temporal interval representing *when* the particular state is assumed to be valid. For example, $(\mathtt{open}, [2, 13])$ states that the value `open` is assumed to be true starting at time 2 and until time 13.[3] The interval lends a temporal extent to variables, and allows us to use them for representing a current state, e.g., robot-1 is at the charging station; a past state, e.g., John was in his bed one minute ago; a predicted state, e.g., John will be at the entrance within two minutes; or a desired (goal) state, e.g., robot-1 should be at the entrance at 13:45.

Temporal Constraints. State variables can be correlated by means of temporal constraints. In our work, we represent temporal constraints in *Allen's Interval Algebra (AIA)* [3]. We use an extended version of AIA [13], in which relations can be both qualitative, like *before, after, meets* and so on; and quantitative, where relations come with bounded temporal intervals.

Constraint Network. The evolution of the world is described by a constraint network whose nodes represent actions, environmental states, etc., while edges are temporal relations holding between pairs of variables. An example is shown in Figure 1.

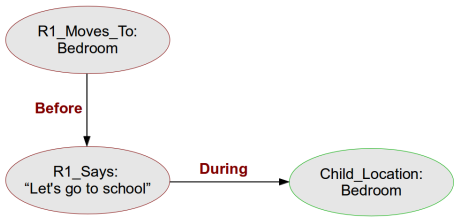

Figure 1. Example of constraint network: Robot have to move to the bedroom, before it can say: 'Let's go to school'. Also, robot can say this only during the child is in the bedroom.

In the constraint network, state variables and temporal constraints together provide the means to express states of relevant parts of the environment and how they change over time. The evolution of states is affected by other state variables representing actions performed by robots. The representation also allows to model non-predicted changes of state (e.g., due to human intervention, like the user turning on the light).

2.2 Reasoning

The planning process used in the CBP approach is incremental in nature, and yields a refined constraint network, which itself represents a plan which achieves the given goals.

[3] More genrally, the system uses flexible intervals in which both the start and the end time have a lower and an upper bound.

The resulting constraint network represents one or more temporal evolutions of the state variables that guarantee the achievement of the goals under nominal conditions. Feasible and goal-achieving plans are obtained by means of one or more solvers, operating on the same constraint network. In this paper, we will describe some of them relevant to our research.

Temporal solver. The temporal consistency of the constraint network is checked through temporal constraint propagation by means of a Simple Temporal Problem (STP) [6] solver. The solver propagates temporal constraints to refine the bounds of the variables in the network, and returns failure if and only if temporally consistent bounds cannot be found.

State variable scheduler. State variable scheduling ensures that state variables do not take on conflicting states in overlapping intervals. This solver posts temporal constraints which impose a temporal separation between conflicting variables. For example, the constraint network may contain a variable encoding the robot's interaction behavior with the values $(\mathtt{askToFollowMe}, I')$ and $(\mathtt{sayWelcome}, I'')$. The state variable scheduler will take care that those two values do not occur at the same time, which it achieves by imposing the temporal constraint I' *before* I'' (or vice-versa).

Planning Module. The task of this solver is to modify the constraint network by adding appropriate variables (usually robot actions) and constraints in order to achieve goals. Operators in the domain describe the causal dependencies between variables — e.g., that an operator achieves a certain effect. The solver instantiates (into the constraint network) relevant operators in the form of state variables and temporal constraints in order to enforce the causal dependencies of the plan. For example, if we have a constraint network with state variables indicating that the robot is in the kitchen, but also another variable representing current location of the robot which is not the kitchen, then planner may add a state variable representing the fact that the robot should move from its current location to the kitchen (with appropriate constraints).

