Incorporating Social Practices in BDI Agent Systems

Stephen Cranefield[1(✉)] and Frank Dignum[2,3,4]

[1] University of Otago, Dunedin, New Zealand
stephen.cranefield@otago.ac.nz
[2] Umeå University, Umeå, Sweden
frank.dignum@umu.se
[3] Czech University of Technology in Prague, Prague, Czech Republic
frank.dignum@aic.fel.cvut.cz
[4] Utrecht University, Utrecht, The Netherlands
f.p.m.dignum@uu.nl

Abstract. When agents interact with humans, either through embodied agents or because they are embedded in a robot, it would be easy if they could use fixed interaction protocols as they do with other agents. However, people do not keep fixed protocols in their day-to-day interactions and the social environment is often dynamic, making it impossible to use fixed protocols. Deliberating about interactions from fundamentals is not very scalable either, because in that case all possible reactions of a human have to be considered in the agent's plans. In this paper we argue that social practices can be used as an inspiration for designing flexible and scalable interaction mechanisms that are also robust. However, using social practices requires extending the traditional BDI deliberation cycle to monitor landmark states and perform expected actions by leveraging existing plans. We define and implement this mechanism in Jason using a periodically run meta-deliberation plan, supported by a metainterpreter, and illustrate its use in a realistic scenario.

1 Introduction

Imagine the scenario where a disabled person, living alone, is assisted by a care robot. The robot makes sure that the person gets up every morning and that he drinks some coffee and takes his morning pills (if needed). Then they read the newspaper together, which means that the person looks at the pictures in the paper and the robot reads the articles out loud for the person to hear.

When agents in the role of this type of personal assistant or care robot have to interact with humans over a longer time period and in a dynamic environment (that is not controlled by the agent), the interaction management becomes very difficult. When fixed protocols are used for the interaction they are often not appropriate in all situations and cause breakdowns and consequent loss of trust in the system. However, to have real-time deliberation about the best response during the interaction is not very scalable, because in real life the contexts are dynamic and complex and thus the agent would need to take many parameters into consideration at each step. Thus we need something

L. A. Dennis et al. (Eds.): EMAS 2019, LNAI 12058, pp. 109–126, 2020.
https://doi.org/10.1007/978-3-030-51417-4_6

in between a completely scripted interaction that is too brittle and a completely open interaction that is not scalable.

As we have done before in the agent community, we take inspiration from human interactions and the way they are managed by individuals. Humans classify situations into standard contexts in which a certain *social practice* can be applied. Social science has studied this phenomenon in social practice theory. Social practice theory comes forth from a variety of different sub-disciplines of social science. It started from philosophical sociology with proponents like Bourdieu [3] and Giddens [8]. Later on Reckwitz [16] and Shove [18] have expanded on these ideas, and also Schatzki [17] made some valuable contributions.

These authors all claim that important features of human life should be understood in terms of organized constellations of interacting persons, which together constitute social practices. People are not just creating these practices, but our deliberations are also based on the fact that most of our life is shaped by social practices. Thus, we use social practices to categorize situations and decide upon ways of behaviour based on social practices. The main intuition behind this is that our life is quite cyclic, in that many activities come back with a certain regularity. We have meals every day, go to work on Monday until Friday, go to the supermarket once a week, etc. These so-called Patterns of Life [7] can be exploited to create standard situations and expectations. It makes sense to categorize recurrent situations as social practices with a kind of standard behaviour for each of them.

Unfortunately social practice theory has not been widely used in computer science or in HCI and thus there are no ready-to-use tools in order to incorporate them in agents. It is clear from the above description that social practices are more than just a protocol or a frame to be used by the agent in its deliberation. Therefore, in this paper we make the following contributions. We propose a mechanism for BDI agents to maintain awareness about active social practices, and to leverage their existing plans to act in accordance with these practices. We focus on these aspects of social practices (discussed in more detail in Sect. 2): (a) they are relevant in specific contexts, defined in terms of the actors, resources and places involved; and (b) they are modelled as plan patterns, structured as a set of partially ordered landmarks, each with an associated purpose (a goal) and a sequence of actions that is a partial prescription for reaching the landmark.

Our mechanism is presented as a meta-deliberation plan that can be directly executed by Jason agents, or treated as a specification for an optimised implementation in an extended agent platform. This plan has been deployed in the (simulated) care robot scenario, to confirm that awareness of and adherence to a social practice enables the robot to have a more successful interaction with the patient over a longer period of time. As some of the features needed to implement this scenario, and to support our meta-deliberation plan, are not available in Jason[1], we also present a Jason metainterpreter, which provides this extended functionality, but can also be used independently to support other research on extensions to BDI practical reasoning.

[1] These features are finding a plan for a goal that ensures a given action is performed, and support for durative and joint actions.

In the next section, we give an introduction to the purpose and structure of social practices. Section 3 elaborates on the care robot scenario and how we have modelled it in Jason. Section 4 describes the role of social practices in this scenario, and discusses the requirements this imposes for a BDI agent. Section 5 presents our mechanism for extending Jason to leverage social practices, and the metainterpreter needed to support this. We finish the paper with some conclusions and suggestions for future work.

2 Social Practices

Social practices are defined as accepted ways of doing things, contextual and materially mediated, that are shared between actors and routinized over time [16]. They can be seen as patterns that can be filled in by a multitude of single and often unique actions. Through (joint) performance, the patterns provided by the practice are filled out and reproduced.

According to Reckwitz [16] and Shove et al. [18], a social practice consists of three parts:

- Material: covers all physical aspects of the performance of a practice, including the human body and objects that are available (relates to physical aspects of a context).
- Competence: refers to skills and knowledge that are required to perform the practice (relates to the notion of deliberation about a situation).
- Meaning: refers to the issues which are considered to be relevant with respect to that material, i.e. understandings, beliefs and emotions (relates to social aspects of a situation).

