EU FP7 CogX
ICT-215181
May 1 2008 (52 months)

DR 1.4:
Integrating Intention Changes into Continual Planning and Acting

Marc Hanheide, Moritz Göbelbecker, Nick Hawes

BHAM, ALU-FR
⟨cogx@cs.bham.ac.uk⟩

Due date of deliverable:	May 31 2012
Actual submission date:	June 30 2012
Lead partner:	BHAM
Revision:	final
Dissemination level:	PU

A core objective of CogX is to develop systems that can accomplish certain *tasks* and *extend* their knowledge about the world autonomously. Such systems have to swiftly switch between, and also accommodate a number of different intentions at the same time to implement the more generic drives of task accomplishment and self-extension. In CogX we are committed to using planning to determine how goals are achieved, so the question is how different goals, relating to the intentions of the system, can be generated, selected and managed. This report complements earlier reports and focuses on the strategies developed within the goal generation and management framework (GGM) of CogX. It discusses three different such strategies and relates it to the state of the art.

Executive Summary

In this deliverable we report on our efforts to combine curiosity- and task-driven behaviour in our systems resulting in an active management of different goals a system featuring self-understanding and self-extension might have. The report presents and discusses the final version of the framework for goal generation and management (GGM) we have developed as part of our efforts in WP1 of CogX. The approach allows the systematic integration of different drives within the planning and execution framework (WP4). "Intention changes", in the notion of our CogX theory, are the result of either *tasks* being given to the system to be accomplished, or new *opportunities* that arise from an extended understanding of the world and from identified *gaps* in the knowledge about it, leading to curiosity-driven goals. This report complements **DR 1.1** which introduced the general architecture for goal generation and management. It extends on that earlier report by presenting and discussing the different strategies that have been proposed, implemented, and analysed within CogX by means of the two validation scenarios Dora and George. We present two rather heuristic strategies tailored to the needs of Dora and George, respectively, and a third novel strategy featuring support for task deadlines and principled selection of opportunities. This last strategy facilitates an informed trade-off of curiosity-driven self-extension and task accomplishment taking into account the temporal constraints that come with tasks assigned to a robot. It enables the robot to be engaged in scenarios where it is expected to support humans in a task-driven way, while still following its drive to extend its own knowledge about the world.

Tasks involved

Task 1.2 and 1.4 play the central role for the actual work on "Opportunistic & interleaved self-extension" in the final year of CogX, in close collaboration with Tasks 4.3 and 7.7.

Role in CogX

The goal generation and management (GGM) framework has played a key role within CogX from the very beginning. Due to the central role of planning in CogX systems the decision about *what* to achieve plays an important role, particularly when it comes to switching between curiosity- and task-driven behaviour, and a combination thereof.

Contribution to the CogX scenarios and prototypes

The GGM framework is employed in both the Dora and George scenarios. In both it plays the central role to control the "macro behaviour" of the respective systems. While in the George scenario, GGM moderates curiosity-driven and tutor-driven (corresponding to task-driven) interactive learning, in Dora it integrates curiosity-driven exploration with the accomplishment of externally assigned tasks.

1 Planned work

In this report period the focus with regard to goal generation and management was on combining – and switching between – curiosity-driven and task-driven behaviour to facilitate self-extension and task accomplishment at the same time. In particular, according to the work plan (page 28) Task 1.4 is concerned with the *"interaction between selected intentions, mechanisms for planning behaviour and the mechanisms for executing behaviour plans."* In this endeavour it focuses on the challenges of *"taking advantage of self-extension opportunities during an otherwise task-driven activity."* Additionally, work associated with Task 1.2 to *"explore designs for multiple concurrent desire instance generators combined with filter mechanisms"* is reported in this deliverable.