3 ENCODING SOCIAL BEHAVIOR

3.1 Enter the Human's Context

In addition to representing the state of the environment and/or the robot's actions, state variables can be used to represent (known or predicted) states of humans. For instance, a variable can be used to represent particular states of the user, e.g., being asleep, playing, reading, and so on. Since variables include a temporal interval, a set of variables asserting the state of a human can be used to represent behavior. Temporal constraints can then be used to model the temporal relations between human behavior and other variables. Sets of such temporal constraints can thus be used to model the criteria for recognizing human behavior from sensor traces. Planning and behavior recognition can be integrated loosely, by allowing a context recognition system to determine the initial state of the planner, as done by Cirillo et al. [5]. In loosely coupled behavior inference and planning, the connection between robot actions and recognized behavior is not modeled. Ideally, we would like our planner to infer human behavior and plan for it contextually — that is, achieve a human-aware form of planning in which goals appear as a consequence of (1) external imposition (e.g., a nurse instructing a robot to "go and fetch the kids for class"); and (2) contextual inference (e.g.,

the system realizing that class has started, but a child is still playing in the playroom).

Proactive Planning. Our domain is encoded in a so-called "proactive" planning paradigm, which builds on the CBP principles described above [18]. The term "proactive" refers to its ability to post goals proactively, as a consequence of context inference (due to its tightly coupled inference and planning, as mentioned above). It defines different types of state variables: context, sensors and behaviors. A sensor is a variable whose value derives from a real or simulated sensor. Behavior variables represent robot behaviors. The values of context variables are inferred from the states of other variables in the network. For example, the value "SkippingSchool" of context variable "ChildState" is inferred through the following set of constraints, collectively called an *operator* (we use the format `variable::value`):

```
(Sensor ChildLocation)
(ContextVariable ChildState)

(Operator
 (Head ChildState::SkippingSchool)
 (RequiredState req1 ChildLocation::Bedroom)
 (Constraint Finishes(Head,req1))
)
```

Specifying an operator consists in listing all the conditions that must be present in the network for the `Head` of the rule to be asserted. These are either `RequiredStates` (i.e., symbolic values of state variables) or `Constraints` modeling the temporal relations among these values. For `SkippingSchool`, for instance, the first requirement (`req1`) is that the child is in the bedroom (`ChildLocation::Bedroom`). We also want `SkippingSchool` to be true until the `ChildLocation` sensor reports a different value (AIA constraint *finishes*).

Proactiveness and goal posting follow the same principle: a required state, or the head of a rule, may refer to a state variable modeling an active behavior rather than a context variable. If we added, for instance, the requirement `(RequiredState req2 Robot::ReachChild)` to the above rule, this would produce a subgoal to appear in the constraint network representing this desired behavior of the robot. As usual, temporal constraints can be used to model the necessary temporal relation between this requirement and the head or other requirements in the rule. Furthermore, other operators may be used to describe other necessary actions that must be taken for the robot to actually carry out the `ReachChild` action (e.g., turning on its camera).

3.2 Adding a Social Context

In this paper, we leverage heavily the concepts introduced by proactive planning, in particular the notion of context variables. They are good candidates for representing a social context, which is required to solve our initial problem. By modeling social aspects in the environment as context variables, we extended proactive planning only on its semantic level, gaining the ability to plan in various social situations. More concretely, to encode social context in the initial example, we might model another context variable that defines the social context of the situation we are interested in. For example, if John is sleeping in the same room as Tom, then the `SocialContext` variable will have a value `Silent`, indicating that any behavior carried out by the robot should be as silent as possible. Social context is thus inferred from the sensors in the environment. The following example shows how we can model the `SocialContext` context variable's value `Silent`.

```
(ContextVariable SocialContext)

(Operator
 (Head SocialContext::Silent)
 (RequiredState req1 OtherChild::Sleeping)
 (Constraint Finishes(Head,req1))
)
```

As in the previous example, this means that the `SocialContext` with value `Silent` will be true only when the sensor value of other child has the value `Sleeping`. Also, `Silent` will be true until the sensor ceases to indicate this reading, as modeled with the `Finishes` temporal constraint.

Social norms are represented as constraints between particular values of the social context variable and concrete robot actions. Hence, we can model constraints so that while the social context is silent, the robot is able to execute only silent robot actions — for example, that the robot can inform Tom only silently with a beep and message on its screen.

