Too Cool for School – Adding Social Constraints in Human Aware Planning

Stevan Tomic and **Federico Pecora** and **Alessandro Saffiotti** [1]

Abstract. Robots operating in the presence of human 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 up on the known principles for human context recognition, by extending them to support different social situations. In this new paradigm, we are able to define social norms and rules, which then can be added to the planning mechanism and as a result, to have plans which consists 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 stuff 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 in the school room. It tells that to the teacher, who issues a command to the robot to go to inform Tom about the school. The robot's task is to navigate to the room where Tom is located, inform him about the classes with its voice and then help him by escorting him to schoolroom. 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 the one of scenarios described in the European project MOnarCH [14]. This project deals with managing a group of robots to help children and staff in the hospital with various tasks as playing and learning. The project brings many scientific challenges, one of which we are addressing in this paper.

Recently, planning was extended to support human activities, so robots can adapt their own behaviors accordingly to human needs. This is known as human aware planning (HAP). There are different approaches to it, e.g., based on forward search [5], on constraint based planning [17], and on hierarchical tasks networks (HTN) [15]. We are interested in the problem of using 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 situation in which humans operates, and depending on that, use social norms to plan different 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 recognized social context.

Since HAP has so far not addressed the issue of accommodating social norms, we need a planning method, which is extend-able enough but also deals with most of the other problems in the story, in order to extend it to support planning on social level. We can differentiate between several problems that need to be solved. Behind obvious – planning and dispatching behaviors to the robot, we also need a human aware component, that is able to recognize human behavior and plan/act upon it. We also need a mechanism, that handles 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 the things happen 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.

2 Constraint Based Planning

To realize our approach to social aware planning, we build upon existing Constraint Based Planning (CBP) techniques (see [17, 8]) since these have most of the listed characteristics that we need. First, they are characterized by temporal flexibility. Thay can take care of action's timings, since they are explicitly represented in the planning domain, and can represent qualitative as well as quantitative temporal relations 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. Interesting, in the CBP paradigm context inference and planning are different products of the same algorithm [17]. Regarding fast reactions and on-line planning control, this planning approach supports constant feedback on relevant states in the system during execution to make sure that the system is converging towards desired goals. Depending on actual development of events during execution, it may update current plan or re-plan and update execution queue. This is known as "closed-loop" planning and execution [7].

2.1 Representation

Variables. In the planning domain everything is represented through state variables. State variables have a symbolic and a temporal component, and represent elements in the environment which

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

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 domain is a pair $(\mathtt{v}, I)$,where 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, $(\mathtt{open}, [[2,5][13,22]])$ states that the value open is assumed to be true starting between time 2 and 5, and until some time between 13 and 22. 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, in which relations can be both qualitative, like *before, after, meets* and so on; and quantitative, where relations come with a bounded temporal interval.

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

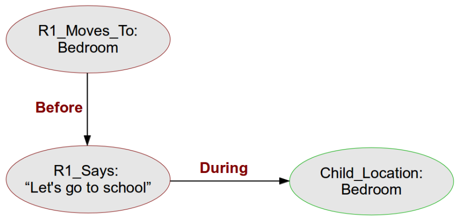

Figure 1. Example of constraint network: Robot have to move to the bedroom, before it can say: 'Let's go to school'. Also, robot can say this only during the child is in the bedroom.

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

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 activities 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 prescribe conflicting states in overlapping intervals. This solver posts temporal constraints which impose a temporal separation between conflicting variables. For example robot's behavior, encoded with variable 'RobotInteraction' can have two values – saying 'Follow me' and saying 'Welcome'. State variable scheduler will take care that those two values are not conflicted, e.g. both true in the same time.

