Find Out Why Reading This Paper is an Opportunity of Type Opp_0 [1]

Jasmin Grosinger and **Federico Pecora** and **Alessandro Saffiotti** [2]

Abstract. Under what conditions should a cognitive robot act? How do we define "opportunities" for robot action? How can we characterize their properties? This paper offers an apparatus to frame the discussion. Starting from a simple introductory example, we specify an initial version of a formal framework of *opportunity* which relates current and future states and *beneficial* courses of action in a certain time horizon. An opportunity reasoning algorithm is presented, which opens up various new questions about the different types of opportunity and how to interleave opportunity reasoning and action execution. An implementation of this algorithm is tested in a simple experiment including a real mobile robot in a smart home environment and a user.

1 INTRODUCTION

Reasoning on affecting change is central to proactive robot systems. Most current techniques focus on *how* to affect change, but not *why*. Consider the following example.

There is a physical object in our world, a banana, which can be either *fresh*, *ripe*, *overripe* or *rotten*. The banana changes state over time, from the first to the last. Maintaining a *desirable* world state includes that physical objects in the world, the banana, are in states that are desirable. It is desirable that the banana is fresh, ripe, or overripe, while a rotten banana is undesirable. Also, the banana should be eaten before it becomes rotten. Assume there is a human user who can be in either of the states *breakfast*, *reading*, *lunch* or *away* and there is a mobile robot capable of bringing the banana to the human for consumption. Assume also that the robot possesses a model representing the user's and the banana's states and how long it takes to transition over them. How should the robot choose, among all possible intermediate states of the banana, when to act? Is it desirable that the robot immediately takes action as soon there is a banana in the world, no matter what the states of the banana and the user are?

The act of bringing the banana to the user for consumption achieves a desired state from the banana's perspective — but what does this imply in terms of the world state? Not only should the banana be eaten in a favorable state, but the robot should not act intrusively against the user. For example, it would not be appropriate for the robot to force-feed the sleeping user just because the banana will soon be rotten! There are states in the world which are more suitable for taking a specific action than others — they offer *opportunities* for acting. For instance, the robot may offer the banana the next morning for breakfast. Consuming the banana before it is rotten and not being intrusive towards the user are desirable, and thereby contribute to the maintenance of a desirable world state.

How do the desirable states of the banana affect whether we classify a state of the user as being suitable for robot action? When the banana is fresh, it is not necessary to act. This may even result in an undesirable state of the user, therefore an undesirable world state. A ripe banana increases the necessity to act (we can predict that it will eventually become rotten), but we can afford to choose among few, well tailored states that are suitable for taking action. An overripe banana is closer to being rotten. This influences which states we now classify as suitable for taking action: this is a larger set, and potentially less perfectly tailored to acting.

For all the desirable states of the banana in this simple example, the actions of the robot are always the same: bring the banana to the user for consumption. The banana's desirable states do not differ in their influence on *what* the robot should do, rather *when* it should act. If the banana becomes rotten, the user's state has no influence on which context the robot uses as a "trigger" to act. Also, the robot will act differently: instead of bringing the banana to the user, the robot will decide to dispose of it.

This paper presents a formal framework to capture the issues suggested by the example illustrated above. It also presents an algorithm that uses this framework in order to offer a first approach to the problem of what action to select in which context. The proposed approach opens numerous questions and future directions, suggesting that the deliberation needed for managing goals — generating, activating/suspending, adding/removing goals — and selecting and scheduling actions is a rich pool of under-explored issues.

The next section introduces the formal framework. Section 3 gives an algorithm for opportunity reasoning based on this framework. Section 4 shows a simple example of this algorithm controlling a real robot. The last two sections discuss related work and conclude.

2 FORMALIZING OPPORTUNITY

We consider a system $\Sigma = \langle S, U, f \rangle$, where S is a finite 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. 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 $s_t \in S$.

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 $\perp$, and is given by:

[1] This work was funded by the EC Seventh Framework Programme (FP7/2007-2013) grant agreement no. 288899 Robot-Era.