2 Actual work performed

The work actually performed followed the work plan for the given periods almost to the letter, with minor delays in the actual delivery. We have extended the overall architecture (see Sec. 2.1) to accommodate different types of curiosity (Sec. 2.2.1) and task-driven goals (Sec. 2.2.2), and implemented appropriate filter mechanisms (Sec. 2.3.1). We have revisited the different implementations that stem from fulfilling the requirements of the different scenarios and integrated them into a unified schema and implementation (Sec. 2.3.3 and 2.3.4).

Most effort in this last reporting period went into the implementation of a new activation strategy called "Opportunistic Goal Expansion (OGE)" (Sec. 2.3.5). This novel strategy for the first time really allows the scheduling of opportunistic, curiosity-driven behaviour alongside the accomplishment of assigned tasks within the CogX domains. As part of the efforts in Task 1.4 we have therefore extended the planning framework mainly developed in WP4, integrated it with the goal generation and management architecture, and developed an evaluation scenario on the basis of the Dora domain used for case-based analysis of the implementation. Additionally, we have discussed our work in the context of recent developments of state of the art by related efforts. This discussion concludes this document in Sec. 3.

2.1 The CogX Motivation Architecture for Goal Generation and Management

Throughout the CogX project, we adopted a planning-central approach to self-understanding and self-extension, resulting in the CogX Layered Architecture Schema (CLAS) sketched in Fig. 1(a) and described in greater detail in **DR 7.3**, **DR 7.4**, and **DR 7.5**. Goals are being generated, selected, and

managed by the *goal generation and management (GGM) framework* depicted in Fig 1(b). In the general aim of CogX this framework allows to trade-off between *task-driven* and *curiosity-driven* behaviour. Hence, the framework's purpose is to generate either *task goals* resulting from a verbalised intention of a human (e.g. to find and deliver a named object) or from an analysis of the robot's current beliefs to identify *gaps* of knowledge (see **DR 1.5** for a taxonomy and detailed discussion of knowledge gaps in CogX) that subsequently give rise to *epistemic goals*. The framework is employed in both the Dora and the George validator systems with minor adaptation and devises tasks for the *switching planner* framework.

The GGM taken together implements the *drives* of a CogX system which can be summarised as to autonomously explore the world (curiosity-driven) to gain the most benefit in short time, and to obey human orders (task-driven). These drives manifest themselves in the different goal generators that are implemented and the way a subset of these goals is chosen for accomplishment at any given time.

