Making Robots Proactive through Equilibrium Maintenance

Jasmin Grosinger, Federico Pecora, Alessandro Saffiotti
Center for Applied Autonomous Systems (AASS),
Örebro University, 70182 Örebro, Sweden,
{jngr,fpa,asaffio}@aass.oru.se

Abstract

In order to be proactive, robots should be capable of generating and selecting their own goals, and pursuing activities towards their achievement. Goal reasoning has focused on the former set of cognitive abilities, and automated planning on the latter. Despite the existence of robots that possess both capabilities, we lack a general understanding of how to combine goal generation and goal achievement. This paper introduces the notion of *equilibrium maintenance* as a contribution to this understanding. We provide formal evidence that equilibrium maintenance is conducive to proactive robots, and demonstrate our approach in a closed loop with a real robot in a smart home.

1 Introduction

The aim of this paper is to investigate the general problem of making robots proactive. By proactive we mean a robot that is able to generate its own goals and pursue activities towards their achievement. Consider the following example. Anna, who owns a robot, has instructions from her physician to take pills daily at meal times. Not taking the pills can result in Anna being unwell. During the day, there are moments in which it would be beneficial for the robot to *proactively* prompt Anna to take her pills, and moments in which it would not be. For instance, it would be pedantic to remind her of the pills at breakfast, but it may be adequate at lunch and really needed at dinner. If she goes to bed without taking taking her pills, the robot may become more invasive and bring the pills to her. It should also bring the pills at breakfast, if it knows that Anna will be out for the rest of the day.

The decision on when to act and what to do must consider, in general, a multitude of aspects related to the current and future state of the whole system, including the robot, the user, and the pills. An important capability of a proactive robot is planning, i.e., the ability of finding a way to achieve a certain goal, given a starting situation. In the field of service robotics, for example, a considerable amount of work has been done in using planning to satisfy goals manually provided by a user [Hertzberg and Chatila, 2008]. However, we strongly agree with the view expressed by [Pollack and Horty, 1999], who state a need for deliberation functions in intelligent agents that go beyond planning. This view is echoed by [Ghallab *et al.*, 2014], who call for robots that, in the first place, are actors that "may use planning and other deliberation tools, before and during acting".

One of these additional deliberation tools is goal autonomy, that is, the ability to autonomously formulate and dispatch the goals that should be achieved, rather than receiving these goals as input. Goal autonomy enables robots to proactively support humans, a capability that has been shown to enhance the effectiveness of human-robot cooperation [Zhang *et al.*, 2015]. The field of goal reasoning is concerned with bringing about this capability [Aha, 2015]. Surveys on the topic have identified requirements and created designs for goal autonomous agents [Hawes, 2011; Beaudoin, 1994; Vattam *et al.*, 2013]. Several cognitive architectures [Rao and Georgeff, 1991; Anderson *et al.*, 2004; Laird *et al.*, 1987; Cox and Oates, 2013] generate goals by encoding internal agent drives and maintaining belief states. Other approaches use "anomalies" in the environment as a condition that triggers the generation of new goals [Cox, 2007; Molineaux *et al.*, 2010; Galindo and Saffiotti, 2013].

In line with existing work [Hawes, 2011; Beaudoin, 1994], in this paper we identify the need for a process that is independent of planning to achieve goal autonomy, where goals are not provided by a user, an external agent or a hard-coded condition in the environment. Solutions for goal autonomy have been developed to address specific application domains [Cox, 2007; Molineaux *et al.*, 2010; Hanheide *et al.*, 2010; Galindo and Saffiotti, 2013], however we lack a general understanding of how to combine goal generation and goal achievement. This paper proposes a general formalization and computational framework that can be deployed in real robotic systems, based on the notion of *equilibrium maintenance*.

We first define the notion of *opportunity* as a way to relate current and future states, courses of action and desirable states: an action is an opportunity if it keeps the system in desired states, and if the robot has the ability to enact it. Second, we define *equilibrium*, i.e., the absence of opportunities, as the persistent meta-goal which our system should aim to maintain. Third, we provide an *equilibrium maintenance* algorithm, that closes the loop between desirable states and plan execution by continuously evaluating potential opportunities, deciding which ones to act upon. Finally, we show that equilibrium maintenance is conducive to proactive robots via formal and empirical evidence.

2 Formalizing Equilibrium

Let $\mathcal{L}$ be a finite set of predicates. We consider a system $\Sigma = \langle S, U, f \rangle$, where $S \subseteq \mathcal{P}(\mathcal{L})$ is a set of states, U is a finite set of external inputs (the robot's actions), and $f \subseteq S \times U \times S$ is a state transition relation. Each state $s \in S$ is completely determined by the predicates that are true in s (closed world assumption). If there are multiple robots, we let U be the Cartesian product of the individual action sets, assuming for simplicity synchronous operation. The f relation models the system's dynamics: $f(s, u, s')$ holds iff Σ can go from state s to s' when the input u is applied. We assume discrete time, and that at each time t the system is in one state $s \in S$. In our pills example, the states in S may encode the time of day or the current activity of the user.