[2] Centre for Applied Autonomous Sensor Systems (AASS), Örebro University, 70182 Örebro, Sweden, email: {jngr, fpa, asaffio}@aass.oru.se

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

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 in which the banana is rotten is in *Undes*, whereas any state in which the banana is gone is in *Des* (whether because it was eaten, or disposed of, or was never there). 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. An action scheme α abstracts all details of action: $\alpha(S') = S''$ only says that there is a way to go from any state in S' to some state in S''. We denote by *dom*(α) the domain where α is defined.

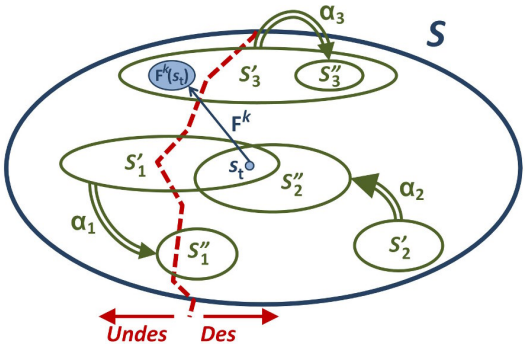

Figure 1. Graphical illustration of action schemes. The state space S is partitioned in the *Des* and *Undes* subsets. Each action scheme α_i may change the state from being undesirable to being desirable or vice-versa.

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 $s' \in S'_1$, and when applied it will bring Σ to some new state $s'' \in S''_1$. At the current time t the system is in the desirable state s_t, and if no action is applied it will move in k steps to some state in the set $F^k(s_t)$, which will be undesirable.

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

$$\mathit{Bnf}(\alpha, s) \text{ iff } \exists S' \in \mathit{dom}(\alpha) \text{ s.t. } s \in S' \wedge \alpha(S') \subseteq \mathit{Des}$$

In Figure 1, α_1 is applicable in s_t since its domain is S'_1 and $s_t \in S'_1$. However, it is not beneficial in s_t since it does not bring the system into states which are all desirable, i.e., $\alpha_1(S'_1) = S''_1$ and $S''_1 \not\subseteq$ *Des*. Scheme α_2 would be beneficial in another state, but it is not applicable in s_t. Scheme α_3 is not beneficial in s_t, but it will become so in k steps. In our banana example, the scheme α_{bring}, which delivers a banana to the user, can be applied in any state s where the user is having breakfast and the banana is either ripe of overripe: these conditions characterize *dom*(α_{bring}). This scheme is beneficial in any such state s, since the resulting states are desirable because the banana has been eaten.

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

$$\mathit{Bnf}(\alpha, s, k) \text{ iff } \exists S' \in \mathit{dom}(\alpha) \text{ s.t. } s \in S' \wedge F^k(\alpha(S')) \subseteq \mathit{Des},$$

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

2.2 Opportunities

We can use the above apparatus to characterize the different types of opportunities for action discussed in our example. Let $s \in S$ and let $k \in \mathbb{N}$ be a finite time horizon. There are at least six properties that determine whether α is an opportunity for acting in s:

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

The first two properties characterize schemes that can be *applied in the future* in response to a *current* undesired situation. In particular, $\mathit{Opp}_1(\alpha, s, k)$ 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. $\mathit{Opp}_2(\alpha, s, k)$ 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. The third and fourth properties characterize schemes that can be *applied in the future* in response to either a *foreseen* 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

$$\mathit{Opp}(\alpha, s, 0) \text{ iff } s \in \mathit{Undes} \wedge \mathit{Bnf}(\alpha, s), \tag{1}$$