Planning Module. The task of this solver is to modify the constraint network, by adding appropriate variables (usually robot's actions) and constraints, in order to make it consistent. Operators in the domain, encapsulates causal dependencies between variables. The solver then, instantiates (into the constraint network) relevant operators in the form of state variables and temporal constraints, to enforce the causal dependencies of the plan. For example, if we have constraint network, with state variables indicating that the robot is in the kitchen, but also another variable that represent current location of the robot, which is not the kitchen, then planner may add state variable representing movement of the robot from current location to the kitchen (with appropriate constraints), thus making the network consistent.

3 Encoding Social Behavior

3.1 Enter Human's Context

In addition to representing the state of the environment and or robot's actions (behaviors), 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. Temporal rules can also 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 [5]. As a consequence, in loosely coupling of behavior inference and planning, connection between robot activities and recognized behavior is not modeled. Ideally, we would like our planner to infer human behavior and plan for it contextually – i.e., achieve a human-aware form of planning in which goals appear as a consequence: (1) of external imposition (e.g., a nurse instructing a robot to "go and fetch the kids for class"); (2) of contextual inference (e.g., the system realizing that class has started, but a child is still playing in the playroom).

Proactive Planning. We encoded our domain in a so-called 'proactive' planning. It is built on the CBP principles described above, and it is simplified version of the [17]. The name 'proactive' comes from 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. Sensor is a variable that get its value from the real or simulated sensor. Behavior variable represent robot behaviors. Context variable is the one used to infer its value from the states of the other variables in the network. For example, the context variable 'ChildState' with a value 'SkippingSchool' is defined as:

```
(Sensor ChildLocation)
(ContextVariable ChildState)

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

First we define two variable types: sensor and context variable. Format that we are using in the domain is `variable::value`. Then we list all of the requirements, for 'SkippingSchool' to be true. First requirement (`req1`) is that the child is in the bedroom – indicated by 'ChildLocation::Bedroom'. We also want 'SkippingSchool' to be true until activity of the sensor finises. This is indicated by temporal constraint 'Finishes'.

Proactiveness and goal posting work as follows: In the definition of the value of the context variable, we can specify also robot's actions. In this case, whenever context value becomes active, proactive planner will post this action as a goal and it will try to solve it. For example, suppose that another requirement for 'SkippingSchool' value is certain robot's interaction behavior, where the robot is informing Tom about the school, e.g. saying “Follow Me”. This robot's action will be automatically considered as a goal. As a result of this mechanisms, as the 'SkippingSchool' becomes active, robot will go to the room where the Tom is, and start interacting with him.

3.2 Adding a Social Context

In this paper, we leverage heavily on the concepts introduced by proactive planning, in particular on the notion of 'context' variables. They are perfect candidates for representing a social context, which is required to solve our initial problem. By being able to introduce social aspects in the environment, as already existing context variables, we extended proactive planning only on its semantic level, gaining ability to plan in various social situations. More concretely, to encode social context in the initial example, we could have another context variable that defines social context of the situation we are interested in. For example, if John is sleeping in the same room as Tom, then the e.g. 'SocialContext' variable will have a value 'Silent', indicating that all of the behaviors that robot makes, should be as silent as possible. Even more, as already explained, the social context is inferred from the sensors in the environment. In the next example we show definition of 'SocialContext' context variable, with value 'Silent'.

```
(ContextVariable SocialContext)

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

Similarly as in previous domain 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 sensor finishes its activity due to `Finishes` temporal constraint.

Social norms are represented as constraints between particular values of the social context variable and the concrete robot's action. Hence, we could have constraint so that while the social context is silent, robot is able to execute only silent robot actions. For example, 'Silent' social context will be active while the John is sleeping, and in that case, robot could inform Tom only silently with a beep and message on its screen.

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

3.2.1 Planner's Domain Description.

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

- ChildLocation. Its value is the location of the child of interest. It can be ‘Bedroom’, ‘Corridor’ or ‘School’.
- OtherChild. Its purpose is to show weather the other child is sleeping or not. Its possible values are ‘Sleeping’ and ‘None’.

Context Variables. In the domain, we also defined two context variables. Their values describes the context that is inferred from the sensors. One context variable is named ‘ChildState’ and the other ‘SocialContext’.