The free-run behavior F^k of Σ determines the set of states that can be reached from s in k steps when applying the null input $\perp$, that is, the natural evolution of the system when no robot actions are performed. F^k is given by:

$$F^0(s) = \{s\}$$
$$F^k(s) = \{s' \in S \mid \exists s'' : f(s, \perp, s'') \wedge s' \in F^{k-1}(s'')\}.$$

We consider a set *Des* $\subseteq S$ and a set *Undes* $\subseteq S$ meant to represent the *desirable* and *undesirable* states in S. For instance, a state where the user is having lunch and the pills are taken is in *Des*, whereas a state where the day is over and the pills have not been taken is in *Undes*. For the time being, we assume that *Des* and *Undes* form a partition of S.

2.1 Action schemes

We want to capture the notion that Σ can be brought from some states to other states by applying appropriate actions in the appropriate context. We define an *action scheme* to be any partial function

$$\alpha : \mathcal{P}(S) \to \mathcal{P}^+(S),$$

where $\mathcal{P}^+(S)$ is the powerset of S minus the empty set. We denote by A the set of all action schemes. An action scheme α abstracts all details of action: $\alpha(X) = Y$ only says that there is a way to go from any state in X to some state in Y. We denote by *dom*(α) the domain where α is defined. For example, the scheme α_{remind}, which reminds the user to take the pill, can be applied in any state s where the user is present and the robot is on: these conditions characterize *dom*(α_{remind}). Action schemes can be at any level of abstraction, be it simple actions that can be executed directly, sequential action plans, policies, high level tasks or goals for one or multiple planners.

Figure 1 illustrates the above elements. Each action scheme can be applied in some set of states and brings the system to other states. For instance, scheme α_1 can be applied to any state $s' \in X_1$, and when applied it will bring Σ to some new state $s'' \in Y_1$. The system is currently in the desirable state s, and if no action is applied it will move in k steps to some state in the set $F^k(s)$, which is undesirable.

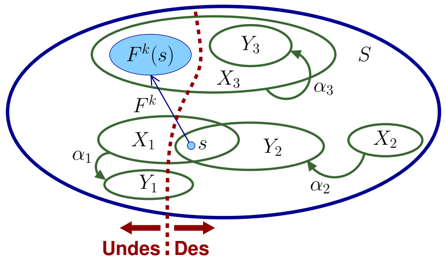

Figure 1: How α's and F change the state of the system.

We now define what it means for an action scheme α to be *beneficial* in a state s:

$$\textit{Bnf}(\alpha, s) \text{ iff } \exists X \in \textit{dom}(\alpha) \text{ s.t. } s \in X \wedge \alpha(X) \subseteq \textit{Des}.$$

Intuitively, α is beneficial in s if it is applicable in s, and if applying it will necessarily result in a desirable state. In Figure 1, α_1 is applicable in s since its domain includes X_1 and $s \in X_1$. However, it is not beneficial in s since it does not bring the system into states which are all desirable, i.e., $\alpha_1(X_1) = Y_1$ and $Y_1 \not\subseteq$ *Des*. Scheme α_2 would be beneficial in another state, but it is not applicable in s. Scheme α_3 is not beneficial in s, but it will become so in k steps.

We can extend the notion of being beneficial to take a time horizon k into account:

$$\textit{Bnf}(\alpha, s, k) \text{ iff } \exists X \in \textit{dom}(\alpha) \text{ s.t. } s \in X \wedge F^k(\alpha(X)) \subseteq \textit{Des},$$

where $F^k(X) = \cup_{s \in X} F^k(s)$. Intuitively, such a scheme is a way to bring the system from the current state to a state that will be desirable after k time steps. Note that *Bnf*$(\alpha, s, 0) =$ *Bnf*(α, s). One may also define a durative version in which all future states up to k are desirable.

2.2 Opportunities

We can use the above apparatus to characterize different types of opportunities for action, which we define formally here and exemplify in Section 5. Let $s \in S$ and let $k \in \mathbb{N}$ be a finite time horizon. We define six properties that determine whether α is an opportunity for acting in s:

$$\textit{Opp}_1(\alpha, s, k) \text{ iff } s \in \textit{Undes} \wedge \big(\exists s' \in F^k(s) : \textit{Bnf}(\alpha, s')\big)$$
$$\textit{Opp}_2(\alpha, s, k) \text{ iff } s \in \textit{Undes} \wedge \big(\forall s' \in F^k(s) : \textit{Bnf}(\alpha, s')\big)$$
$$\textit{Opp}_3(\alpha, s, k) \text{ iff } \exists s' \in F^k(s) : (s' \in \textit{Undes} \wedge \textit{Bnf}(\alpha, s'))$$
$$\textit{Opp}_4(\alpha, s, k) \text{ iff } \forall s' \in F^k(s) : (s' \in \textit{Undes} \to \textit{Bnf}(\alpha, s'))$$
$$\textit{Opp}_5(\alpha, s, k) \text{ iff } \big(\exists s' \in F^k(s) : s' \in \textit{Undes}\big) \wedge \textit{Bnf}(\alpha, s, k)$$
$$\textit{Opp}_6(\alpha, s, k) \text{ iff } \big(\forall s' \in F^k(s) : s' \in \textit{Undes}\big) \wedge \textit{Bnf}(\alpha, s, k)$$

The first two properties characterize schemes that can be *applied in the future* in response to a *current* undesired situation. In particular, *Opp*$_1(\alpha, s, k)$ says that s is an undesirable state for Σ, and that if no action is taken Σ *may* evolve in a state s' in which action scheme α is beneficial — that is, α can be applied in s' to bring the system into a desirable state. *Opp*$_2(\alpha, s, k)$ is the same except that Σ *will* evolve in a state in which α is beneficial. In Figure 1 above, α_3 is an opportunity of this type in s. The third and fourth properties characterize schemes that can be *applied in the future* in response to either a *foreseen* or *possible* undesired situation. The last two properties characterize schemes that can be *applied now* in order to prevent *future* undesired situations. Note that for $k = 0$ all the above properties collapse to

$$\mathit{Opp}(\alpha, s, 0) \text{ iff } s \in \mathit{Undes} \wedge \mathit{Bnf}(\alpha, s),$$

that is, α can be used *now* to resolve a *current* threat. Henceforth, we indicate this opportunity type with Opp_0.