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

A few examples help to appreciate the differences among these properties. Consider a system whose free run behavior goes through the sequence of states s_0 (user sleeping, ripe banana); s_1 (user having breakfast, overripe banana); and s_2 (user at work, rotten banana). Let the current state be s_1 and let $k = 1$. Then, scheme α_{bring} is an opportunity of type Opp_6 in s_1 because, if applied now, it will avoid reaching the undesired state s_2. Scheme α_{dump} (dump the banana into the trash can) is Opp_4 because it can be applied later, once we get in undesired state s_2, and bring the system back to a desired one. Imagine now a GM banana which may take longer to become rotten, i.e., $F^1(s_1)$ includes both s_2 as above and s'_2 in which the banana is still overripe. Then, scheme α_{bring} is Opp_5 in s_1, but not Opp_6. Finally, suppose in state s_3 a garbage-bot will stop at the door; then, the scheme α_{ho} (hand-over the banana to the garbage-bot) is Opp_2 in s_2, since we need to wait until the garbage-bot passes by.

3 COMPUTING OPPORTUNITIES

A robot might use the above framework of opportunities to make informed autonomous decisions about what actions to perform in which situations. Instead of receiving goals from a human operator, the robot can apply the opportunity framework to generate its own goals, given a model of the desired and undesired states and a model of the world dynamics. Endowing the robot with such a capability requires identifying the computational tasks involved in the framework. An *opportunity reasoning* algorithm first needs to determine whether there exist one or more opportunities in a particular state. The found opportunities then need to be evaluated, that is deciding which ones among the several opportunities at a particular time to select for execution. We then need to schedule when to execute the opportunity, and when to re-evaluate the opportunities, i.e., how to interleave oppourtunity execution and opportunity evaluation.

3.1 Assumptions

In order to avoid complexities which are, at this stage, not relevant, we make here the standard assumptions of classical planning. These are admittedly restrictive, and we shall explore their relaxation in future versions of our framework. We consider a finite set of predicates $P = \{p_1, \ldots, p_n\}$. A state s is completely determined by the predicates that are `true` in s (*closed world assumption*). In the banana example, banana_fresh, banana_ripe, ... user_reading, user_breakfast, ... are predicates. The set of all states S is partitioned in the sets *Des* and *Undes* of desirable and undesirable states. We also consider a finite set of *action schemes* $A = \langle \alpha_1, \ldots, \alpha_m \rangle$. For now, we take each action scheme to be a *linear plan*, i.e., a finite sequence of actions $\alpha_i = \langle u_1, ..., u_{|\alpha|} \rangle$, where $|\alpha|$ denotes the number of actions in the plan. We assume that actions are *deterministic*, and hence the state transition function γ [6] has the form $\gamma : S \times U \to S$.

The function γ can be extended in the obvious way to whole action schemes: $\gamma(s, \alpha)$ is the ending state resulting from successively applying the actions in α starting from state s (note that action schemes are also deterministic). We consider an abstract action-based planning model to encode the state transition behavior: we associate each action scheme α_i to a set of preconditions $P_i \subseteq P$, a set of add (positive) effects $E_i^+ \subseteq P$, and a set of delete (negative) effects $E_i^- \subseteq P$. Using this model, the result of applying scheme α_i can be written as $\gamma(s, \alpha_i) = (s \setminus E_i^-) \cup E_i^+$ if $P_i \subseteq s$, and undefined otherwise.

In addition to classical planning assumptions, we also assume to have *complete* knowledge of the time models of *all* entities that affect the world state. In the banana example this means we know the temporal evolution of the state of the banana and of the user, i.e., the free-run behavior F.

Figure 2 shows an example assuming a time horizon of 1. State s_0 constitutes of the user having breakfast and the banana being fresh. It does not ential an opportunity for action, nor does s_1, in which the banana is ripe and the user is reading. In state s_2 the user is still reading, and the banana has turned overripe. Analyzing the free-run behavior we can see that an undesirable state – the banana being rotten – can be reached in one step from here. Therefore, there is an opportunity of type Opp_5 in state s_2 to apply the action scheme α_{bring}, i.e.: $Opp_5(\alpha_{\text{bring}}, s_2, 1)$ is true.