For a clearer view and better understanding of the mechanisms described above, we describe how we encoded our story into the planner's domain description language. We list all the concepts that are discussed previously in the text, e.g., variables representing robot behaviors, sensors, context variables, etc.

3.2.1 Domain Description

Sensors. In the domain language of the planner, we specify two sensors:

- `ChildLocation`: the location of the child of interest; it can be `Bedroom`, `Corridor` or `School`.
- `OtherChild`: modeling whether the other child is sleeping or not; its possible values are `Sleeping` and `None`.

Context Variables. In the domain, we also define two context variables. Their values describe the context that is inferred from the sensors:

- `ChildState` represents the state of the child from the perspective which the planner is interested in (the child's attending school). It is defined so that if the child is in the bedroom (information coming from the `ChildLocation` sensor), it will have value `SkippingSchool`; if the child is in the corridor, the variable assumes value `Engaged`, describing that the child is available for the escort behavior; if the child is in school, this context variable will have value `InTheSchool`. Based on these values, goals (defined in the domain) are posted in the constraint network, which the planner satisfies by applying the appropriate operators for robot action modeled in the domain (described below).
- `SocialContext` models the current social context. The value of this variable is also inferred from the sensor value `OtherChild`: when the `OtherChild` is `Sleeping` the `SocialContext` variable will be `Silent`, it will be `None` (see definition of `SocialContext::Silent` in Section 3.2).

Behaviors. The domain defines three state variables modeling the robot's capabilities:

- `RobotInteraction`: a state variable whose values represent the different behaviors of the robot. These are:
  - `FollowMe`: the robot informs the child that school has started, asking him/her if escorting help is needed. In our domain language this behavior is defined as follows:

```
(Operator
 (Head RobotInteraction::FollowMe)
 (RequiredState req1 RobotMoveTo::Bedroom)
 (RequiredState req2 ChildState::SkippingSchool)
 (RequiredState req3 SocialContext::NoContext)
 (Constraint After(Head,req1))
 (Constraint Duration[1500,INF](req1))
)
```

Note that the robot's interaction behavior `FollowMe` is applicable only when `SocialContext` has value `NoContext` and `ChildState` is `SkippingSchool`. Additionally, the operator requires an action by another robot (described later). There are two constraints in the definition, indicating that `FollowMe` must be *after* robot movement to the bedroom, and that movement must have some *duration*.

- `DisplayFollowMe`: a variant of the above, requiring the robot to display the request it on its screen.

```
(Operator
 (Head RobotInteraction::DisplayFollowMe)
 (RequiredState req1 RobotMoveTo::Bedroom)
 (RequiredState req2 ChildState::SkippingSchool)
 (RequiredState req3 SocialContext::Silent)
 (Constraint After(Head,req1))
 (Constraint Duration[1500,INF](req1))
)
```

Note that `DisplayFollowMe` is applicable *only* when context variable `SocialContext` is `Silent`.

- `Welcome`: the robot verbally welcomes the child to school once escorting has completed.

- `RobotMovesTo`, whose values are:
  - `School`: the robot moves to schoolroom.
  - `Bedroom`: the robot moves to bedroom.
- `EscortTo`, modeling escorting capabilities and whose domain contains only one location: `School`

4 ILLUSTRATIVE EXPERIMENT

The purpose of this section is to demonstrate how things work together in practice as well as to further clarify the underlying concepts. The key-point that we want to stress is that in different social situations our planner will find different solutions to achieve the same goal, if such a solution exists. Thus, we created two similar scenarios where the only difference is the social context. The goal remains the same in both scenarios.