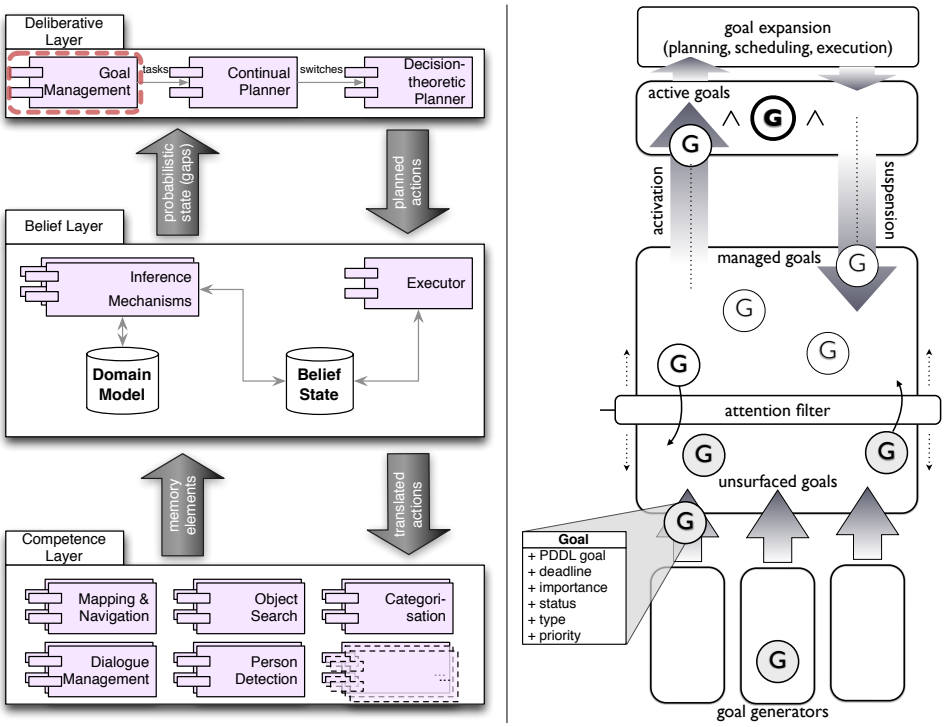
2.2 Generating Goals

At the bottom of Fig. 1(b) one can see the *goal generators*. They are responsible for populating the framework with potential goals that are then actively managed to decide which of those potential large set of goals are currently considered by the system. A goal is defined as a tuple

$$g \quad = \quad (\gamma_g, T_g, i_g, t_g, \Sigma_g, P_g)$$

with a dedicated *goal condition* γ_g to be reached by the plan. This is represented in PDDL and directly understood by the planning framework. Each goal also is assigned an optional *deadline* T_g indicating a temporal constraint[1] for the planner. The goal condition γ_g has to be true at the given time T_g. If no deadline is given then no constraint is present. Additionally to those parameters that control the planning process directly, some meta-management information is present with each goal as well: A positive *importance* i_g value is used to determine the relevance of a goal, with higher numbers indicating a higher importance. This value is considered when deciding which goals to pursue in different ways together with the *priority* P_g which is a discrete value, being either *low*, *normal*, or *high*. The *type* t_g of the goal indicates the origin of the goal (see below) and its discrete *status* Σ_g that can yield specific information about the goal relevant for the management processes. Effectively, completed goals are removed from the framework after their successful completion.

[1]Time and durations, as far as planning and deadlines are concerned in this report, are quantified in *time step* units for practical reasons. These units are proportional to seconds in real time. Hence, a deadline of 1 corresponds to about 3 seconds in the latest Dora system.



(a) The CogX Layered Architecture Schema (CLAS). The GGM framework is located in the *deliberative layer* of the architecture and devises tasks for the *planner* based on an analysis of the probabilistic belief state. Due to its abstract nature it is employed both in Dora and George systems with minor modifications.

(b) A visualisation of the implemented design for a goal generation and management (GGM) framework. Goals are depicted as circles. Each goal g is composed of a PDDL definition of the associated *goal condition* γ_g to be reached, a temporal *deadline* T_g until when a goal has to be achieved, a positive *importance* i_g value, the *type* t_g of the goal, its discrete *status* Σ_g, and a discrete *priority* P_g. The priority of a goal is also visualised by the thickness of the icon's lines and font.

Figure 1: The goal generation and management (GGM) framework within CLAS.

2.2.1 Curiosity-driven Goals

In CLAS, the different modal sub-systems in the competence layer also explicitly represent the hypotheses formed about the world (for instance so-called placeholders in the spatial representation of Dora) and encode the uncertainty about specific entities or concepts in a probabilistic framework. Hence the probabilistic belief state comprises a number of *gaps* of the robot's knowledge about the world that subsequently give rise to epistemic goals which are generated by individual goal generators, one for each type of gap that the system is expecting. In CLAS, these goal generators exploit do-

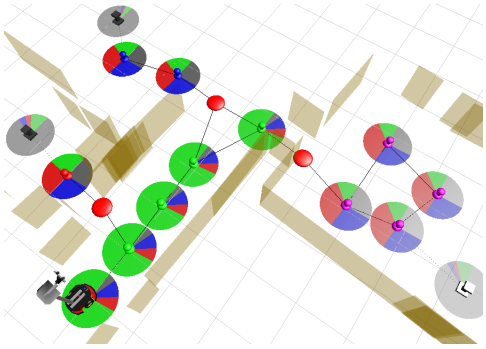


Figure 2: Places (coloured) and hypothesised placeholders (grey) in Dora's topological map.