We assume that the robot system makes available a procedure *exec(α, s, Opp, numSteps)* that is able to schedule the execution of the first *numSteps* actions in α at a given (present or future) state s. Execution of α means the sequential application of actions of α. The

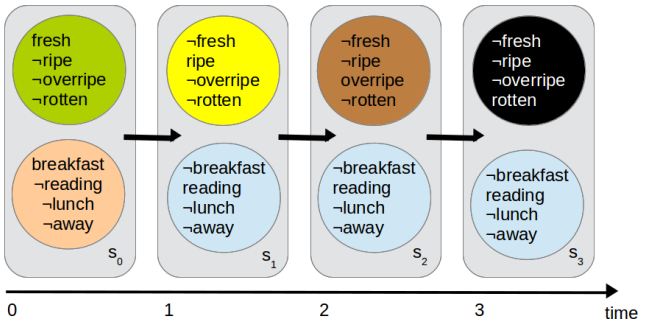

Figure 2. State automaton for the banana (top) and user (bottom) comprising the world states s_0 to s_3.

exec procedure might use knowledge of the type of opportunity *Opp* being executed to inform scheduling decisions. For example, when faced with bounded resources an α_i of opportunity type Opp_0 might be executed, whereas an α_j of type Opp_3 might be delayed in favor of another action of *exec* that consumes the same resource.

3.2 Algorithm

Algorithm 1 performs opportunity reasoning in state s. It first finds what opportunities exist in s, that is, it categorizes each $\alpha_i \in A$ into opportunity types from a given set *OppTypes* — here, this is the set $\{Opp_0, \ldots, Opp_6\}$ defined above. The algorithm that does so, Algorithm 2, checks increasingly long time horizons k up to an upper bound K.[3] It collects, through the use of the *check_opp* functions, the action schemes that constitute opportunities, together with the states where they can be used, their opportunity types and their time horizons. A different *check_opp* function is implemented for each opportunity type. For example, algorithms 4 and 5 show the *check_opp* for Opp_0 and Opp_5, respectively. These are the opportunity types used in the experiment in section 4. Note that these algorithms return a singleton state because of our deterministic assumption, but this need not be true in the general case.

Data:
$K \in \mathbb{N}$: max time horizon
$<_{\text{opp}}$: a partial order relation over oppotunity types *OppTypes*
numSteps $\in \mathbb{N}^+$: number of steps to be executed
1 **while true do**
2 OppQueue $\leftarrow$ `find_opp` (s, K)
3 $\langle \alpha, s', Opp, k \rangle \leftarrow$ `select` (OppQueue, $<_{\text{opp}}$)
4 **if** $\langle \alpha, s', Opp, k \rangle \neq \perp$ **then**
5 `exec` $(\alpha, s', Opp,$ numSteps)
6 **end**
7 **end**
Algorithm 1: opportunity_reasoning($s, K, <_{\text{opp}}$, numSteps):-

The second step in Algorithm 1 is to select which one among the found opportunities should be executed. Selecting an opportunity (Algorithm 3) may depend on the specific action scheme, on the type of opportunity, and on the time horizon by which this opportunity will have an effect. How to perform this selection is a key question for opportunity reasoning, which our framework will hopefully help to explore. In this paper, we simply use a given strict partial

[3] Algorithm 2 is slightly simplified for readability. The actual algorithm treats $k = 0$ and Opp_0 as special cases since all opportunity types collapse to Opp_0 when $k = 0$ (equation (1)).

```
forall the Opp_i ∈ OppTypes do
    forall the α ∈ A do
        for k ← 0 to K do
            S' ← check_opp (Opp_i, α, s, k)
            forall the s' ∈ S' do
                push (⟨α, s', Opp_i, k⟩, OppQueue)
            end
        end
    end
end
return OppQueue
```