4.1 Description and Methods of the Experiment

Trials. The domain described above was employed to run a trial of the system with a TurtleBot robotic platform [1]. Sensors values are controlled by the experimenter with the keyboard, and behaviors are simplified as explained further down. The middleware used was ROS [20], and the planner was wrapped by a ROS node. The experiment consisted of two trials. Upon starting the system and running the planner node, there was no input from the sensors. Then, the experimenter simulated values `Bedroom` and `None` for the sensors `ChildLocation` and `ChildState` respectively. The value of the sensor `OtherChild` was set to `None` throughout the first trial, reflecting that there was no other child in the bedroom. Upon activation of the sensor readings, the planner inferred two values for its context variables: the value of `ChildState::SkippingSchool` and the value for `SocialContext::NoContex`. At that point, in order to support the `SkippingSchool` value, as described in the domain description, the planner posted the goal `RobotInteraction::FollowMe`, reflecting that the robot should verbalize the request to follow it. The goal was satisfied by requiring the robot to move to the bedroom before verbalizing the request. The first action in the plan was dispatched, and the TurtleBot moved to the bedroom where a human participant was located. Upon finishing the action, the TurtleBot executive published the status message as `SUCCEEDED` on a feedback topic. The planner node, which subscribes to this topic, dispatched the next behavior, `RobotInteraction::FollowMe`. Instead of actually verbalizing the request, the behavior was implemented in such a way as to make a predefined sound for simplicity. Also for simplicity, the planner assumed that the child agreed to go to school. The experimenter then simulated the value `ChildLocation::corridor`. The context variable `ChildState` became `Engaged`, meaning that the child was ready to be escorted to the school. Since one of requirements of this inference was the escorting behavior, this was posted as a goal, leading the planner to enrich the plan using the escorting behavior. This behavior was implemented by means of the built-in "Follower" TurtleBot demo node, which resulted in the robot following the experimenter down the corridor. Once the robot and experimenter were in the schoolroom, the latter hit the keyboard button to set the value of the `ChildLocation` sensor to `Schoolroom`. At this point, the context variable `ChildState` became `InTheSchool`, and the planner posted the goal `RobotInteraction::Welcome`. This behavior was dispatched to the robot, and consisted in playing another predefined sound.

The second trial followed the same procedure, with the only difference that that sensor value for the `OtherChild` was `Sleeping`, indicating that there was another child sleeping in the bedroom. Thus the context variable `SocialContext` assumed value `Silent`. Also, the same goal to inform the child was posted, and the planner synthesized a plan to inform him and escort him to the school room. Under the new social context, the planner could not use the same behavior as in the first trial, rather selecting the behavior `RobotInteraction::DisplayFollowMe`, which led beeping and the message being showed on screen. The trial proceeded similarly to the previous case, with the robot dispatching behaviors and informing it of successful completions of behaviors.

4.2 Hardware and Software Framework

Environment. The experiment was carried out in the PEIS Home environment [21] at Örebro university (see fig. 2). The environment has 3 rooms, two of which were used to represent schoolroom and the bedroom. The room between those two rooms was used as a corridor. We used a Turtlebot 2 robot [1] to execute the planner's commands, and an adult human male participant acted as the child skipping school. The robot was equipped with a laptop (Intel Celeron(R) CPU 1.10Ghz x2 and 3.8GB RAM) with Ubuntu 12.04.1 LTS (32-bit) and ROS Hydro.

Software Architecture. Two ROS nodes were developed to realize the experiment:

- A node encapsulating the planner. The planner is based on the Meta-CSP Framework (see `metacsp.org`) and is implemented in Java. As a consequence, the ROS note was written in ROSJava. The node publishes custom ROS message that contain information about the current dispatched action, coordinates of the location for navigation behaviors, and the upper bound of the behavior's duration, which is used as a timeout by the robot executive. Messages

Figure 2. The PEIS environment and the Turtlebot 2. In the experiment, one room represented the bedroom, another the schoolroom, and the room between them was used as a corridor.

are published only when the planner dispatches a behavior. This node subscribes to a feedback topic that informs it of behavior termination. The node runs from a remote location (not locally on the Turtlebot), publishing ROS massages over the network to the ROS master running on the Turtlebot's laptop.

- A node responsible for receiving planned behaviors, setting up their execution and generating feedback messages. It subscribes to the topic published by the planner. This node prepares and sends all messages to lower-level Turtlebot nodes for behavior execution. It also subscribes to the Turtlebot's topics in order to monitor the status of execution, reflecting it in its own feedback messages for the planner. This node is written in c++, and it is executed locally on the turtlebot's laptop.