- ‘ChildState’ is the variable showing the state of the child that the planner is interested in – child that should be in the school. It is defined so that if the child is in the bedroom (information coming from ‘ChildLocation’ sensor), it will have value ‘SkippingSchool’, if in corridor this variable will become 'Engaged', describing that the child is available for the escort behavior. If the child is in the school, this context variable will have value 'InTheSchool'. Based on these values, goals (defined in the domain) will be posted in the under-laying constraint network, and the planner will solve it by adding appropriate behaviors to make network consistent.
- ‘SocialContext’ – it shows the current social context and the value of this variable is also inferred from the sensor value ‘OtherChild’. Thus, when ‘OtherChild’ value is 'Sleeping' then ‘SocialContext’ variable will be ‘Silent’. Otherwise it will be ‘None’. In the 3.2 we showed a definition of 'SocialContext::Silent'.

Behaviors. There are couple of behaviors defined in the domain, that are used by the planner for solving goals. They are:

- RobotInteraction. It can have several values. Each represent different behavior of the robot:
  - FollowMe – Robot is informing the child that the school has started, and 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))
    )
    ```

    This shows that robot's interaction behavior 'FollowMe' will be true and thus be able to be used in the plan, only when 'SocialContext' has value 'NoContext' and 'ChildState' is 'SkippingSchool'. Additionally, since another robot's behavior is added as a requirement, it will be considered as a goal. Also, 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 – The purpose of this behavior is the same as above, but instead of saying it, robot should display 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))
)
```

  Except the same definitions as in 'FollowMe' behavior, the important part here is that 'DisplayFollowMe' will be true *only* when context variable 'SocialContext' is 'Silent'.

- Welcome – Robot just says 'welcome' once the child is successfully escorted to the school.

• RobotMovesTo.

  - School – Robot moves to schoolroom.
  - Bedroom – Robot moves to bedroom.

• EscortTo. Used to escort child to the location. In the domain we have 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 under-laying 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 solution exists. Thus, we created two similar scenarios where the only difference is the social context by which robot plans and executes its behaviors. The goal remains the same in both scenarios.

4.1 Description and Methods of the Experiment

Trials. In the real system used in the MOnarCH project, the planner will receive a high level inputs from other modules. Same goes for the behaviors. In order to carry out our trials on turtlebot and to concentrate on key-points we want to stress, we simplified inputs and behaviors. Sensors values are controlled by the experimenter with the keyboard, and behaviors are simplified as explained further down. As already noted there were two trials in the experiment. Upon starting the system up, and running the planner node, there was no input from the sensors. Then, the experimenter simulated values for two sensors. The value of 'ChildLocation' become 'Bedroom' and the value of 'OtherChild' became 'None' indicating that there ware no other child in the bedroom. Upon activation of those sensors, the planner inferred two values for its context variables. The value of 'ChildState' became 'SkippingSchool' and the value for 'SocialContext' became 'NoContex'. At that point, in order to justify SkippingSchool value, as described in the domain description, the planner posted a goal in its network — say 'Follow me' to the child. To satisfy its goal, it found a plan to first move robot in the bedroom, then say to the child 'follow me to the school'. So, planner dispatched its first action, and the turtlebot moved in the bedroom where the human participant was located. Upon finishing the action, the node on turtlebot published its status message as 'SUCCEEDED' on the feedback topic, and since the planner node was subscribed to it, the planner dispatched the next behavior, which was 'FollowMe'. Instead of