Let us consider these three parts of a social practice in the scenario of the care robot scenario introduced in Sect. 1. Material refers to the room where the robot serves morning coffee for the disabled person. It includes the materials that are needed to make coffee (such as coffee and a coffee maker) and serve it (such as a cup and tray). However, it also includes the table and other furniture in the room, the newspaper (if present), the TV, radio, computer, tablet, and the robot and person (and possible other people that may be present).

Competence describes the activities every party can perform and expectations about what they will actually do. For example, the robot is capable of making coffee and serving it. The person can drink his coffee by himself. They can jointly read the newspaper or watch TV. The expectation is that the robot wakes the person if he is not awake yet, makes the coffee and gives it to the person. After that they will read the newspaper together to provide mental stimulation. Note, these are expectations, not a protocol. So, parties can deviate from it and they can also fill the parts in, in ways they see fit best.

Meaning has to do with all the social interpretations that come with the social practice, e.g. drinking coffee in the morning might give the person a sense of well-being that he can use to face the challenges of the rest of the day. When the coffee is cold or weak the person might interpret it as disinterest on the part of the robot in his well-being. The goal of reading the newspaper might also be not just to get the information from it, but a form of entertainment and feeling related to the robot, because the human and robot are doing something together.

From the above description it can already be seen that social practices are more encompassing than conventions and norms. Conventions focus on the strategic advantage that an individual gets by conforming to the convention. The reason to follow a convention is that if all parties involved comply, a kind of optimal coordination is reached, i.e. if we all drive on the left side of the road, traffic will be smoother than when everyone chooses the side to drive on freely. Thus, conventions focus on the actual actions being performed and how they optimize the coordination. Social practices focus on common expectations and ways to achieve them. For example, if we go to a presentation, we sit down as soon as we see chairs standing in rows in the room. However, we could also keep standing (as is often done outside).

Social practices are also different from norms. Norms usually dictate a very specific behaviour rather than creating a set of loosely coupled expectations as is the case for social practices. For example, if the norm states that a car has to stop for a red light, it gives a very specific directive. If a norm is more abstract (like "drive carefully") then we need to translate this into concrete norms for specific situations.

One framework that seems very close to social practices is the notion of *scripts*. However, social practices are not just mere scripts in the sense of Minsky [14]. Practices are more flexible than the classical frames defined by scripts, in that they can be extended and changed by learning, and the "slots" only need to be filled in as far as they are needed to determine a course of action. Using these structures changes planning in many common situations to pattern recognition and filling in parameters. They support, rather than restrict, deliberation about behaviour. For example, the social practice of "going to work" incorporates usual means of transport that can be used, timing constraints, weather and traffic conditions, etc. Normally you take a car to work, but if the weather is exceptionally bad, the social practice does not force the default action, but rather gives input for deliberation about a new plan in this situation, such as taking a bus or train (or even staying home). Thus, social practices can be seen as a kind of flexible script. Moreover, scripts do not incorporate any social meaning for the activities performed in them as social practices do.

Social practices have been used in applications in a variety of ways. In [12,15] they have been used as part of social simulations. In those applications, social practices are used as a standard package of actions with a special status. Thus individuals can use them with a certain probability given the circumstances are right. However, these applications do not use the internal structure of social practices for the planning of the individuals. Social practices have been used for applications in natural language and dialogue management in [1,9]. Here, the social practices are used to guide the planning process, but are geared towards a particular dialogue rather than as part of a more general interaction. In [13] it is shown how social practices can be used by a traditional epistemic multi-agent planner to provide efficient and robust plans in cooperative settings. However, in this case the planner was not part of a BDI agent with its own goals and plans, but completely dedicated to finding a plan for the situation at hand. In [6] a first structure of social practices was presented that is more amenable for the use by agents. The paper is only conceptual and no implementation was made yet. In this paper we will follow the structure described in [6]. However, we mainly concentrate on the plan patterns that are a core part of the social practices and show how they work with BDI agents in the Jason platform.

The complete structure for social practices (based on [6]) is as follows:

Context

- *Roles* describe the competencies and expectations about a certain type of actor. Thus the robot is expected to be able to make a cup of coffee.
- *Actors* are all people and autonomous systems involved, that have capability to reason and (inter)act. This indicates the agents that are expected to fulfil a part in the practice. In our scenario, these are the robot and the person.
- *Resources* are objects that are used by the actions in the practice, such as cups, coffee, trays, curtains, and chairs. So, they are assumed to be available both for standard actions and for the planning within the practice.
- *Affordances* are the properties of the context that permit social actions and depend on the match between context conditions and actor characteristics. For example, the bed might be used as a chair, or a mug as a cup.
- *Places* indicates where all objects and actors are usually located relative to each other, in space or time: the cups are in the cupboard in the kitchen, the person is in the chair (or in bed), etc.

Meaning

- *Purpose* determines the social interpretation of actions and of certain physical situations. For example, the purpose of reading the newspaper is to get information about current affairs and to entertain the person.
- *Promotes* indicates the values that are promoted (or demoted, by promoting the opposite) by the social practice. Giving coffee to the person will promote the value of “caring”.
- *Counts-as* are rules of the type “X counts as Y in C” linking brute facts (X) and institutional facts (Y) in the context (C). For example, reading the newspaper with the person counts as entertaining the person.

Expectations

- *Plan patterns* describe usual patterns of actions defined by the *landmarks* that are expected to occur (states of affairs around which the inter-agent coordination is structured). For example, the care robot first checks if the person is awake then makes sure there is coffee served. Landmarks are usually very naturally given by the people involved. They describe a social practice in terms of the phases of which it consists and use the landmarks to denote fixed points that have to be reached before the next phase can start.
- *Norms* describe the rules of (expected) behaviour within the practice. For example, the robot should ask the person if he wants coffee, before starting to make it.
- *Strategies* indicate condition-action pairs that can occur at any time during the practice. For example, if the person drops the coffee, the robot will clean it up. If the robot notices the person is asleep (again) it will try to wake him.

- A *Start condition*, or trigger, indicates how the social practice starts, e.g. the practice of having morning coffee starts at 8 am.
- A *Duration*, or *End condition*, indicates how the social practice ends, e.g., the morning routine takes around 45 min and ends when the newspaper is read and the coffee is finished.