4.3 Results

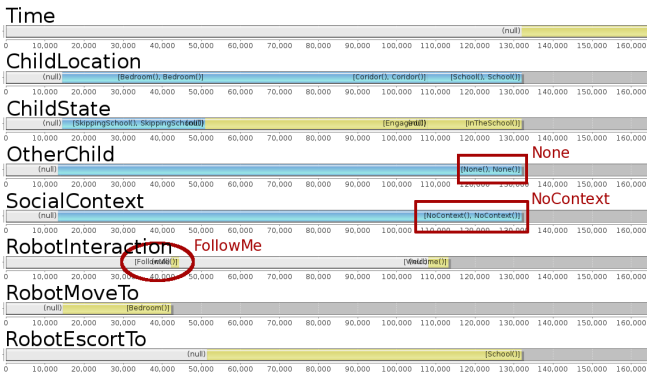

Figure 3. Time-lines of trial 1. The time-lines of each variable represent the evolution of its values over time. As shown, the `OtherChild` sensor variable has a value `None`, which implies `SocialContext::NoContext`, indicating that there is no social context of importance. Consequently, the executed behavior is `RobotInteraction::FollowMe`, indicating that robot is speaking (possibly loudly) to the child.

A video of the experiment is available online [10]. Figures 3 and 4 show the time-lines for the two scenarios. The only difference in the time-lines is robot's interaction behavior: in Figure 3 there is no child sleeping in the room, so the robot can execute the `RobotInteraction::FollowMe` behavior; in the second scenario (Figure 4) this is not possible since the

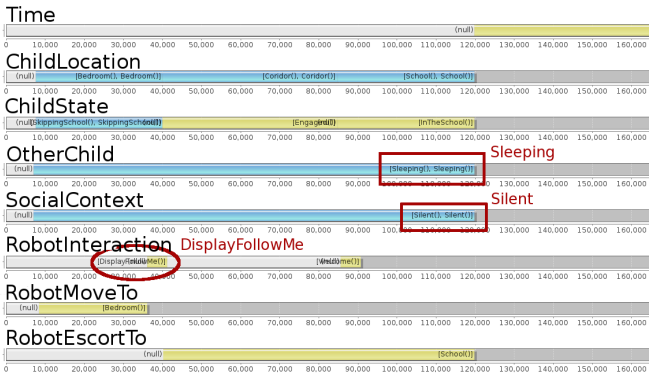

Figure 4. Time-lines trial 2. In this case, the value of the `OtherChild` sensor is `Sleeping`. Thus the planner infers that the `SocialContext` is `Silent`, and the robot executes behavior `RobotInteraction::DisplayFollowMe`.

other child is sleeping in the bedroom, prompting the robot to execute `RobotInteraction::DisplayFollowMe` as a result of the social constraint between `SocialContext` and `RobotInteraction` variables. This illustrates the main point discussed in this paper, namely the feasibility of using social constraints to model conditions on the robot's behavior.

5 DISCUSSION

Norms in our domain are still at the basic level. They are related to concrete variables, and they should be more general. They are fixed, meaning that cannot be changed online. They are also built as hard constraints, meaning that the robot must always follow them. In this section we discuss related work and other issue connected to generalizing the approach.

Köckemann et al. [12] have developed a planner that uses so-called *interaction constraints* to model relations between robot actions and human activities and preferences. A casual planner is combined with a constraint-based representation for modeling interaction constraints between humans and robot actions. Separating the casual planner from the interaction constraints leads to computational advantages in large-scale scenarios. In contrast, Pecora et al. [18] use context recognition techniques to infer the state of humans, which is further used to generate human aware plans. Our research can be seen as a combination of these two approaches: context inference is heavily leveraged, as in the latter, but extended to the social level; however the relation between human activities and the robot's action are not modeled by hand-coded "interaction constraints", rather as more general relations between social context and robot actions.