actually saying: 'Follow Me' and explaining situation to the child, turtlebot made a predefined sound for simplicity, and then finished this behavior. Also for simplicity, the planner assumed that the child agreed to go to school. Then the experimenter forced the value 'corridor' in the 'ChildLocation' sensor. The context variable 'ChildState' became 'Engaged', meaning that the child is ready to be escorted to the school. Since one of preconditions to this inference was escorting behavior, it is posted as a goal and the planner founds a solution using the escorting behavior. This behavior is implemented with a help of 'Follower' turtlebot demo node, so that in the experiment robot was actually following experimenter. Once the robot and participant were in the schoolroom, experimenter hit the keyboard button to set values of the 'ChildLocation' sensor, to 'Schoolroom'. At this point, context variable 'ChildState', became 'InTheSchool' and with the same mechanism planner posts a goal – say 'Wellcome'. So, the last behavior that has been dispatched by the planner was – saying 'Welcome'. In the experiment version of this behavior, robot was playing predefined sound, instead real communicative act.

In the other trial we followed the same procedure. Only difference was that sensor value for the 'OtherChild' was 'Sleeping'. It indicates that there were other person in the bedroom that was sleeping. In this case, the other context variable, 'SocialContext', got activated, and then, goal was posted. The goal was the same: inform the child that the school has started and prepare him/her for escorting action. But, in this case, 'knowing' that there is a social context, planner couldn't use behavior used in previous trial, since it is not appropriate for this social context. Instead there was another behavior called 'DisplayFollowMe', which was appropriate, and thus planned and executed. It displays message about escorting on the robot's screen. The rest of procedure was the same as in previous trial.

In summary, in the first trial, where there was only one person present, turtlebot went to the person in bedroom, executed RobotInteraction::FollowMe behavior, then it follows the person to the schoolroom and finished trial by executing RobotInteraction::Welcome behavior. In the second scenario, other person was sleeping in the bedroom (according to sensors), so turtlebot went to the bedroom executed RobotInteraction::DisplaFollowMe which was better suited for that context and continue execution as in the first trial.

4.2 Hardware and Software Framework

Environment. Experiment is carried out in the PEIS Home environment [19] at Orebro university (see fig. 2). PEIS environment has 3 rooms and we used two of them to represent schoolroom and the bedroom. The room between those two rooms, is used as a corridor. We used a Turtlebot 2 robot [1] to execute planner's commands and adult human male participant for human-robot interaction. Turtlebot 2 was equipped with laptop (Intel Celeron(R) CPU 1.10Ghz x2 and 3.8GB RAM) with Ubuntu 12.04.1 LTS (32-bit) and installed ROS Hydro.

Software Architecture. We develop two ROS nodes in order to be able to execute behaviors generated by the planner:

• Node that encapsulates the planner. Since planner is based on meta-CSP framework and since it is written in Java, node that encapsulates it is a rosjava node. It publishes custom ROS message that contains information about the current dispatched action, coordinates of the location in case it is navigation behavior and the upper bound of the behavior's duration – timeout. Message is

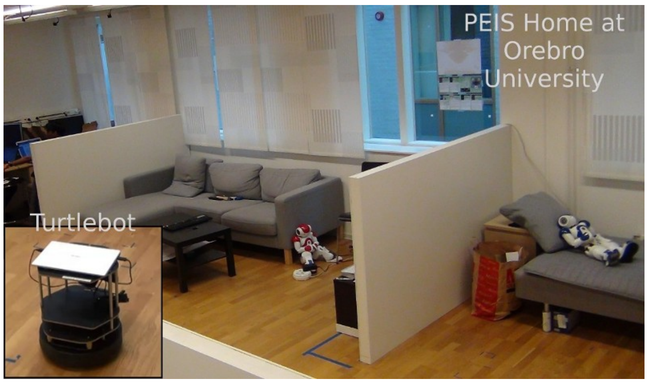

Figure 2. PEIS environment and the turtlebot. In the experiment one room represented the bedroom, another represented the schoolroom, and the room between them were used as a corridor.

publish only when planner dispatches a behavior. This node subscribes on a feedback topic, so it has information about the status of last dispatched behavior. Planning node runs from a remote location (not locally on the turtlebot). It publish ROS massages over the network to the ROS master that is on the turtlebot's laptop.