Activities

- *Possible actions* describe the expected actions of actors in the social practice, e.g. making coffee, reading the newspaper, and opening curtains.
- *Requirements* indicate the type of capabilities or competences that the agent is expected to have in order to perform the activities within this practice. For example, the robot is expected to know how to make coffee and read the newspaper.

In [5] there is a first formalization of all these aspects based on dynamic logic. Due to space limitations we will not include this formalization here, but just discuss a few points that are important for the current implementation of social practices in Jason. The core element of the social practice for an agent is the plan pattern, which gives it handles to plan its behaviour. Plan patterns are parallel, choice or sequential combinations of plan parts expressed as $\gamma\phi$. These plan parts stand for all possible sequences of actions γ that contain actions contributing towards the achievement of ϕ (starting from a particular situation). ϕ is the *purpose* of that part of the practice. There can be more effects, but they are not all specified. So, in our morning routine practice, the plan pattern can be defined as $\gamma_1\phi_1; (\gamma_2\phi_2 \& \gamma_3\phi_3); \gamma_4\phi_4$, where ϕ_1 denotes the person being awake, ϕ_2 denotes the coffee being served, ϕ_3 denotes the pills being taken, and ϕ_4 denotes the person being mentally stimulated.

Thus, the purpose of the first part of the morning routine is that the person is awake. This might be done by opening the curtains, making a loud noise, or otherwise. If the purpose is achieved by opening the curtains, not only is the person awake, but the curtains are also open. The latter is merely a side effect of achieving the purpose.

Two more things should be noted about these patterns. One is that the overall pattern is supposed to achieve the overall purpose of the social practice. This is a formal constraint, but we only treat this implicitly. The other is that after a part of the plan pattern is finished, it automatically triggers the start of the next part of the pattern. In the full formalism this is assured, but is not explicit from only this fragment. In the same way, a social practice is started when the start condition becomes true. It then becomes available for execution and can be used by any agent present in the situation.

Finally, the formalism of social practices also guarantees that there is a common belief in the elements of the social practice and if actions are taken everyone has at least a common belief about the effects in as far as they are important for the social practice. Thus it guarantees a common situation awareness.

3 The Care Robot Scenario

In this section we elaborate on the care robot scenario outlined in the introduction, and describe how we have modelled and implemented it using Jason.

We assume the high-level operation of the robot is based on a BDI interpreter, and that it comes equipped with goals and plans to trigger and enact its care activities (most likely with some customisation of key parameters possible). In this section, we consider only a small subset of the robot's duties: to wake the patient at a certain time in the morning, to provide coffee as required, and to provide mental stimulation. We do not specify any goals of the robot outside the practice here, but normally the care robot would also have its own goals such as powering its battery, cleaning a room and taking care of the health of the patient.

Social practices provide patterns of coordination for multiple agents in terms of landmark states rather than explicit sequences of actions. Therefore they do not make limiting assumptions about the temporal aspects of actions and their effects leading up to a landmark. Only the landmarks themselves are explicitly temporally ordered. Monitoring of landmark states is necessarily decoupled from the performance of actions, as reaching a landmark may depend on another agent or agents, or may be the result of a delayed effect of an action. To provide a non-trivial test case, we include some temporal complexity in the scenario by including durative actions (i.e. those that take place across an interval of time), an action with a delayed effect, and a joint durative action, which has its desired effect only if two participants perform it during overlapping time intervals. Durative and joint actions are implemented using a Jason metainterpreter[2] that is described in Sect. 5. To simulate the passing of time, we use a "ticker" agent with a recursive plan that periodically performs a tick action to update the time recorded in the environment. We use Jason's synchronous execution mode, so the robot, patient and ticker agents perform a single reasoning cycle in every step of the simulation.

Listing 1 shows the robot's initial beliefs, rules and plans. The plans in lines 22–41 have declarative goals (i.e. their triggering goals express desired states) and use Jason preprocessing directives to transform them according to a predefined declarative achievement goal pattern [2].

The first set of plans (lines 22–26) is for achieving a state where the patient is awake, with alternative plans for talking to the patient, shaking him, and opening the curtains and waiting for the light to wake him. The *exclusive backtracking declarative goal* ("ebdg") pattern specifies that additional failure-handling logic should be added to ensure that all the plans will be tried (once each) until the goal is achieved, or all plans fail. Opening the curtains has a delayed effect: it will eventually wake the patient[3].

The second set of plans (lines 28–33) handles the goal of having the patient mentally simulated, and also uses the ebdg pattern. The first plan waits for the patient to be awake, and then fails so that the other plans will be tried. The other two alternatives involve playing the music of Mozart to the patient, and initiating the joint action of reading the newspaper with the patient. As joint actions are not directly supported by Jason, line 32 calls this action via a `solve` goal that is handled by our metainterpreter.

[2] A metainterpreter is a programming language interpreter written in the same, or a similar, language to the one being interpreted. It can be used to prototype extensions to the base language.