main knowledge to monitor the belief state in order to identify such gaps and subsequently generate goals to fill them. For example, in George, there are dedicated goal generators that monitor beliefs about objects in the scene for confidence about a number of different features, such as colour, shape, or identity. If the robot is uncertain about any of these for any object, it detects this lack of confidence as a gap and generates a corresponding goal g_c (e.g. γ_{g_c} =`(colour-learned <belief-id>)` with `<belief-id>` being an unique reference to the belief about the object). Another example, this time taken from the domain of Dora, is the exploration of *placeholders*. These are hypothesised places in the topological map of the mobile robot that are generated through an analysis of the occupancy map. Fig. 2 depicts an exemplary map produced by Dora with hypotheses generated by the mapping and navigation components (cf. Fig 1(a)).

If such a hypothesis is spotted within the belief state it gives rise to a goal g_p with γ_{g_p} =`(= (status <belief-id>) trueplace)`. In both cases, the respective epistemic goal originates from a detected gap of knowledge and will – if scheduled for planning and execution – lead to curiosity-driven exploration [12] and learning [22] of our robots[2].

2.2.2 Task-driven Goals

Besides the above introduced curiosity-driven goals, our robots also feature support for task-driven behaviour, i.e. tasks that they have been commanded by a human to achieve. Of course, such goals can indeed also lead to gaps being filled whenever this is required for a task to be achieved. For example, if Dora is given a task to find a box of cereal in a house this will drive her to also explore placeholders in order to find the object. But in

[2] A Dora video of such curiosity-driven behaviour is available from `http://youtu.be/1o43nnY_e2Y`

contrast to curiosity-driven behaviour, this exploration is directly caused by the actual task given to the planning system.

In our system we employ abductive reasoning to infer a human's intention in situated dialogues in order to translate them into the respective goal if they relate to a task the robot can accomplish. The number of different tasks our systems can understand and achieve is still limited, but the principle can easily be extended to more of them. We have specialised an abstract implementation of a generic goal generator to implement components for specific types of goals inferred from the respective intention. More details on the generation of goals from interpreted intentions through dialogue can be found in [16, 22, 21].

In general, task-driven goals do not at all differ from curiosity-driven ones. Just by assigning different importance i_g and priority P_g a difference in managing such goals in comparison to curiosity-driven goals can be achieved as will be detailed in Sec. 2.3. An example of a task-driven goal g_t taken from the Dora domain is

γ_{g_t} = `(exists (?o - visualobject) (and (= (label ?o) magazine) (reported ?o)))`

which requests Dora to find an object of type "magazine" and report back its position to the human who expressed the intention that the robot should engage with this task.

2.3 Managing Goals

Once goals are created and populate the GGM framework the management of these goals has to take place. By "managing" we refer to the processes and mechanisms that filter and choose a subset of all the potential goals for actual accomplishment – called "activation" – by the robot at a given time. Inspired by the work in [3], we adopted a multi-stage process to the management of goals resulting in a disjunct division of all the goals in the framework into individual subsets which are visualised in Fig. 1(b).

2.3.1 Attention Filter

We work from the assumption that as a goal passes up through the framework from a generator and influences a systems behaviour, it is inspected by processes of greater and greater computational complexity. Therefore the lower strata of the framework exist to protect these processes (and thus overall system resources) from having to consider more goals than is necessary (where this could be a contingent judgement). The main mechanism in the framework for protecting the management processes is the attention filter. This is a coarse barrier which uses simple, fast processing to let some goals through to the management mechanisms whilst blocking others. Goals which make it through this filter are described as *surfaced*, forming the subset $G_s \subseteq G$ of all possible goals G. Thus the goals which fail to pass the

filter are referred to as *unsurfaced* $G_u \subseteq G$ with $G_s \cap G_u = \oslash$. In Fig. 1(b) all goals shown with a shade of gray are unsurfaced and are not considered for any further processing. It shall be noted that filtering, and likewise managing in higher stratas of the framework, are non-monotonic processes which continuously monitor all goals to potentially relocate them within the framework.