Algorithm 2: find_opp(s, K): $OppQueue$

order $<_{\text{opp}}$ over the set of opportunity types: we select the opportunity whose type is highest according to $<_{\text{opp}}$. We break ties using the value of k: if there are two opportunities $\langle \alpha_i, s_i, Opp_i, k_i \rangle$ and $\langle \alpha_j, s_j, Opp_j, k_j \rangle$, such that Opp_i and Opp_j are not comparable according to $<_{\text{opp}}$, then we select the one with the lower k. If $k_i = k_j$ we select one randomly.

```
if OppQueue ≠ ∅ then
    sort (OppQueue, 3rd, <_opp)
    sort (OppQueue, 4th, ≤)
    return (first (OppQueue))
end
return ⊥
```

Algorithm 3: select(OppQueue, $<_{\text{opp}}$): $\langle \alpha, s', Opp_i, k \rangle$

Next, Algorithm 1 invokes execution of the selected opportunity, indicating how many steps of its action plan should be executed before starting a new iteration of opportunity reasoning.

```
if s ∈ Undes ∧ P_α ⊆ s then
    s' ← γ(s, α)
    if s' ∈ Des then
        return {s}
    end
end
return ∅
```

Algorithm 4: check_opp(Opp_0, α, s, k):$S' \subseteq S$

Concerning the partial order $<_{\text{opp}}$ over *OppTypes*, in our experiments we use the one graphically represented in Figure 3. The motivation behind this choice is as follows. Opp_0 is at the top because the current state s is undesirable and Opp_0 is an immediate way out of it, i.e., it provides an α that is beneficial *now*. On the next level there are Opp_5 and Opp_6 which as well provide an opportunity for action that is beneficial *now*. The current state is not in *Undes*, but some future states will be so it is beneficial to act now in order to prevent this. Since there is no general way to assess whether Opp_5 or Opp_6 is more appropriate for acting, the two are not ordered. On the next level of opportunity types we put Opp_1 and Opp_2 — the current state is undesirable but there is no action plan available that can help escape from it now. However, there exists one for at least one reachable state in the future that brings the state in *Des*. The last level in the opportunity type hierarchy contains Opp_3, Opp_4. The reason for their low priority is that both the possible undesirability of a state and the benefit of applying an action plan are placed in the future.

Finally, it is worth spending some words on the *exec* function — the executive layer of the system that schedules actions, dispatches

```
if P_α ⊆ s then
    s' ← γ(s, α)
    s'' ∈ F^k(s')
    if s'' ∈ Des then
        forall the s''' ∈ F^k(s) do
            if s''' ∈ Undes then
                return {s}
            end
        end
    end
end
return ∅
```

Algorithm 5: check_opp(Opp_5, α, s, k):$S' \subseteq S$

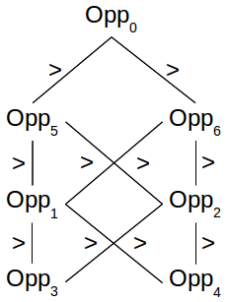

Figure 3. Opportunity types - partial order.

them, and possibly also assesses their execution progress. Significant work has been done in plan execution and monitoring [12], and little attention has been devoted to the issue of determining how to interleave action execution and opportunity reasoning. Specifically, suppose the current state s is an opportunity of type Opp_0 for plan α, hence calling for the execution of α immediately. How many steps (*numSteps*) should be executed before re-assessing whether there exist new opportunities? This is an interesting question, which we plan to address in future work. For the purposes of this article, we assume *numSteps* = $|\alpha|$, that is, the system executes the whole plan α before re-evaluating the opportunities.

4 A REAL-WORLD EXAMPLE

In the European FP7 project Robot-Era [15] we use robots integrated in a smart environment to provide elderly people with services like bringing them medications. Currently, users invoke the services by means of a suitable interface, but in the future the Robot-Era system should proactively decide if and when to provide each service. The approach presented above is meant to provide a stepping stone toward this goal. In this section, we show a simple example implemented in the context of the Robot-Era system.