[3] Actions are implemented in Jason by defining an *execute* method in a Java class modelling the environment. The delay is currently hard-coded in this class.

```
1  /* Initial beliefs and rules */
2  durative(makePodCoffee).
3  durative(readNewspaper).
4  joint(readNewspaper).
5  durative_action_continuation_pred(readNewspaper, continueReadingNewspaper).
6  durative_action_continuation_pred(makePodCoffee, continueMakingPodCoffee).
7  durative_action_cleanup_goal(readNewspaper, cleanupReadNewspaper).
8  continueReadingNewspaper :-
9      started(readNewspaper, T1) &
10     not started_durative_action(readNewspaper, patient, _) &
11     time(T2) &
12     T2 <= T1 + 20.
13 continueReadingNewspaper :-
14     started_durative_action(readNewspaper, patient, _) &
15     not stopped_durative_action(readNewspaper, patient, _).
16
17 continueMakingPodCoffee :- state(coffee, not_made).
18
19 wake_up_phrase("Good morning sleepyhead!").
20
21 /* Plans */
22 {begin ebdg(state(patient,awake))}
23 +!state(patient,awake) : wake_up_phrase(P) <- talkToPatient(P).
24 +!state(patient,awake) <- shakePatient.
25 +!state(patient,awake) <- openCurtains; .wait(state(patient, awake), 30000).
26 {end}
27
28 {begin ebdg(state(patient,mentally_stimulated))}
29 +!state(patient, mentally_stimulated) <- .wait(state(patient, awake)); .fail.
30 +!state(patient, mentally_stimulated) <- play_mozart.
31 +!state(patient, mentally_stimulated) <-
32     !solve({ readNewspaper[participants([patient,robot])] }).
33 {end}
34
35 +!state(coffee, served) <- !state(coffee, made); serveCoffee.
36
37 {begin ebdg(state(coffee,made))}
38 +!state(coffee, made) : resource(coffee_pods) & resource(coffee_pod_machine) <-
39     makePodCoffee; .wait(state(coffee, made), 10000).
40 +!state(coffee, made) : resource(instant_coffee) <- makeInstantCoffee.
41 {end}
42
43 +performing_durative_action(Act, Agent, Time) :
44         not started_durative_action(Act, Agent, _) <-
45     // Cache percept as a belief
46     +started_durative_action(Act, Agent, Time).
47
48 +stopped_durative_action(Act, StoppedParticipant, Time)[source(percept)] <-
49     // Cache percept as a belief
50     +stopped_durative_action(Act, StoppedParticipant, Time);
51     -started_durative_action(Act, StoppedParticipant, Time).
52
53 +!cleanupReadNewspaper <-
54     -stopped_durative_action(readNewspaper, _, _)[source(self)].
55
56 { include("metainterpreter.asl") }
```

Listing 1. Plans for the care robot domain

These plans are followed by a single plan for serving coffee. This has the subgoal of having the coffee made, and then the action of serving the coffee is performed.

The fourth set of plans (lines 37–41) is triggered by the goal of reaching a state in which the coffee is made. The options are to use a coffee pod machine and wait for it to finish, or to make instant coffee.

The environment sends a percept to all participants of a joint action when any other participant performs the action for the first time or performs a stop action with the joint action as an argument. The remaining three plans handle receipt of these percepts, and a belief 'clean-up' goal that is created by the metainterpreter (if the agent declares that it has one—see line 7) when the agent stops performing a joint action.

The initial segment of the listing contains initial beliefs and rules related to the processing of durative actions: declarations of which actions are declarative and/or joint, and of predicates and associated rules defining the circumstances in which the robot will continue performing the durative actions.

In a real scenario, the patient will be a human, not a BDI agent, but we simulate the patient using a Jason agent. "He" (the patient agent) has a plan to take his pills once he is awake. He also has a plan that will respond to the robot beginning the joint newspaper reading action by also beginning that action. He will continue reading the newspaper for 40 time units if he is in a good mood, but only 20 if he is in a bad mood. Being woken by daylight (after the curtains are opened) leaves him in a good mood; being shaken awake leaves him in a bad mood, and talking will not wake him up. Thus, if the robot begins with goals to have the patient awake and mentally stimulated, the patient will be left in a bad mood by being shaken awake and the newspaper reading will be shorter (and less stimulating) than if he were in a good mood.

4 A Care Robot with Social Practices

Section 3 introduced the care robot scenario. In this section, we consider how the robot could be enhanced using social practices. We focus on the robot's awareness of a social practice's context, and its temporal structure as a partially ordered set of landmarks, each described in terms of a purpose and a sequence of actions to be performed[4].

As noted previously, it is assumed that the robot comes equipped with appropriate goals and plans, and that it is possible to customise certain parameters such as the time the user likes to wake up, and the time and style of coffee that he likes to have. However, customising each plan in isolation will not easily provide the coordination between activities and dynamic adaptability to different contexts that can be provided by social practices. To perform most effectively, the robot should choose, for a given context, the plans for each goal that will achieve the best outcomes for the patient, and furthermore, consider constraints on goal orderings that arise from preferences and habit. For example, if the patient prefers to be woken at a certain time in a given context (e.g. when his family is due to visit) and/or in a certain way (e.g. by the curtains being opened), his mood is likely to be adversely affected if he is woken at a different time, and his engagement with subsequent activities (such as reading the newspaper together) may be reduced. In this section we describe how this type of contextual information can be addressed by the use of a social practice.

In Sect. 3, we described the various plans and actions available to the robot. We now assume that the following "morning routine" social practice has emerged[5]. We present this as a set of beliefs in the form used by our social practice reasoning plans that will be

[4] Currently we only handle a single action for each landmark.

[5] It is beyond the scope of this paper to consider how social practices might be learned.

discussed in Sect. 5. Note that we only illustrate a small subset of what would be likely to be a real morning routine for a patient and his/her care robot, but this is sufficient to highlight the nature of social practices and their relation to BDI agents.

```
social_practice(morningRoutine,
    [state(location, home), resource(coffee_pods), resource(coffee_pod_machine),
     resource(pills), resource(newspaper_subscription),
     (time(T) & T < 1200)]).

landmark(morningRoutine, pa, [],
    [action(robot, openCurtains)], state(patient, awake)).

landmark(morningRoutine, pt, [pa],
    [action(patient, takePills)], state(pills, taken)).

landmark(morningRoutine, cs, [pa],
    [action(robot, makePodCoffee)], state(coffee, served)).

landmark(morningRoutine, ms, [pt,cs],
    [action([robot,patient], readNewspaper)], state(patient, mentally_stimulated)).
```

The first belief above encodes the name of the social practice and a list of conditions that must all hold for it to become active: there are constraints on the location, the resources available, and the time (here, the number 1200 is a proxy for some real-world time that ends the morning routine period).

The other four beliefs model the landmarks, specifying the social practice they are part of, an identifier for the landmark, a list of landmarks that must have been reached previously, a list of actions and their actors that are associated with the landmark, and finally, a goal that is the purpose of the landmark. The landmarks are: (1) to have the patient awake due to the robot opening the curtains, (2) for the patient to have taken his pills, (3) to have the coffee served, which should involve the robot making pod coffee, and (4) for the patient to be mentally stimulated due to the newspaper being read jointly. These landmarks are partially ordered with 1 before 2 and 3, which both precede 4.