2.3 Equilibrium

Using the above ingredients, we can now define equilibrium as a property Eq of a system Σ in a state s. Given a time horizon $K \in \mathbb{N}$, we say that Σ *is in equilibrium* in a given state s, denoted by $Eq(s, K)$, if there is no incentive to act in the next K steps. That is, no action scheme α can be inferred as being an opportunity given s and given any $k \leq K$:

$$Eq(s, K) \text{ iff for all } k = 0, \ldots, K, \text{ for all } i = 0, \ldots, 6, \\ \nexists \alpha \in A : \mathit{Opp}_i(\alpha, s, k)$$

3 Equilibrium Maintenance

Our next aim is to operationalize the notion of proactivity by defining it in terms of a robot's ability to maintain equilibrium. We start by introducing our assumptions.

The set $A = \{\alpha_1, \ldots, \alpha_m\}$ of action schemes is assumed to be finite. Recall that an action scheme may denote anything from a high-level goal to an individual action. We assume the existence of a suitable execution layer that generates and dispatches appropriate robot actions for executing a given action scheme. We assume to have time models of all entities that affect the state of the system. This assumption is not restrictive, since time models may have non-determinism to account for lack of knowledge — in the extreme case, a "fully ignorant" time model would allow all transitions to happen.

We define the process of Equilibrium Maintenance in terms of three fundamental capabilities: (i) to *determine* if the system is in equilibrium, that is, whether there exist opportunities for acting; (ii) to *select* one of the found opportunities; and (iii) to *dispatch* the selected opportunity for execution.

Figure 2 summarizes how Equilibrium Maintenance is assumed to work for a system Σ given an appropriate State Estimation function and an appropriate Plan-Based Executive for action scheme execution. Note that Equilibrium Maintenance realizes a feedback loop at a higher level of abstraction than the Plan-Based Executive: the former closes the loop around action scheme selection, execution, and state estimation; the latter determines how given action schemes are executed by the system. The level of abstraction at which Equilibrium Maintenance operates is determined by the models used to characterize the system's possible states, action schemes, and free-run. This typically differs from that of the Plan-Based Executive: The α action schemes, output of Equilibrium Maintenance in Figure 2, can be individual actions, plans, policies, or planning goals, as mentioned in Section 2. Correspondingly, the Plan-Based Executive can range from a

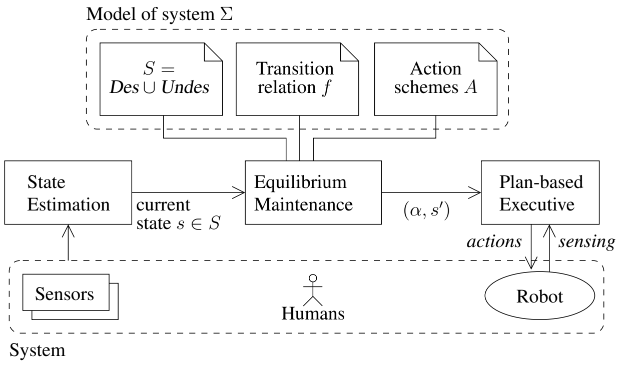

Figure 2: The Equilibrium Maintenance loop (realized by the $EqM(K)$ algorithm) for a system $\Sigma = \langle S, U, f \rangle$, where action schemes A are defined in terms of the robot's actions U.

simple dispatcher to a full plan-based robot controller. If α is a plan, executing an opportunity means to dispatch the actions in α. If α is a goal, it means to generate and execute a plan for α. In the experimental system in Section 5, α are plans from a plan library. No matter which level of abstraction is chosen, the Plan-Based Executive makes available a procedure that results in the physical execution of an action scheme α. Also, since Equilibrium Maintenance may determine an α that should be applied in a future state s' (this is the case for $\mathit{Opp}_{1\text{–}4}$), this state is also passed to the Plan-Based Executive.

```
1 while true do
2     s ← current state
3     if s has changed then
4         OppQueue ← ∅
5         for k from 0 to K do
6             OppQueue ← OppQueue ∪ find_opp (s, k)
7         if ¬Eq(s, K) then
8             ⟨α, s′⟩ ← select (OppQueue)
9             dispatch (α, s′)
```

Algorithm 1: $EqM(K)$

Algorithm 1 describes the $EqM(K)$ procedure which realizes the Equilibrium Maintenance loop. It finds existing opportunities in the current state s (lines 5–6) with incremental time horizons k to categorize each $\alpha_i \in A$ into opportunity types $\{\mathit{Opp}_0, \ldots, \mathit{Opp}_6\}$. `find_opp`($s$,$k$) returns each action scheme that constitutes an opportunity in state s with time horizon k, together with the state where it can be applied, its opportunity type and its time horizon, as a tuple $\langle \alpha, s', \mathit{Opp}_i, k \rangle$. *OppQueue* collects all such action schemes for all $k \leq K$. If *OppQueue* is not empty, then the system is not at equilibrium (line 7) and an action scheme is selected and dispatched (lines 8–9).