Boella [4] noted that the roots of social norms come from different scientific fields. In sociology from Gibbs [11] and Therborn [24], in philosophy from Alchourron and Bulygin [2]. In computer science, influential work underscoring the importance of using deontic logic to model relations between agents comes from Meyer and Wieringa [14]. These works have not been applied to a robotic context so far, but we plan to use them for inspiration to extend our approach to model more complex social norms and behaviors.

Regarding human environments, adding social norms into the robots' plans should lead to more acceptable behaviors of robots and better joint cooperation in human-robot interactions. But the concept of (social) norms is also central to multi-agent systems [4]. By introducing social norms into multi-agent systems, it is possible to model more complex relations between agents, to control the dynamics of agents interactions, to employ agent hierarchies, roles, etc.

Some or all of these concepts may be applicable to our research in social aware planning, and future work will investigate this avenue.

One of the important questions in research regarding norms is the impact norms have on agent autonomy. One way to address this problem is to have soft norms [4]. Dignum has divided social norms into three levels: the private level, the contract level, and the conventional level [9]. On the private level, an agent could decide not to carry out an obligation, thus paying a price for breaking it. He also uses a deontic logic to model relations between agents.

As Therborn notes [24], social rules and norms are usually not fixed in a given environment. In our future work we plan to use notion of institutions [23, 19] to model social dynamics and the relations between different sets of social norms and rules.

Generally speaking, there are two main reasons to use norms for regulating interaction between agents (robots or humans). One is human robot interaction. Humans are used to using different social norms in different social contexts. Artificial agents should also adhere to these or similar norms in order to behave naturally. Another, perhaps more interesting reason to use social norms, is for robot-robot/multi-agent interaction, e.g., to facilitate developing these interaction skills. If biological systems evolved to use them (see, e.g., social insects), can they be helpful in AI and robotics, and how? Norms are particularly useful for reducing the need to communicate explicitly among agents. Shoham and Tennenholtz explain how norms are useful in regulating the traffic of small mobile robots [22]. Following norms like "drive on the right side of the street" allows to avoid both constant negotiation between robots and centralized coordination. This view of norms was first explicitly stated by Moses and Tennenholtz [17]. The authors also argued that there is a trade-off in using norms: on one hand, norms limiting the degree of freedom of the agent; on the other, the same agent can expect certain behavior from other agents.

Both social norms and institutional frameworks may be considered as an additional level of complexity for the computational system responsible for an agent's behavior. However, their use may lead to simplified social dynamics and more coherent group behavior, significantly lowering the computational effort for the agent. This is another trade-off that will be more closely addressed in our future work. By measuring processing effort, we may be able to define quantitative means to measure the efficiency of adding norms in the interaction system.

With the introduction of social constraints into human-aware planning, we place our research at the intersection of at least two disciplines, namely human-aware planning and multi-agent systems. The advantage of our approach, for now, is its simplicity. We also see significant possibilities of extension in different directions.

6 CONCLUSIONS

In this paper we have argued that awareness of social rules and norms should be included in robotic systems that operate in human environments. We have defined a social aware planning approach by introducing constraints between human social context and robot actions. Our approach is based on the Constraint-Based Planning paradigm, which has already been employed in multi-robot planning [7] and HAP [12, 18]. A particular implementation of this approach was extended to support our idea.

The presented planning solution is illustrated on a simple example, which is meant to serve only as a proof of concept and as a starting point in this research. The planning methodology used is more general than the example suggests, and future work will include developing a general theoretical framework for Social Aware Planning. This will include a more general integration of norms, the study of dynamic norms using an institutional framework, and norms that could be broken in certain conditions.

REFERENCES

[1] Turtlebot 2. http://www.turtlebot.com/. retrived May 2014.

[2] Carlos E Alchourrón and Eugenio Bulygin, *Normative systems*, volume 5, Springer Wien, 1971.

[3] J.F. Allen, 'Towards a general theory of action and time', *Artificial Intelligence*, **23**(2), 123–154, (1984).