- This node is responsible for receiving planned behaviors, setting up their execution and sending feedback to the planner about the status of the behaviors currently executing on the robot. It subscribes on the topic on which planner is sending its messages containing the information about behavior that needs to be executed next. Then, this node, prepares and sends all messages to turtlebot nodes, so that desired behavior can be executed. It also subscribes to the turtlebot's topics in order to monitor the status of the execution, and then it publishes the status to the feedback topic, to which planner is subscribed, so that planner can have feedback about the last action it dispatched. This node is written in c++, and it is executed locally on the turtlebot's laptop.

4.3 Results

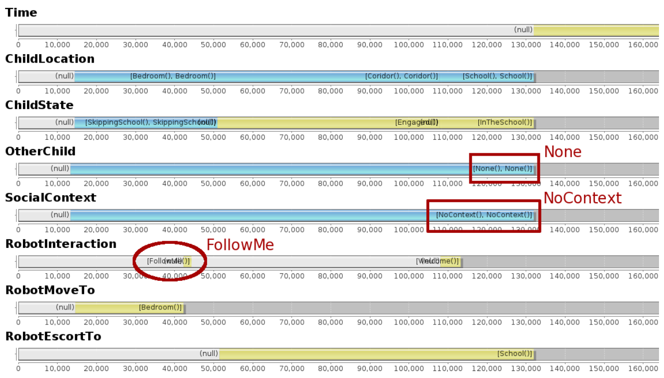

Figure 3. Time-lines of trial 1. Time-lines of each variable represent evolution of its values over time. As it is shown, 'OtherChild' sensor variable has a value 'None'. In this case 'SocialContext' variable has inferred value 'NoContext' indicating that there is no social context of importance to the planner. Since there is no special social constraint, executed behavior is 'FollowMe', indicating that robot is speaking (possibly loudly) to the child.

Experiment went as predicted. There is on-line video of the trials [10]. Figures 3 and 4, show the time-lines for the two different scenarios explained earlier. The only difference in the time-lines is robot's interaction behavior: in fig. 3 there is no child sleeping in the room, so robot is executing 'FollowMe' behavior. In the second scenario (fig. 4) there is the other child who is sleeping in the same room, so the executed (interaction) behavior is 'DisplayFollowMe', as a result of social constraint between 'SocialContext' and 'RobotInteraction' variables. Thus, we showed the main point discussed in this paper – using social constraint to model robot's behavior, and we showed one possible solution of the initial problem described in the story.

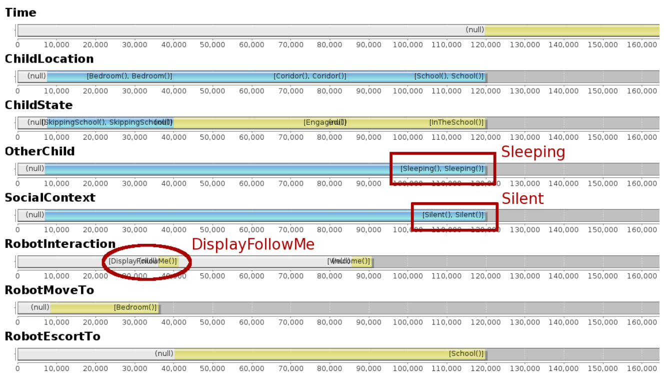

Figure 4. Time-lines trial 2. Time-lines of each variable represent evolution of its values over time. As it is shown, in this case 'OtherChild' value is 'Sleeping'. Thus, by inference mechanisms, 'SocialContext' variable has value 'Silent', and the robot executes 'DisplayFollowMe' action.

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 can not be changed later on the run. They are also built as a hard constraints, meaning that the robot must always follow them. Thus, in this section, we will discuss related work, and things that we plan to do to improve our research.

In [12] research, planner uses so called *interaction constraints* to model relations between robots actions and human activities and preferences. In their research, they are using a casual planner combined with CSP approach to model interaction constraints between humans and robot actions. By separating casual planer from the interaction constraints, they are gaining the computational advantage in the large-scale scenarios. In SAM architecture [17] as already mentioned, they are using context recognition techniques to infer human state, which is further used to generate human aware plan. Our research can be seen as a combination of those two approaches, since we are using context inference, as in SAM, but extended to social level. Moreover, the relation between the human activities and the robot's action are modeled by hand-coded "interaction constraints". By contrast, we model relations between social context and robot's action in a more general way using so called *social constraints* which then defines social norms.

Boella [4] noted that roots of social norms come from different scientific fields, in sociology from Gibbs (1965) [11] and Therborn (2002) [22], in philosophy from Alchourron and Bulygin (1971) [2]. In computer science, influential work underlying importance of using deontic logic to model relations between agents comes from Meyer and Wieringa, 1993 [13]. These work 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 the multi-agent systems. Their need in multi-agent systems are now well known and established [4]. By introducing social norms into the multi-agent systems, it is possible to model more complex relations between agents, to control dynamic of agents interactions, to have agent's hierarchies, their roles, etc.