$EqM(K)$ is driven by state change (line 3), hence opportunities for acting are assessed only if the state changes as a result of free-run or the execution of an action scheme α. Whether all opportunities for action can be seized depends on the synchronization between the Equilibrium Maintenance loop and the control loop of the Plan-Based Executive. If no assumption is made on the synchronization between the two loops, it is important to dispatch action schemes that are beneficial in a future state s' (opportunities of type *Opp*$_{1\text{–}4}$), as opportunity evaluation may not occur when s' becomes the current state. There might be additional reasons to re-evaluate opportunities beyond the fact that the state has changed. This is the case, for example, if the execution of α deviates from expectations, or if the set *Des* changes. While these are important issues to investigate, in our current experiments we make the common assumptions that *Des* is fixed and that action scheme execution and free-run of the system do not co-occur, and henceforth we simply use $EqM(K)$ shown above.

Among the found opportunities, one is selected for dispatching to the Plan-Based Executive via procedure `select` (line 8). Selecting an opportunity may depend on the specific action scheme, on the type of opportunity, and on the time horizon by which the opportunity's action scheme will have an effect. How to perform this selection is a key question for equilibrium maintenance. In this paper, `select` selects the opportunity whose type is highest according to a given partial order $>_{\text{greedy}}$ on opportunity types. $>_{\text{greedy}}$ prioritizes opportunities for acting now over opportunities for acting in a future state, that is, $\mathit{Opp}_0 >_{\text{greedy}} \{\mathit{Opp}_5, \mathit{Opp}_6\} >_{\text{greedy}} \{\mathit{Opp}_1, \mathit{Opp}_2\} >_{\text{greedy}} \{\mathit{Opp}_3, \mathit{Opp}_4\}$. Ties are broken using the value of k: if there are two opportunities $\langle \alpha', s', \mathit{Opp}_i, k' \rangle$ and $\langle \alpha'', s'', \mathit{Opp}_j, k'' \rangle$, such that Opp_i and Opp_j are not comparable according to $>_{\text{greedy}}$, then we select the one with the lower k. If $k' = k''$ we select one randomly.

In the next section we show that $>_{\text{greedy}}$ guarantees that $EqM(K)$ always leads to states in *Des*, subject to reasonable assumptions on the starting state(s) and the time horizon K.

4 Properties of Equilibrium Maintenance

In this section, we analyze the behavior of $EqM(K)$. We shall show that this algorithm allows us to keep the given system Σ within *Des*, provided that suitable action schemes are available.

We proceed in three steps. First, we characterize the subset of S from where one can reach *Des* through an adequate choice of action schemes. We denote the states that can possibly result from applying α in s by $\alpha(s) = \{s' \in S \mid \exists X \in dom(\alpha) \text{ s.t. } s \in X \wedge s' \in \alpha(X)\}$. For any $C \subseteq S$, we define the *controllable neighborhood* $N(C)$ of C as

$$N(C) = \{s \in S \mid \exists \alpha \in A \text{ s.t. } \alpha(s) \subseteq C\}.$$

Intuitively, $N(C)$ is the set of states from where some α is available that guarantees to reach C. For any $C \subseteq S$, we define the *basin of attraction* of radius r of C, with $r \geq 0$, as

$$Bas^r(C) = \{s \in S \mid \exists r' \leq r \text{ s.t. } F^{r'}(s) \subseteq C\}.$$

$Bas^r(C)$ is the set of states from which the system will go into C by free-run within r steps. Finally, we define the set $Rec(\Sigma, r)$ of *recoverable states* with radius r as follows:

$$Rec(\Sigma, r) = N(Bas^r(\mathit{Des})) \cup Bas^r(N(\mathit{Des})).$$

Intuitively, the set $Rec(\Sigma, r)$ are those states from which one can reach *Des* given the available α's. More specifically, $s \in Rec(\Sigma, r)$ means that: (1) one can apply an α in s, which will bring Σ into *Des* within r steps; or (2) within r steps one can apply an α, which will bring Σ into *Des*. Note that $N(\mathit{Des}) \subseteq Rec(\Sigma, r)$ for any r, that is, s is always in $Rec(\Sigma, r)$ if there is an α that can bring Σ from s into *Des*.

Second, we characterize the ways in which Σ can evolve under the effect of a given strategy. We call *action selection function* any function $\sigma : S \to \mathcal{P}(A)$ that, for each state $s \in S$, gives a set $\sigma(s)$ of action schemes that can be applied in s. (If there are none, we let $\sigma(s) = \perp$.) $EqM(K)$ implements such a σ, where $\alpha \in \sigma(s)$ iff Algorithm 1 dispatches (α, s). The set of states that can be reached from s in k steps under a given σ is given by $Reach^k(s, \sigma)$ defined as follows.

$$\begin{cases} Reach^0(s,\sigma) = & \{s\} \\ Reach^k(s,\sigma) = & \{s' \in S \mid \exists s'' \in Reach^{k-1}(s,\sigma). \\ & \quad \exists \alpha \in \sigma(s'') \text{ s.t. } s' \in \alpha(s'')\} \end{cases}$$

Finally, we show that, if starting from a recoverable state, $EqM(K)$ produces trajectories that end up in *Des*, provided a sufficient lookahead K.