In order to keep the computational complexity manageable (we might have goals in the order of high tens or even hundreds within our system, originating from different gaps of knowledge and tasks assigned to the robot) we look at the goals individually and not within context to one another at this stage. Hence filtering can be dependent on the overall system's context, but it does not compare the goals with each other. The filters implemented within the George and Dora systems are mostly based on the type t_g, the importance i_g, and the status Σ_g of each individual goal. Different filters can be chained together so that a goal has to pass all of them in order to become surfaced. The list of implemented goal filters includes:

- type-based manual selection which allows the supression of specific types of goals, mainly for system configuration purposes;
- status-based supression of goals that have failed to be accomplished a number of times (to sort out invalid goals); and
- importance-based supression of goals that are heuristically assessed as not being beneficial enough to be considered for further processing.

2.3.2 Activation

Goals that are surfaced G_s are subject to more computationally complex management that involves *comparison* between different goals and a more informed selection of which goals shall become *activated*. The activated goals G_a again form a subset of the surfaced goals so that $G_a \subseteq G_s$ and $G_a \neq \oslash$, i.e. there is at least one of the surfaced goals active at any given time.

In the course of the CogX project we have implemented three different strategies to realise the activation process in order to select the goals to be included in G_a – all utilising the very same GGM framework, but allowing for different drives to be chosen, each appropriate to the respective domain. The three strategies are built upon each other with the later ones subsuming functionality of the earlier ones. The first one to be presented in Sec. 2.3.3 selects a *single* goal for activation at any given time taking into account the predicted information-gain and the respective costs of each individual goal. It is tailored to the requirements of purely curiosity-driven behaviour as presented in the Dora scenario of year 1 of the project. The second one (Sec. 2.3.4) adds priority-based selection to the first strategy to realise sequential tasks of interactive learning in the George scenario facilitating

different modes of initiative. Finally, in the third strategy (Sec. 2.3.5), we present a method to accomplish task-driven goals given temporal constraints while still exploiting opportunities to extend knowledge in a curiosity-driven way. The final strategy selects one of the task-driven goals similarly to the second and first strategy, but expands the set of activated goals G_a by including other curiosity-driven goals in an opportunistic way.

2.3.3 Heuristic Gain-Cost Trade-off (GCT)

In [10] and [12] we have presented the details of autonomous extension and exploration realised by means of the GGM framework and the heuristic gain-cost trade-off (GCT) strategy, and presented results of its application in the Dora scenario. We shall now only briefly revisit the main aspects of this work and its results in the context of the overall approach discussed here.

The GCT strategy evolves around the idea that a single goal is selected to be activated at any given time, while the other goals remain surfaced. The goal is chosen based on trade-off between an estimated *information gain* ι_g and the respective *costs* c_g the goal would incur. Both factors are compiled into the goal's importance $i_g = \frac{\iota_g}{c_g}$ and the goal with the highest i_g is chosen for activation. Naturally, this strategy is best suited for curiosity-driven *epistemic goals* which actually yield an information gain assigned to each individual goal. In order to assess this strategy we implemented two curiosity-driven goal generators for two types of knowledge gaps. One for the placeholders occurring in the map, as introduced already in Sec. 2.2.1, and another one that gave rise to epistemic goals to determine the category of rooms found in the map (cf. Fig 3(b) for a map with two rooms separated by a doorway). When it comes to the actual selection of goals, the costs and the information gain for each goal have to be estimated. The cost metric to be minimised by the planning algorithm is the overall duration to achieve a goal. Hence, all actions are assigned costs in the domain definition according to their average execution time based on our own experience. While this allows the costs for all the goals to be computed by querying the planner directly, a more efficient estimate is usually preferable to reduce computational load. Hence, for specific types of goals, dedicated estimators have been implemented. For instance, the cost of exploring a placeholder is estimated as the distance in the topological graph from the current position to that placeholder. If no such estimator is available for a specific type, the planner is queried for the costs.