[4] Guido Boella, Leendert Torre, and Harko Verhagen, 'Introduction to normative multiagent systems', *Computational & Mathematical Organization Theory*, **12**(2-3), 71–79, (2006).

[5] Marcello Cirillo, Lars Karlsson, and Alessandro Saffiotti, 'Human-aware task planning: an application to mobile robots', *ACM Transactions on Intelligent Systems and Technology (TIST)*, **1**(2), 15, (2010).

[6] Rina Dechter, Itay Meiri, and Judea Pearl, 'Temporal constraint networks', *Artificial intelligence*, **49**(1), 61–95, (1991).

[7] Maurizio Di Rocco, Federico Pecora, and Alessandro Saffiotti, 'When robots are late: Configuration planning for multiple robots with dynamic goals', in *Intelligent Robots and Systems, 2013 IEEE/RSJ International Conference on*. IEEE, (2013).

[8] Maurizio Di Rocco, Federico Pecora, Prasanna Kumar Sivakumar, and Alessandro Saffiotti, 'Configuration planning with multiple dynamic goals', in *Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots, Stanford, California*. AAAI Press, (2013).

[9] Frank Dignum, 'Autonomous agents with norms', *Artificial Intelligence and Law*, **7**(1), 69–79, (1999).

[10] Experiment. https://www.youtube.com/user/kjernfs/videos. retrived May 2014.

[11] Jack P Gibbs, 'Norms: The problem of definition and classification', *American Journal of Sociology*, **70**(5), 586–594, (1965).

[12] Uwe Kockemann, Federico Pecora, and Lars Karlsson, 'Grandpa hates robots interaction constraints for planning in inhabited environments', in *Association for the Advancement of Artificial Intelligence*. AAAI, (2014).

[13] I. Meiri, 'Combining Qualitative and Quantitative Constraints in Temporal Reasoning', in *Artificial Intelligence*, pp. 260–267, (1996).

[14] John-Jules Ch Meyer and Roel J Wieringa, *Deontic logic in computer science: normative system specification*, John Wiley and Sons Ltd., 1994.

[15] MOnarCH. http://monarch-fp7.eu/. retrived May 2014.

[16] Vincent Montreuil, Aurélie Clodic, Maxime Ransan, and Rachid Alami, 'Planning human centered robot activities', in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pp. 2618–2623. IEEE, (2007).

[17] Yoram Moses and Moshe Tennenholtz, 'Artificial social systems', *Computers and Artificial Intelligence*, **14**, 533–562, (1995).

[18] Federico Pecora, Marcello Cirillo, Francesca Dell'Osa, Jonas Ullberg, and Alessandro Saffiotti, 'A constraint-based approach for proactive, context-aware human support', *Journal of Ambient Intelligence and Smart Environments*, **4**(4), 347–367, (2012).

[19] José N. Pereira, Porfírio Silva, Pedro U. Lima, and Alcherio Martinoli, 'Formalization, Implementation, and Modeling of Institutional Controllers for Distributed Robotic Systems', *Artificial Life*, **20**(1), (2014).

[20] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, 'ROS: an open-source Robot Operating System', in *ICRA Workshop on Open Source Software*, (2009).

[21] Alessandro Saffiotti, Mathias Broxvall, Marco Gritti, Kevin LeBlanc, Robert Lundh, Jayedur Rashid, BeomSu Seo, and Young-Jo Cho, 'The peis-ecology project: vision and results', in *Intelligent Robots and Systems, 2008. IEEE/RSJ International Conference on*, pp. 2329–2335. IEEE, (2008).

[22] Yoav Shoham and Moshe Tennenholtz, 'On social laws for artificial agent societies: off-line design', *Artificial intelligence*, **73**(1), 231–252, (1995).

[23] Porfírio Silva and Pedro U Lima, 'Institutional robotics', in *Advances in Artificial Life*, 595–604, Springer, (2007).

[24] Göran Therborn, 'Back to norms! on the scope and dynamics of norms and normative action', *Current Sociology*, **50**(6), 863–880, (2002).