As actuator we use a Scitos G5 mobile robot extended with a Kinova Jaco robot arm. The location of the user is determined by pressure sensors under chairs on which the user may sit. As a communication interface between opportunity reasoning, planning, executive layer, robot, and sensors, we use the PEIS middleware [16]. This middleware transparently connects heterogeneous robotic devices providing a common interface amongst system components. We assume to have complete knowledge of the time models of the entities that comprise the overall world state – the banana and the user – that is, the opportunity reasoning retrieves the hard-coded time models from an internal storage.

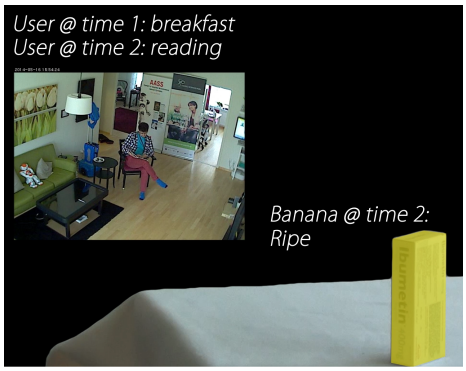

Figure 4. Time 2: User reading, Banana ripe - No opportunity.

Figures 4, 5 ,6 show different states of the execution of a real experiment with time horizon $K = 1$. The figures show that we use a pill box representing the banana: this is meant to simplify object grasping, and it reflects the fact that one of the real Robot-Era services concerns bringing a pill box to the user. In the initial situation, the user is having breakfast and the banana being in state fresh — this situation, which persists for one time unit, does not offer an opportunity for acting. At the next time step (see Figure 4), the user is in state reading while the banana has turned ripe. Again, no opportunity for acting is inferred.

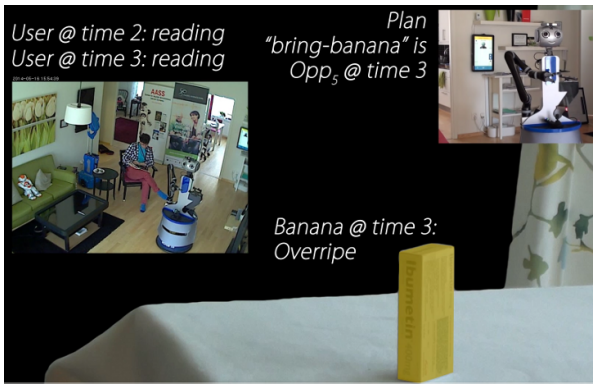

Figure 5. Time3: User reading, Banana overripe - opportunity Opp_5: Robot moves to the banana.

Figure 6. At time 3 the robot picks up the banana, brings it and hands it over to the user following an opportunity of type Opp_5.

At time three, the user is still reading, and the banana is now overripe. The opportunity reasoning framework is aware of the time models of the banana and the user, thus inferring that in one time step from now the banana will be rotten. Hence, the algorithm deduces an opportunity of type Opp_5 in this situation and issues the action plan to bring the banana to the user in order to avoid the future undesirable state of a rotten banana. (Please refer to Section 2.2 for more examples of inferring other opportunity types in different settings). A given executive layer [4] takes care of planning and monitoring the navigation of the robot to the banana, grasping it, navigating to the user and handing it over. Figure 7 shows screen dump of the opportunity reasoning system at this time point, while figures 5 and 6 show selected scenes from the real experiment.[4]