Theorem 1. *Let σ be the action selection function defined by $EqM(K)$. For any state $s \in Rec(\Sigma, r)$, with $r < K$, there is a $t \leq K$ such that $Reach^t(s, \sigma) \subseteq$ Des.*

Proof. The proof is by contradiction. Assume that, for the given state s, we have $Reach^t(s, \sigma) \not\subseteq$ *Des* for all $t \leq K$. In particular, we must have $Reach^0(s, \sigma) \not\subseteq$ *Des*, and since $Reach^0(s, \sigma) = \{s\}$, then $s \in$ *Undes*. Consider then the possible cases of $s \in Rec(\Sigma, r)$.
(Case 1) $s \in N(Bas^r(\mathit{Des}))$. This means that there is at least one $\alpha \in A$ such that $\alpha(s) \subseteq Bas^r(\mathit{Des})$. For each such α, $F^{r'}(\alpha(s)) \subseteq$ *Des* for some index $r' \leq r$. Let's choose the α for which this r' is the smallest one – if there is more than one, we pick one arbitrarily. We will show that $Reach^{r'}(s, \sigma) \subseteq$ *Des*, thus contradicting our assumption. To do so, we distinguish two cases. (a) Suppose that $r' = 0$. Then we have $\mathit{Bnf}(\alpha, s)$, and since $s \in$ *Undes*, then $\mathit{Opp}_0(\alpha, s)$ is the case. Since $>_{\text{greedy}}$ has Opp_0 at the top, $EqM(K)$ will select this α. For this choice, $Reach^1(s, \sigma) = \alpha(s) \subseteq$ *Des*. (b) Suppose instead that $r' > 0$. Then, we have $\mathit{Bnf}(\alpha, s, r')$, which means that we might have $\mathit{Opp}_5(\alpha, s, r')$, depending on whether states $s' \in F^{r'}(s)$ are in *Des*. If all such s' are in *Des*, then there is no α that is an Opp_i in s and $\sigma(s) = \perp$, thus $Reach^{r'}(s, \sigma) = F^{r'}(s) \subseteq$ *Des*. If some s' is in *Undes*, then $\mathit{Opp}_5(\alpha, s, r')$. Given the order $>_{\text{greedy}}$ and since there is no Opp_0 and no smaller r', $EqM(K)$ will select and dispatch this α. Then, $Reach^{r'}(s, \sigma) = F^{r'}(\alpha(s)) \subseteq$ *Des*. In both cases (a) and (b), $Reach^{r'}(s, \sigma) \subseteq$ *Des*, and $r' \leq r < K$, thus contradicting our assumption that $Reach^t(s, \sigma) \not\subseteq$ *Des* for all $t \leq K$.
(Case 2) $s \in Bas^r(N(\mathit{Des}))$. This means that there is an $r' \leq r$ such that, for any $s' \in F^{r'}(s)$, there is at least one $\alpha \in A$ such that $\alpha(s') \subseteq$ *Des*. Let's fix one such s' and α. We have $\mathit{Bnf}(\alpha, s')$, and since $s \in$ *Undes*, we also have $\mathit{Opp}_2(\alpha, s, r')$. Given the order $>_{\text{greedy}}$ and since there is no Opp_0 and no Opp_5 (else, we would be in case 1 above), and therefore no Opp_6, then $EqM(K)$ will select and dispatch this α at state s'. From any $s' \in Reach^{r'}(s, \sigma)$, then, we have $\sigma(s') = \{\alpha\}$ and $\alpha(s') \subseteq$ *Des* because $\mathit{Bnf}(\alpha, s')$. Hence, $Reach^{r'+1}(s, \sigma) \subseteq$ *Des*, and $r' + 1 \leq r + 1 \leq K$, contradicting our assumption that $Reach^t(s, \sigma) \not\subseteq$ *Des* for all $t \leq K$.
This completes the two cases. In each one we have reached a contradiction, so it must be the case that $Reach^t(s, \sigma) \subseteq$ *Des* for some $t \leq K$. Since the above arguments do not depend on the initial choice of s, we have proved the theorem. $\square$

5 A Real Robot Example

In this section we present an example scenario implemented in a domestic service robotic system. The system as well as the services it provides have been developed in the context of a project on robotics for elderly people [Cavallo *et al.*, 2014]. In that project, the user requests services like reminding, bringing pills, helping with the laundry and many more, via a speech or tablet interface. Here we show a simple example of how these services can be provided proactively. Simplicity was chosen to exemplify $EqM(K)$ in detail, while showing that the approach can be used in closed loop to control a real robotic system.

The system architecture is an instance of the general schema illustrated in Figure 2. The robot, a Scitos G5 with a Kinova Jaco arm, moves freely in a smart home with Xbee pressure sensors mounted under chairs (see Figure 3). The current state s is inferred by a simple rule-based state estimation module which uses both real and simulated sensors. The model used by $EqM(K)$ contains hand-coded specifications of Des/Undes states, and of state transitions (α's and F). We used HTN operators to model both action schemes and F, seen as actions executed by the "environment". This encodes the environment's behavior in a compact way, including non-determinism. We used the open-source planning system JSHOP [Nau *et al.*, 2003] to perform the temporal projections needed to compute opportunities. $EqM(K)$ dispatches selected action schemes (entire sequential JSHOP plans) to a simple timeline-based executive module, which dispatches actions to the robot, and monitors their execution.