Comparing this social practice to the robot plans shown in Listing 1, it can be seen that it avoids an ineffective attempt to wake the patient by talking to him, and prevents him from being left in a bad mood after being shaken awake. It agrees with the first-ordered plan for making coffee (by making pod coffee), and avoids an ill-fated attempt by the robot to provide mental stimulation by playing Mozart. Furthermore, it specifies an ordering on these activities that is not intrinsic to the plans themselves. Note also, that the social practice does not provide complete information on how to reach the landmark of having coffee served: it indicates that the robot should make pod coffee, but doesn't specify the action of serving the coffee. While a planning system could deduce the missing action using a model of actions and their effects [13], a BDI agent does not have this capability. Instead, a BDI agent using social practices must reason about how its existing plans could be used to satisfy landmarks given potentially incomplete information about the actions it must perform.

Furthermore, the robot may already have goals to wake the patient, provide mental stimulation, etc., and the activation of a social practice should not create independent instances of those goals. Thus, the activation of a social practice should override the agent's normal behaviour (for the relevant goals) during the period of activation.

As social practices are structured in terms of ordered landmarks, which model expected states to be reached in a pattern of inter-agent coordination, it is necessary for the agent to monitor the status of landmarks once their prior landmarks have been achieved, and to actively work towards the fulfilment of the current landmarks for which it has associated actions. In the next section, we present a meta-deliberation cycle for Jason agents that addresses this and the other issues outlined above, and which enables the successful execution of our care robot enhanced with social practices.

5 Implementation

5.1 Meta-level Reasoning About Social Practices

Maintaining awareness of social practices (SPs), and contributing to them in an appropriate way, requires agents to detect when each known social practice becomes active or inactive, to monitor the state of the landmarks in an active social practice, and to trigger the appropriate activity if an active SP has an action for the agent associated with the current landmark. This is a type of meta-level reasoning that the agent should perform periodically, and it may override the performance of any standard BDI processing of goals, which is not informed by social practices. We note that, on an abstract level, the same was done in [1] where the plan pattern was translated into a global pattern in Drools (Java based expert system) and the specific interactions within each phase were programmed in a chatbot.

The question then arises of how best to implement such a meta-level reasoner in a BDI architecture. The best performance can, no doubt, be achieved by extending a BDI platform using its underlying implementation language. However, it would require a change of the basic deliberation cycle to include not only reasoning about goals, plans and intentions, but also taking into account the social practice context. Thus, this approach requires significant knowledge of the implementation and requires using an imperative coding style that is not best suited to reasoning about goals [10] and for rapid prototyping and dissemination of new reasoning techniques. Therefore, in this work we define the meta-level reasoner as a plan for a `metadeliberate` goal that reasons about social practices, sleeps and then calls itself recursively. This, and some other plans it triggers, are shown in Listing 2. The plans make use of some extensions to Jason, handled by a metainterpreter that is described in the following subsection[6].

The social practice reasoner runs in response to the goal `metadeliberate` (line 10 in Listing 2). Lines 13 to 31 show the plan for this goal. The `atomic` annotation on the plan label ensures that steps of this plan are not interleaved with steps of other plans. The plan begins by (re)considering which social practice (if any) should be active. It uses the rules in lines 3 to 7 to find social practices that are relevant (i.e. all their requirements hold), and to select one (currently, the first option is always selected). If none are relevant (lines 17–20), any existing belief about the currently selected social practice is retracted. Otherwise (lines 21–29), if the selection has changed, the belief about the selection is updated. Any monitored landmarks are then checked to see if their purpose has been fulfilled (lines 26–28). If so, a belief about their completion is added. The plan then sleeps for a period, before triggering itself to be re-run in a new intention (lines 30–31). The new intention is needed for the recursive call because the plan is atomic, and the agent's other plans must be allowed to run.

[6] See https://github.com/scranefield/jason-social-practices for source code.

```
/* Rules */
// Omitted: has_plan_generating_action/3 and for_all/1
relevant_sp(SP) :-
    social_practice(SP, Requirements) & forall(Requirements).
sp_selection(Options, CurrentSP) :-
    selected_sp(CurrentSP) & .member(CurrentSP, Options).
sp_selection([SP|_], SP).

/* Initial goal */
!metadeliberate.

/* Plans */
@metaplan[atomic]
+!metadeliberate <-
    .findall(SP, ( relevant_sp(SP) & not completed_sp(SP) ),
             RelevantSPs);
    if (RelevantSPs == []) {
        if (selected_sp(CurrentlySelectedSP)) {
            -selected_sp(CurrentlySelectedSP)
        }
    } else {
        if ( sp_selection(RelevantSPs, SelectedSP) &
             not selected_sp(SelectedSP) ) {
           -+selected_sp(SelectedSP)
        }
        for (monitored(Purpose, SP, ID)) {
            if (Purpose) { +completed_landmark(SP, ID, Purpose) }
        }
    }
    .wait(500);
    !!metadeliberate.

+selected_sp(SP) <-
    for (landmark(SP, ID, _, _, Purpose)) {
        PurposeNoAnnots[dummy] = Purpose[dummy];
        if (.intend(PurposeNoAnnots)) {
             .suspend(PurposeNoAnnots);
             +suspended_intention(SP, ID, PurposeNoAnnots)
        }
        .add_plan({@suspend_purpose(SP,ID)
                   +!PurposeNoAnnots <- .suspend(PurposeNoAnnots)},
                  landmark(SP,ID), begin)
    }
    for (landmark(SP, ID, [], Actions, Purpose)) {
        !activate_landmark(SP, ID, Actions, Purpose)
    }.

@activate_landmark[atomic]
+!activate_landmark(SP, ID, Actions, Purpose) <-
    PurposeNoAnnots[dummy] = Purpose[dummy];
    +monitored(PurposeNoAnnots, SP, ID)
    if (Actions = [action(Actors, Act)] &
          (Actors = Me | (.list(Actors) & .member(Me, Actors)))) {
        if (has_plan_generating_action({+!Purpose}, Act, Path)) {
            !!solve({ !Purpose }, Path)
        } elif (joint(Act)) {
            !!solve({ Act[participants(Actors)] })
        } elif (durative(Act)) {
            !!solve({ Act })
        } else { Act }
    } else { .print("Multiple actions are not yet supported"); }.