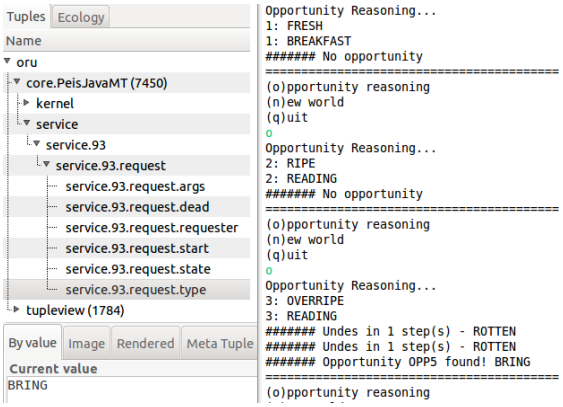

Figure 7. Time3: System screen dump - opportunity Opp_5

Note that without reasoning about opportuninities the robot would have had to *explicitly be told* what to do – bring the banana – and when – a *ripe* banana can be brought when the user is in state *breakfast*; an *overripe* banana can be served to a user in state *lunch*; the robot should not intrude when the user is *reading*, but on the other hand if the banana is anticipated to be *rotten* soon, this constraint can be neglected. It is evident that modeling the problem of goal selection from general principles, like desirability and opportunity, can reduce the amount of ad-hoc programming needed for realizing a competent robot. We believe that opportunity reasoning is a step in the direction of realizing a general framework for this purpose.

5 RELATED WORK

In this paper we have focused on characterizing the problem of opportunity reasoning. Our starting point was a simple, yet revealing example involving a robot and a rotting banana. The attentive reader has certainly spotted that in our realization of this example (see Section 4), the robot must be able to perform a wide range of cognitive tasks, which include perceiving, planning and acting. Studies in cognitive architectures, like ACT-R [1], BDI [14, 10] and SOAR [11], lend support to the argument that diverse cognitive capabilities must be studied jointly. However, while the BDI framework aims to capture intentions and desires of an agent in a formal theory, our notion of opportunity is placed at a more global system level: instead of capturing one single agent's *inner* cognitive world, we use the notion of opportunity to account for the *whole* world state, relating it to beneficial courses of action within certain time horizons. Another difference is that we are not concerned with implementing human-like cognition, rather with realizing non-trivial cognitive capabilities in robots. Similarly to cognitive architectures, the view proposed by

[4] The complete video of the experiment is available at `http://www.youtube.com/watch?v=F_b6QKID8D4`.

Ghallab et al. [7] defines the deliberative capabilities that enable a robot to act appropriately, and points out that a model capturing motivation still is unexplored. Pollack et al. [13] find the need for a wider range of reasoning than just plan generation – they call this *Active Planning Process Management* – but implement only parts of the challenges and do not offer an overall formal model to state the problem. The view put forth in this paper agrees with these holistic perspectives, and is inspired by both cognitive architectures and "planning as acting" or "plan management". However, we argue that this specific problem deserves a formal definition, and we believe that formal methods can be exploited to provide general solutions.

A problem that is related to our work as well is the issue of goal management. Cox's work [3] is similar to ours in that the author aims to establish a system with the capability of *Awareness* — "...being able to comprehend when the world is in need of change and, as a result, being able to form an independent goal to change it". A similar idea underlies the approach to goal generation proposed by Galindo and Saffiotti [5]. However, again both approaches adopt an agent's perspective rather than the more global, system-centric view of opportunity proposed here. Also, goals as formulated by Cox are hard-coded (something we aim to avoid): goal generation is achieved by inference from explanations of "anomalies" or "interesting events" which are inserted in the system by hand. Hawes [8] identifies the need to investigate questions related to selecting and scheduling goals. The author suggests to address this problem with the notion of urgency, which is so far undefined. We believe that this idea could be accommodated and defined explicitly within the framework of opportunity reasoning presented here, and we plan to study this issue in future work.

Heintz et al. [9] deal with stream-based reasoning as an attempt to integrate deliberative reasoning functionalities for rational goal-directed behavior in autonomous agents. Their *DyKnow* system focuses on bridging the gap between incomplete metric sensor data of the environment and crisp symbolic knowledge representing nominal system behavior. While our work too addresses the integration of deliberation functionalities in autonomous robotic agents in sensorized environments, our focus lies more in reasoning on which actions to schedule depending on perceived or inferred context.

Beetz [2] uses the same term as we have in this paper — *opportunity*. However, the author's purely reactive understanding of this notion is fundamentally different from ours. Opportunity according to Beetz supports the decision whether to interrupt the current execution of a plan to execute another task, or finish a previously interrupted task. Also, no formal model of opportunity is established, and different types of opportunities are not considered.