The scenario addresses the "take pills" example introduced in Section 1. The relevant predicates used to represent the state of the system are the following: the time of day is represented qualitatively by predicates $\mathrm{morning}$, noon, $\mathrm{evening}$ and night; the user locations include $\mathrm{kitchen}$, and his activities include lunch and $\mathrm{sleeping}$; his state of health is captured by predicate well, and whether he has taken the pills is denoted by predicate $\mathrm{pillstaken}$.

The free-run model is captured by the transitions $f(\mathrm{morning}, \perp, \mathrm{noon})$, $f(\mathrm{noon}, \perp, \mathrm{evening})$ and $f(\mathrm{evening}, \perp, \mathrm{night})$, together with the user model encoded by the following rules:

$$\begin{aligned} \mathrm{morning} &\Rightarrow \neg\mathrm{pillstaken} \\ \mathrm{kitchen} \wedge \mathrm{noon} &\Rightarrow \mathrm{lunch} \\ \mathrm{night} &\Rightarrow \mathrm{sleeping} \\ \mathrm{night} \wedge \neg\mathrm{pillstaken} &\Rightarrow \neg\mathrm{well} \end{aligned}$$

All states in which well does not hold are in *Undes*. It is desirable that pills are taken during lunch, hence states that include $(\mathrm{lunch} \wedge \neg\mathrm{pillstaken})$ are also in *Undes*.

There are two relevant action schemes for the robot: *remind* the user to take the pills; and *bring* the pills to the user. The *remind* action scheme has a non-deterministic result: the predicate $\mathrm{pillstaken}$ will be true either in the resulting state, or in the state following it by free-run; *bring*, on the other hand, makes $\mathrm{pillstaken}$ true in the state resulting from application of the action scheme. Both action schemes are applicable in all states where the user is $\neg\mathrm{sleeping}$. In addition, *bring* is not applicable during lunch. The time horizon used to compute $Eq(s, K)$ in $EqM(K)$ is $K = 1$.

Table 1: Equilibrium maintenance results ("pills")

State s	k	**Eq.**	**Found opportunities**	**Exec.**
morning, well	0	true	–	–
	1	false	$Opp_3(remind, s, 1)$	
noon, lunch, well, kitchen	0	false	$Opp_0(remind, s, 0)$	*remind*
	1	false	$Opp_1(remind, s, 1)$ $Opp_1(bring, s, 1)$	
evening, well	0	true	–	*bring*
	1	false	$Opp_5(bring, s, 1)$	
night, sleeping, well, pillstaken	0	true	–	–
	1	true	–	

Salient moments during one run of this example are shown in Figure 3.[1] The user was asked to sit in the living room, then transfer to the kitchen at noon, and subsequently move back to the living room in the evening. In all locations, the user was told to sit on chairs equipped with pressure sensors, whose readings were used by the state estimation module to derive the presence of the user in the location. Qualitative time was advanced artificially via terminal input.

The results (opportunities, action scheme selection and execution) are summarized in Table 1, where the state is characterized by its positive predicates. As shown, several opportunities are inferred. In the morning, an $Opp_3(remind, s, 1)$ is found, because one of the possible following states will be in *Undes* $(\mathrm{lunch} \wedge \neg\mathrm{pillstaken})$, and *remind* is both applicable and beneficial in that state. The opportunity is not executed now, since it should be enacted in a future state.

At noon, the same action becomes an Opp_0. The state also supports an opportunity of type Opp_1 for both *remind* and *bring*, when lookahead is increased to 1 in $EqM(K)$. Because of $>_{\mathrm{greedy}}$, the Opp_0 is selected, dispatched and executed. The resulting action scheme is executed as follows (see Figure 3(a,b)): the robot derives the user's location from the pressure sensor, moves to the user, utters a spoken reminder to take the pills, and moves back to its home position. Despite the reminder, the user does not take his pills.

In the evening, the pills are still not taken, and an $Opp_5(bring, s, 1)$ is inferred. This is because the user will be $\neg\mathrm{well}$ in the next state, unless he takes the pills now. Imagine having the additional rule $\mathrm{evening} \Rightarrow \mathrm{away}$ in the user model. Then, *bring* would have been inferred to be an Opp_5 already at noon, but only in case of $K = 2$. The action scheme *bring* is $Bnf(bring, s, 1)$, that is, if applied, all next states will be in *Des*. Therefore, *bring* is selected, dispatched and executed. The robot moves to the table to pick-up and deliver the pills after having obtained the updated user position (see Figure 3(c,d,e)). Note that *remind* is applicable but is not beneficial in this state because of its non-determinism, which may lead to the state being in *Undes* $(\mathrm{night} \wedge \neg\mathrm{pillstaken})$.

Note that the action scheme *bring* is the last opportunity to reach a state in *Des*, as the user will be $\mathrm{sleeping}$ at night, therefore no action scheme will be applicable. This shows the importance of temporal projection for proactive robots: a

[1]A video of another run is available at `http://www.youtube.com/user/MRLabSweden`.