```

Listing 2. Rules and plans for social practice reasoning

```
63 @completed_landmark[atomic]
64 +completed_landmark(SP, ID, Purpose) <-
65     .succeed_goal(Purpose);
66     -monitored(Purpose, SP, ID);
67     .remove_plan(suspend_purpose(SP,ID));
68     for ( landmark(SP, ID2, PrecedingLMs, Actions, Purpose2) &
69           not completed_landmark(SP, ID2, _) &
70           .findall(PrecID, (.member(PrecID, PrecedingLMs) &
71                             completed_landmark(SP, PrecID, _)),
72                    CompletedPrecIDs) &
73           .difference(PrecedingLMs, CompletedPrecIDs, []) ) {
74       !activate_landmark(SP, ID2, Actions, Purpose2)
75     }
76     .findall(ID2, ( landmark(SP, ID2, _, _,_) &
77                     not completed_landmark(SP, ID2, _) ),
78              PendingLandmarks);
79     if (PendingLandmarks == []) { +completed_sp(SP) }.
```

Listing 2. Rules and plans for social practice reasoning (continued)

A new belief about a selected social practice is handled by the plan in lines 33–46. This loops through the landmarks to check if the agent already has intentions to achieve any of their purposes[7]. If so, these intentions are suspended, and this is recorded in a belief so the intentions can be later marked as successful if the landmark is completed (see line 67). A plan is also temporarily added (lines 40–42) to ensure that if some other active plan of the agent separately creates this intention, it will be immediately suspended (the new plan is placed before any existing plans for that goal). For each landmark in the social practice that has no prior landmarks, a goal is created to activate it (lines 44–46).

Landmark activations are handled by the plan in lines 48–61. A belief recording that the landmark's purpose should be monitored is added, then the action associated with the landmark is processed (only a single action is supported currently). If the action is to be performed by the agent, three options are considered. First (line 54), a query is made to find a solution for achieving the landmark's purpose that involves performing the specified action. A set of rules (not shown) handle this query by searching for the action recursively (up to a prespecified depth bound) through the plans that achieve the purpose, and the subgoals in those plans, and so on[8]. Context conditions are checked for the top level plans (those for the landmark's purpose), but not for the recursive calls, as, in general, it cannot be known how the state of the world will change as these plans are executed. If such a solution is found, it is recorded as a goal-plan tree "path" (see Sect. 5.2) and passed to our Jason metainterpreter via a `solve` goal (line 55). If no such solution is found, and the action is joint, or durative but not joint, the metainterpreter is called to handle this (lines 57 and 59). Otherwise, the action is performed directly (line 60). Note that this plan is declared to be executed atomically (line 48). This prevents steps of the agent's other plans from being interleaved with this one, and thus ensures the landmark is activated promptly. To ensure that other plans can run again once the correct course of action has been identified by this plan, the calls to the metainterpreter are created as a separate intention (using "`!!`").

[7] The unifications in lines 35 and 50 instantiate the variable on the left with the value of the variable on the right, but with any Jason annotations removed.

[8] At present we assume that the plan body will contain only one achievement subgoal.

```
// Solve plan body, with optional path through the goal-plan tree
// Example goal: !solve({action; ?test(X); !g})
+!solve(PlanBody) <- !solve(PlanBody, no_path).

+!solve({}, _).
+!solve({BodyTerm;BodyTerms}, Path) <-
    !solve_bt(BodyTerm, Path);
    !solve(BodyTerms, no_path). // Path is applied to first body term only

// Solve body term
+!solve_bt(G, _)
  : .member(G, [test(_), addBel(_), delBel(_), internalAction(_)]) <- G.
+!solve_bt(achieve(solve(PB)), Path) <- !solve(PB, Path).
+!solve_bt(achieve(G), Path) <-
    .relevant_plans({+!G}, RPlans);
    if (.list(Path) & Path = [N|PathTail]
                    & .nth(N, RPlans, plan(Label,_,Context,PlanBody))
                    & Context                                        ) {
        !solve(PlanBody, PathTail);
    } else {
        !applicable_plans(RPlans, [plan(Label,_,_,PlanBody)|_]);
        !solve(PlanBody, Path);
    }.
// Handle durative action
+!solve_bt(BT, _) :
        BT = action(A) & NoAnnotsA[dummy] = A[dummy] & durative(NoAnnotsA) <-
    ?durative_action_continuation_pred(NoAnnotsA, Query);
    if (durative_action_cleanup_goal(NoAnnotsA, CleanupGoal)) {
        CUGoal = CleanupGoal;
    } else {
        CUGoal = true;
    }
    if (joint(NoAnnotsA) & NoAnnotsA[participants(P)] = A) {
        ParticipantsAnnot = [participants(P)];
    } else {
        ParticipantsAnnot = [];
    }
    AnnotatedA = NoAnnotsA[durative|ParticipantsAnnot];
    .term2string(AnnotatedA, S);
    .concat("{", S, "}", BodyTermString);
    .term2string(AnnotatedBT, BodyTermString);
    AnnotatedBT;
    ?time(T); // Must be supplied as a percept from the environment
    -+started(NoAnnotsA, T)[source(meta)];
    !solve_durative(Query, AnnotatedBT, NoAnnotsA, ParticipantsAnnot, CUGoal).
// Handle non-durative action
+!solve_bt(ActionGoal, _) : ActionGoal = action(_) <- ActionGoal.

+!solve_durative(Query, ActionBT, Action, ParticipantsAnnot, CleanupGoal) <-
    if (Query) {
        ActionBT;
        !solve_durative(Query, ActionBT, Action, ParticipantsAnnot, CleanupGoal);
    } else {
        stop(Action)[durative|ParticipantsAnnot];
        if (CleanupGoal \== true) { !CleanupGoal; }
    }.
-!solve_durative(_, _, Action, _, _) <- -started(Action, _).

@applicable_plans[atomic]
+!applicable_plans([], []).
+!applicable_plans([P|T], [P|T2]) : P = plan(_,_,C,_) & C <-
    !applicable_plans(T, T2).
+!applicable_plans([P|T], T2) : P = plan(_,_,C,_) & not C <-
    !applicable_plans(T, T2).
```