6 CONCLUSIONS

The need to include opportunity into action theory is increasingly evident. Although tentative, the above formulation of opportunity points to several under-addressed issues connected with acting in robotic systems. Characterizing types of opportunities helps to discover and discriminate between qualitatively different contexts in which robot action is called for. Of course, one could encode explicit ad-hoc rules, for instance *'Bring me the banana if it is not rotten!'*, but that is not our aim. Instead, we investigato how decisions for acting can be derived from first principles by recognizing "opportunities for action".

Future work will explore different ways to characterize priorities between opportunity types. We will also investigate the open question of how to interleave action execution and opportunity reasoning. Our ambition is to develop an accurate formulation of opportunity whose formal properties can be studied and which can be related to the general problem of action selection. We believe it is necessary to introduce degrees of desirability of states in order to account for more fine-grained forms of opportunity. Furthermore, we aim to define urgency and utilize it in goal management, i.e., selecting and scheduling of goals. Introducing a richer representation of state or context may lead to re-formulations in the opportunity framework. We aim to mount these extensions to our framework in a meta-level system of interrelated participants to find the best trade-off decision for acting.

Current techniques for planning, acting, context awareness and other cognitive abilities that a robot should possess, are usually ignorant of the *reason* for affecting change. At least part of this reason is *opportunity*. We believe that it is useful to characterize this formally, if only to discover which existing techniques are applicable in a useful and proactive robot system, which have to be adapted, and which are missing entirely.

REFERENCES

[1] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin, 'An integrated theory of the mind.', *Psychol Rev*, **111**(4), 1036–1060, (2004).

[2] Michael Beetz, 'Towards comprehensive computational models for plan-based control of autonomous robots', in *Mechanizing Mathematical Reasoning*, 514–527, Springer, (2002).

[3] Michael T Cox, 'Perpetual self-aware cognitive agents', *AI magazine*, **28**(1), 32, (2007).

[4] Maurizio Di Rocco, Federico Pecora, Subhash Sathyakeerthy, Jasmin Grosinger, Alessandro Saffiotti, Manuele Bonaccorsi, Raffaele Limosani, Alessandro Manzi, Filippo Cavallo, Paolo Dario, and Giancarlo Teti, 'A planner for ambient assisted living: From high-level reasoning to low-level robot execution and back', in *Proc of the AAAI Spring Symposium on Qualitative Representations for Robots*, pp. 10–17, (2014).

[5] Cipriano Galindo and Alessandro Saffiotti, 'Inferring robot goals from violations of semantic knowledge', *Robotics and Autonomous Systems*, **61**(10), 1131–1143, (2013).

[6] Malik Ghallab, Dana Nau, and Paolo Traverso, *Automated planning: theory & practice*, Elsevier, 2004.

[7] Malik Ghallab, Dana Nau, and Paolo Traverso, 'The actor's view of automated planning and acting: A position paper', *Artif Intell*, **208**, 1–17, (2014).

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

[9] Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty, 'Bridging the sense-reasoning gap: Dyknow–stream-based middleware for knowledge processing', *Advance Engineer Informatics*, **24**(1), 14–26, (2010).

[10] François Felix Ingrand, Michael P Georgeff, and Anand S Rao, 'An architecture for real-time reasoning and system control', *IEEE Expert*, **7**(6), 34–44, (1992).

[11] John E Laird, Allen Newell, and Paul S Rosenbloom, 'Soar: An architecture for general intelligence', *Artif Intell*, **33**(1), 1–64, (1987).

[12] Ola Pettersson, 'Execution monitoring in robotics : a survey', *Robotics and Autonomous Systems*, **53**(2), 73–88, (2005).

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

[14] Anand S. Rao and Michael P. Georgeff, 'Modeling rational agents within a BDI-architecture', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, (1991).

[15] Robot-Era. http://www.robot-era.eu. Accessed: 2014-05-30.

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