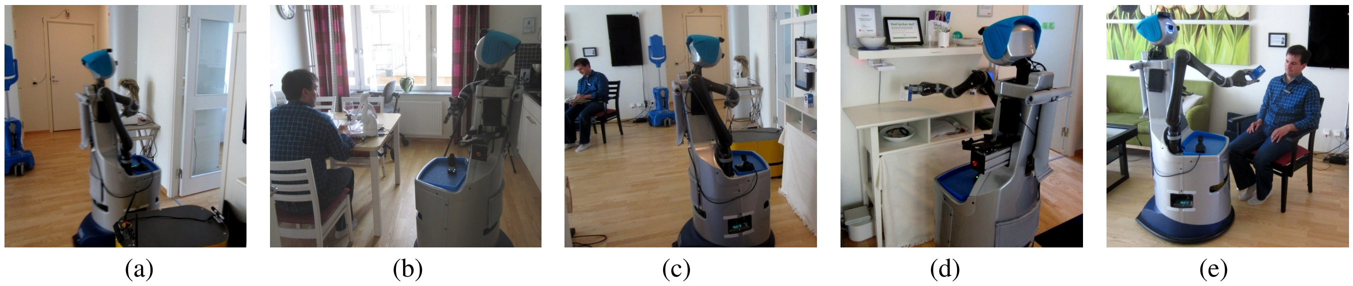

Figure 3: Salient moments of one run: the robot reminds the user to take the pills (a, b); since the user does not heed the reminder, the robot proactively fetches (c, d) and brings him the pills while he is reading in the living room (e).

purely reactive system would have missed this opportunity.

6 Discussion and Conclusion

We have presented a framework that aims at making robots proactive by providing them with the ability to generate their own goals and act upon them. By decoupling the factors that determine conditions for acting – states, desirability and robot capabilities – our framework is able to *infer* whether to act (condition), how (what action) and when (in which current or future state), while this is explicitly encoded in many other approaches [Vattam *et al.*, 2013]. Our framework is not reactive, but anticipatory: we do temporal projection to predict future undesired states and identify actions that avoid them. For this purpose, we introduce the notion of *opportunities* as basis for deciding which goals to pursue and when.

Opportunities for acting were mentioned before in [Pollack and Horty, 1999], where a function called *Alternative Assessment* is used to assess them; in [Beetz, 2002], who defines them in a purely reactive way; and in [Muñoz-Avila *et al.*, 2015], who uses them as motives to direct an agent towards states where more actions are available. These usages of the term "opportunity" are substantially different from the one proposed here. Our opportunities are closer to Bratman's *intentions* [Bratman, 1987], which are derived from an agent's goals and beliefs. Like intentions, an opportunity to act holds as long as the agent believes that that action is achievable (as encoded by $s \in \mathit{dom}(\alpha)$) and that it leads to its goals (as encoded by $\mathit{Bnf}(\alpha, s)$) [Cohen and Levesque, 1990; Parsons *et al.*, 2000]. As such, opportunities are potentially subject to the same problems as Bratman's intentions: an agent could reach equilibrium by going outside the domain of all actions, thus making them not achievable. Theorem 1 above guarantees that $EqM(K)$ will recover equilibrium by moving into desirable states rather than moving outside the domain of actions.

Intentions have been extensively used in robotics in the context of BDI architectures and of the PRS system [Ingrand *et al.*, 1992], as well as in frameworks for action selection based on forward models [Anderson *et al.*, 2004; Laird *et al.*, 1987]. Most instantiations, however, only consider their immediate applicability or benefit, as we do with Opp_0 opportunities. Our framework introduces anticipation based on a predictive model of the system, where the applicability and/or the benefit of intentions may be in the future, as per opportunity types Opp_{1-6}. Knowing about opportunities for future action may be important: for instance, to allow the robot to plan and schedule other tasks, or to reduce the search space by exploiting the fact that some decisions have already been taken [Pollack and Horty, 1999].

Using a predictive model to determine how to act to satisfy goals is reminiscent of planning. However, we not only enable the robot to autonomously select actions to achieve given goals, as done in planning, but also to infer the goals themselves. In line with [Hawes, 2011; Beaudoin, 1994] we identify the need for a deliberative process independent from planning to realize this. This process, illustrated in Figure 2 above, can be seen as an instance of continuous planning [Dean and Wellman, 1991; Cox and Veloso, 1998] with the meta-goal of maintaining equilibrium.

Our approach is related to computational models of intrinsic motivation in agents [Oudeyer and Kaplan, 2007], and it can be seen as a way to combine predictive and competence-based models. In this view, the intrinsic motivation of our agents is to maintain equilibrium. There are other approaches that provide concrete implementations of goal autonomy in specific autonomous agent systems [Molineaux *et al.*, 2010; Galindo and Saffiotti, 2013; Hanheide *et al.*, 2010]. While these are valuable contributions, our aim is to provide a general formulation of this problem.

Uncertainty is always a major concern when dealing with robotic systems. In our framework, uncertainty is captured by non-determinism: multiple alternative states may be undesirable, the free-run dynamics of the system may be non-deterministic, and action schemes may have non-deterministic effects. It might be useful to quantify this uncertainty, e.g., by associating degrees of desirability to states and probabilities to state transitions. This would enable to define different flavors of *beneficial*, e.g., partial, weak, or comparative ones, and eventually to account for graded preferences and competitive requirements. We plan to explore decision-theoretic planning [Karlsson, 2001] or POMDPs [Kaelbling *et al.*, 1998] as a basis for this extension.