Listing 3. A Jason metainterpreter

Finally, the plan in lines 63–79 handles completed landmarks—those for which the purpose has been achieved. Any suspended intentions for the purpose are succeeded, the belief stating that the landmark should be monitored is retracted, and the temporary plan added in lines 40–42 is removed. The plan then checks for subsequent landmarks that should now be activated (if all their prior landmarks are completed), and finally adds a belief that the social practice has completed if all its landmarks are completed. Another plan (not shown) is needed to handle social practices that become inactive when their relevance conditions cease to hold. In this case, any active landmarks should be abandoned, and original intentions to achieve their purposes can be resumed.

5.2 A Jason Metainterpreter

Listing 3 shows our Jason metainterpreter[9], which extends the AgentSpeak metainterpreter defined by Winikoff [19], and specialises it for use with Jason. The metainterpreter is initiated by calling a `solve` goal with a Jason plan body $\{g_1, \cdots, g_n\}$ as an argument, where the g_i may be of the various types of goals and actions that Jason supports, such as test goals (queries to the belief base), belief additions and deletions, achievement goals that trigger plans, and actions that may be internal (built-in or user-defined) or external (defined by the environment). An additional `Path` argument, explained below, may also be supplied. The plans in lines 5–8 sequentially create `solve_bt` subgoals for each body term ("bt" for short) in the plan body, and these are handled by the plans in lines 11–47. For test goals, belief additions and deletions, and internal and (standard) external actions, lines 11–12 and 47 call these body terms directly to invoke the standard Jason interpreter. Line 13 ensures that the metainterpreter can handle plans containing `solve` subgoals to explicitly make use of the meta-interpreter's extensions. The remaining plans provide extended BDI semantics, as outlined below.

– As explained in Sect. 5.1, when a landmark in a social practice includes an action associated with the current agent, the plan to activate a landmark attempts to find an existing set of plans that can achieve the landmark's purpose while also including the specified action. This is a recursive search through plans and their subgoals, and it results in a pre-selected path through the goal-plan tree [11] representing the search space for satisfying the landmark's purpose. This differs from the standard Jason goal execution mechanism, in which a set of *relevant* plans (those triggered by a goal matching the current one) is determined, these are filtered by evaluating the plans' context conditions to produce the *applicable* plans, and one is selected to be executed (the first-listed one, by default). This process is then repeated for each subgoal appearing in the body of a plan being executed. This standard process will not guarantee that the landmark's associated action will be performed. Thus, the metainterpreter allows a predetermined goal-plan tree path to be passed as an optional argument to a `solve` goal (see line 55 in Listing 2), to guide it directly to the pre-chosen subplans, and eventually the desired action. This feature is useful for plan pre-selection in other meta-reasoning contexts as well, e.g. choosing plans based on their effect on the values of a human user [4]. If no path is provided as

[9] The metainterpreter requires version 2.4 of Jason.

an argument to solve, a default value of no_path is used (line 3 of Listing 3). When a path is provided, lines 16 to 19 select the requested plan for the current goal, if its context condition holds, and call its body via solve. Otherwise, the list of applicable plans (those whose context conditions are true) are computed (lines 21 and 59–64[10]), the first (in order of appearance in the agent's code) is chosen (second argument in line 21), and its plan body is called via solve (line 22). If there are no applicable plans, this plan fails, as does the original solve goal.

– Durative actions, as required by our scenario, are supported (lines 25–45 and 49–57)[11]. A continuation predicate, and optionally a clean-up goal, for the action are looked up (lines 27–32), the time the action was started is recorded as a belief (lines 43–44), and a solve_durative goal is created (line 45) to trigger the performance of the action. The plan for this goal (lines 49–56) checks the continuation condition (passed as variable Query). It is intended that the query is a 0-arity predicate defined by a rule in the agent's program. If the query succeeds, the action is executed with a "durative" annotation (which the environment should check for), and possibly an annotation listing the action participants if it is a joint action (see below). The goal is then called recursively. If the query fails, stop(Act) is executed (again, with the appropriate annotations). Thus, durative actions are implemented by repeated execution of an action until the corresponding stop action is called.
– Joint actions are also supported. These are durative actions with an annotation listing the intended action participants. The environment should notify all intended participants (via a percept) when a durative action is called for the first time or is stopped, thus enabling the participants to coordinate their actions. It should also keep a history of the time intervals over which the participants perform the action, as its outcome will depend on the existence and length of a period of overlap. Lines 33–34 and 38–42[12] ensure that an action term annotation recording the participants is passed on when starting the action, and line 54 passes the annotation on when performing the stop action.

6 Evaluation

We ran our Jason care robot program, with and without the social practice beliefs and meta-deliberation plan, using a Jason agent to simulate the patient (as outlined at the end of Sect. 3). With the social practice support for the care robot agent, the robot and patient agent could more successfully coordinate their actions across the landmarks of the social practice, ensuring that the patient remains in a good mood, and engages in the newspaper reading for longer (compared to the outcome outlined in the last paragraph

[10] Lines 59–64 would be better implemented as Jason rules (Horn clauses), but passing plan bodies to Jason rules via test goals currently causes the goals to fail, even when the rules succeed.

[11] The second context condition in line 26 binds NoAnnotsA to the action term A with annotations removed.

[12] Lines 39 to 41 are a workaround for a current limitation of Jason: annotations cannot be added to an existing plan body term.

of Sect. 3). Of course, the scenario was designed to produce better outcomes when the social practice is followed, but demonstrating this in practice validates our metadeliberation plan and metainterpreter.