One of the important questions in research regarding norms is how much autonomous agents are autonomous if forced to follow norms. One way to address this problem is to have soft norms [4]. Dignum has divided social norms in three levels: the private level, the contract level and the conventional level [9]. In the private level agent could decide not to do its obligation, thus paying a price for breaking it. He also uses a deontic logic to model relations between agents.

As Therborn [22] noted, social rules and norms in the given environment are usually not fixed. In our future work we plan to use notion of institutions [21] [18] to model social dynamic and use of different sets of social norms and rules.

Generally speaking, there are two main reasons to use norms in the systems based on interaction between agents (robots or humans). One is human robot interaction. Humans are used to use social norms, and even further they are using different social norms in different social contexts-es. In order robots to behave naturally they must leverage on this fact. Another, maybe more interesting reason of using social norms, is in the robot-robot interaction or multi-agent interactions. In this approach norms are used by their intrinsic meaning. One can ask why social norms at all, what is their meaning, what they are doing there at the first place? If biological systems evolved to use them (especially true in the case of some social insects), can they be helpful in AI and robotics, and more importantly how? Norms are particularly useful to lower communication between agents in the multi-agent systems. Shoham and Tennenholtz explained usefulness of norms in their research on the example of a traffic system of small mobile robots [20]. If they were following norm as e.g. 'keep right side of the street', then they could avoid both, a constant negotiation between robots and centralized coordinator. This view of norms was first explicitly stated by Moses and Tennenholtz [16]. Shoham and Tennenholtz also argued about trade-off in using norms, where norms are limiting the degrees of freedom of the agent, but from the other hand, the same agent can expect certain behavior from other agents.

Both, social norms and institutional framework may be considered as an additional level of complexity for computational system of agent's behavior. On the other hand, this may lead to simplified social dynamic and more coherent group behavior, so that actual processing effort for the agent will be significantly lowered. This is another trade-off that will be more closely addressed in our future work. By measuring processing effort, we may be able to quantitatively define exact measure of efficiency of adding norms in the interaction system.

With introduction of the social constraints into human aware planning, our research can be seen as intersection point of at least two disciplines, human aware planning and multi-agent systems. Advantage of our approach is its simplicity and possibility to be extended to diffrent directions. Disadvantage however, is that it is based purely only on the semnatic extensions of the existing system.

6 Conslusion

We argued that awareness of social rules and norms should be included in robotic systems that operate in human environments. We have shown that this can be done in a relatively easy way by extending CBP. The examples used in this paper are extremely simple, and only serve as a proof of concept. Future work will address at least, more general integration of norms, their dynamic using 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, 'Planning in inhabited environments - human-aware task planning and activity recognition', *KI*, **25**(4), 355–358, (2011).

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

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

[15] 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).

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

[17] 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).

[18] 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).

[19] 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).

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

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

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