Finally, this paper helps to pinpoint general key open questions for proactive robots (see Section 3): the problem of selecting among different opportunities; and the problem of interleaving equilibrium maintenance, planning and execution. We believe that the equilibrium maintenance approach presented in this paper provides a precise way to frame these questions, as well as some valuable initial answers.

Acknowledgments. This work was funded the Robot-Era project, EC FP7/2007-2013 grant no. 288899.

References

[Aha, 2015] D.W. Aha, editor. *Goal reasoning: Papers from the ACS workshop*. Georgia Institute of Technology, 2015. Technical Report GT-IRIM-CR-2015-001.

[Anderson *et al.*, 2004] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychol Rev*, 111(4):1036–1060, 2004.

[Beaudoin, 1994] L. Beaudoin. *Goal processing in autonomous agents*. PhD thesis, Univ. of Birmingham, 1994.

[Beetz, 2002] M. Beetz. Towards comprehensive computational models for plan-based control of autonomous robots. In *Mechanizing Mathematical Reasoning*, pages 514–527. Springer, 2002.

[Bratman, 1987] M.E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press: Cambridge, MA, 1987.

[Cavallo *et al.*, 2014] F. Cavallo, R. Limosani, A. Manzi, M. Bonaccorsi, R. Esposito, M. Di Rocco, F. Pecora, G. Teti, A. Saffiotti, and P. Dario. Development of a socially believable multi-robot solution from town to home. *Cognitive Computation*, 6(4):954–967, 2014.

[Cohen and Levesque, 1990] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artif Intell*, 42:213–261, 1990.

[Cox and Oates, 2013] M.T. Cox and T. Oates. MIDCA: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. Technical Report CS-TR-5025, Univ. of Maryland, 2013.

[Cox and Veloso, 1998] M.T. Cox and M.M. Veloso. Goal transformations in continuous planning. In *Proc. of the AAAI Fall Symposium on Distributed Continual Planning*, pages 23–30, 1998.

[Cox, 2007] M.T. Cox. Perpetual self-aware cognitive agents. *AI magazine*, 28(1):32, 2007.

[Dean and Wellman, 1991] T.L. Dean and M.P. Wellman. *Planning and control*. Morgan Kaufmann Publishers Inc., 1991.

[Galindo and Saffiotti, 2013] C. Galindo and A. Saffiotti. Inferring robot goals from violations of semantic knowledge. *Robotics and Autonomous Systems*, 61(10):1131–1143, 2013.

[Ghallab *et al.*, 2014] M. Ghallab, D. Nau, and P. Traverso. The actor's view of automated planning and acting: A position paper. *Artif Intell*, 208:1–17, 2014.

[Hanheide *et al.*, 2010] M. Hanheide, N. Hawes, J. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, and G.-J. Kruijff. A framework for goal generation and management. In *Proc. of AAAI workshop on goal-directed autonomy*, 2010.

[Hawes, 2011] N. Hawes. A survey of motivation frameworks for intelligent systems. *Artif Intell*, 175(5):1020–1036, 2011.

[Hertzberg and Chatila, 2008] J. Hertzberg and R. Chatila. AI reasoning methods for robotics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 207–223. Springer, 2008.

[Ingrand *et al.*, 1992] F.F. Ingrand, M.P. Georgeff, and A.S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44, 1992.

[Kaelbling *et al.*, 1998] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif Intell*, 101(1 - 2):99 – 134, 1998.

[Karlsson, 2001] L. Karlsson. Conditional progressive planning under uncertainty. In *Proc. of IJCAI*, pages 431–438, 2001.

[Laird *et al.*, 1987] J.E. Laird, A. Newell, and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artif Intell*, 33(1):1–64, 1987.

[Molineaux *et al.*, 2010] M. Molineaux, M. Klenk, and D.W. Aha. Goal-driven autonomy in a navy strategy simulation. In *Proc. of AAAI*, 2010.

[Muñoz-Avila *et al.*, 2015] H. Muñoz-Avila, M.A. Wilson, and D.W. Aha. Guiding the ass with goal motivation weights. In *Goal Reasoning: Papers from the ACS Workshop*, pages 133–145, 2015.

[Nau *et al.*, 2003] D. Nau, O. Ilghami, U. Kuter, J.W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *J Artif Intell Res*, 20:379–404, 2003.

[Oudeyer and Kaplan, 2007] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2007.

[Parsons *et al.*, 2000] S. Parsons, O. Pettersson, A. Saffiotti, and M. Wooldridge. Intention reconsideration in theory and practice. In *Proc. of ECAI*, pages 378–382, 2000.

[Pollack and Horty, 1999] M. E Pollack and J.F. Horty. There's more to life than making plans: plan management in dynamic, multiagent environments. *AI Magazine*, 20(4):71, 1999.

[Rao and Georgeff, 1991] A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. of KR*, pages 473–484, 1991.

[Vattam *et al.*, 2013] S. Vattam, M. Klenk, M. Molineaux, and D.W. Aha. Breadth of approaches to goal reasoning: A research survey. In *Goal Reasoning: Papers from the ACS Workshop*, page 111, 2013.

[Zhang *et al.*, 2015] Y. Zhang, V. Narayanan, T. Chakraborti, and S. Kambhampati. A human factors analysis of proactive support in human-robot teaming. In *Proc. of IROS*, 2015.