In this paper, we aim to show the promise of using social practices as part of a meta-deliberation cycle for BDI agents in order to cope with real time environments. The main characteristic of using this approach is that we can separate different concerns. One is to deal with a context of an interaction and its consequences for the plans and actions of the agent; the other is to deal with plans within a certain context. It follows from the above that we do not really extend the language or intrinsic capabilities of the agent (as we have shown, the meta-deliberation cycle can indeed be written using Jason rules and plans). Rather we claim that using this separation of concerns will make it more scalable and robust. These are hard properties to prove or evaluate directly.

What would be needed for an evaluation are complex and dynamic scenarios where the agent is interacting with several people and other agents in different roles, and where events in the environment can also influence the interaction. We can then show that in these complex scenarios the agents based on social practices are still programmed in a modular way, with limited complexity, and behaving robustly. Ideally one would show how Jason BDI agents would have to be programmed without the social practices in the meta-deliberation and compare that with a Jason agent including the social practice in the meta-deliberation. That would give some indication of the utility of our architecture. However, one could still object that the specific agent programs should be programmed by people with the same background, using the same information and measuring the time to develop the agents, etc., in order to make a fair comparison.

As first steps in this evaluation process we will develop more complex scenarios for social robots that will act in a physical context in order to evaluate the robustness and efficiency of our approach.

7 Conclusions

We have argued that for interactive settings, as sketched in our scenario, the use of social practices is a good compromise between using a fixed interaction protocol and deliberation and planning from scratch at each point during the interaction. We proposed a mechanism for a BDI agent to maintain awareness about and contribute towards the completion of social practices, and presented this as a meta-deliberation plan for Jason agents. We also presented a Jason metainterpreter to support this plan and our care robot scenario. These contributions provide a specification of potential extensions to the BDI reasoning cycle, but also allow the approach to be directly applied within Jason agents.

Our approach allows BDI agents to use their existing plans to achieve social practice landmarks that do not detail all actions required to achieve the landmark. In future work we intend to investigate more complex interactions between social practices and agent's local plans. We also intend to develop elaborate scenarios that use all aspects of a social practice, and compare these with agent implementations where no social practice is used, both in terms of the outcomes of the agent and the ease of design of the agents.

Acknowledgement. We thank Jomi Hübner for making extensions to Jason in support of the Jason metainterpreter presented in Listing 3.

References

1. Augello, A., Gentile, M., Dignum, F.: Social practices for social driven conversations in serious games. In: de De Gloria, A., Veltkamp, R. (eds.) GALA 2015. LNCS, vol. 9599, pp. 100–110. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40216-1_11
2. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. Wiley, Chichester (2007)
3. Bourdieu (trans. R. Nice), P.: Outline of a Theory of Practice. Cambridge University Press, Cambridge (1972)
4. Cranefield, S., Winikoff, M., Dignum, V., Dignum, F.: No pizza for you: value-based plan selection in BDI agents. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 178–184. ijcai.org (2017)
5. Dignum, F.: Interactions as social practices: towards a formalization. arXiv (2018). https://arxiv.org/abs/1809.08751
6. Dignum, V., Dignum, F.: Contextualized planning using social practices. In: Ghose, A., Oren, N., Telang, P., Thangarajah, J. (eds.) COIN 2014. LNCS (LNAI), vol. 9372, pp. 36–52. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25420-3_3
7. Folsom-Kovarik, J., Schatz, S., Jones, R.M., Bartlett, K., Wray, R.E.: AI challenge problem: scalable models for patterns of life. AI Mag. **35**(1), 10–14 (2014)
8. Giddens, A.: Central Problems in Social Theory: Action, Structure and Contradiction in Social Analysis. University of California Press, Berkeley (1979)
9. Harel, R., Yumak, Z., Dignum, F.: Towards a generic framework for multi-party dialogue with virtual humans. In: Proceedings of the 31st International Conference on Computer Animation and Social Agents, CASA 2018, pp. 1–6. ACM, New York (2018)
10. Logan, B.: An agent programming manifesto. Int. J. Agent-Oriented Softw. Eng. **6**(2), 187–210 (2018)
11. Logan, B., Thangarajah, J., Yorke-Smith, N.: Progressing intention progression: a call for a goal-plan tree contest. In: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, pp. 768–772. IFAAMAS (2017)
12. Mercuur, R., Dignum, F., Kashima, Y.: Changing habits using contextualized decision making. In: Jager, W., Verbrugge, R., Flache, A., de Roo, G., Hoogduin, L., Hemelrijk, C. (eds.) Advances in Social Simulation 2015. AISC, vol. 528, pp. 267–272. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47253-9_23
13. Miller, T., Dignum, V., Dignum, F.: Planning for human-agent collaboration using social practices. In: First International Workshop on Socio-cognitive Systems at IJCAI 2018 (2018)
14. Minsky, M.: A framework for representing knowledge. In: Smith, A., Collins, E. (eds.) Readings in Cognitive Science, pp. 156–189. Morgan Kaufmann (1988)
15. Narasimhan, K., Roberts, T., Xenitidou, M., Gilbert, N.: Using ABM to clarify and refine social practice theory. In: Jager, W., Verbrugge, R., Flache, A., de Roo, G., Hoogduin, L., Hemelrijk, C. (eds.) Advances in Social Simulation 2015. AISC, vol. 528, pp. 307–319. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47253-9_27
16. Reckwitz, A.: Toward a theory of social practices. Eur. J. Soc. Theory **5**(2), 243–263 (2002)
17. Schatzki, T.R.: A primer on practices. In: Practice-Based Education, Practice, Education, Work and Society, vol. 6, pp. 13–26. SensePublishers, Rotterdam (2012)
18. Shove, E., Pantzar, M., Watson, M.: The Dynamics of Social Practice. Sage, Thousand Oaks (2012)
19. Winikoff, M.: An AgentSpeak meta-interpreter and its applications. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) ProMAS 2005. LNCS (LNAI), vol. 3862, pp. 123–138. Springer, Heidelberg (2006). https://doi.org/10.1007/11678823_8