However, the quantification of the information gain for different kinds of goals is a not as straight-forward as the costs. This problem has also been discussed in [12]. While there exist suitable heuristics for quantifying the information gain of goal of specific kinds (e.g. the goals to explore placeholders based on the information gain in a robot's SLAM map [1]) the problem becomes even harder if one aims to compare goals of different

types. For instance, if no further constraints are given, it is not clear if exploration of another placeholder or finding out about the category of a room (whether it is a kitchen or an office) yields more information. This has been discussed as an inevitable challenge of any purely curiosity-driven goal selection mechanism that has to trade-off different types of goals. We have evaluated the heuristic chosen in [12] and were able to show in [10] that this strategy can improve overall performance in exploration compared to a conjunction of all the individual goals in terms of planning and execution time.

In summary, the GCT strategies enables GGM to select the single most promising goal at any time being precisely informed about the costs and weighing this up against the heuristically predicted gain. It suffers from the principally not well defined comparison between different types of goals, a drawback which is addressed by the dynamic prioritisation strategy presented next.

2.3.4 Dynamic Prioritisation (DP)

While the GCT strategy proved suitable for purely curiosity-driven exploration, the interactive nature of the George scenario demanded a slightly modified approach, still implemented within the general GGM framework using a combination of appropriate filter and strategy mechanisms. The George system is a robot which interacts with a single human in order to learn the colours, shapes and identities of objects on a tabletop [23] (see also **DR 7.5**). Because all interaction is done through dialogue, the overall behaviour of the robot is necessarily limited to a *sequence of single tasks*, where these tasks are derived from questions or commands posed by either the human or the robot. These two sources of tasks roughly correspond to task-driven and curiosity-driven goals: task-driven goals are created to respond to the human's learning and questioning utterances ("*this is a red box*", "*what colour is this?*"), curiosity-driven goals are created to provide George with more information about particular objects ("*what colour is the box?*"), or the world in general ("*please show me a compact object*").

The constrained, sequential nature of George's behaviour means that the goal activation problem is reduced to selecting the most appropriate single goal at any time. Whilst this is a simpler problem in many respects than appears in Dora, the fact that there are still many different things George can do at any point means that developing an effective management and activation scheme is still a challenging task. We have approached this using a *dynamic priority filter with delays*. This is based on a priority hierarchy which reflects the different drives George has. The highest priority drive is to respond to the human (in a task-driven fashion). This is followed by the curiosity drive to fill gaps in knowledge via *extrospection* (i.e. inspecting the world external to the agent). At the lowest level is the (curiosity) drive to

fill knowledge gaps by *introspection*.

When goals are generated they are assigned a priority level, P_g, based on their source. The attention filter in George is also associated with a priority level, P_f. When a goal g reaches the filter, the filter inspects the priority level and performs one of the following actions:

- If $P_f > P_g$ then g remains unsurfaced.
- If $P_f == P_g$ then g surfaces.
- If $P_f < P_g$ then g surfaces, P_f is set to P_g, and all currently surfaced goals (which necessarily have priorities lower than P_g) are unsurfaced.

The effect of this approach is that only goals which are of the highest available priority are allowed to be surfaced. When a goal is unsurfaced due to completion or cancellation then this process is (approximately) run in reverse, allowing the priority of the filter to decrease to the highest remaining priority.

Goals which surface are managed using the heuristic trade-off approach described in Section 2.3.3, selecting for only the highest ranked goal. This means that within a priority level George always performs the behaviour which provides the most information at the lowest cost. This is only used at the curiosity levels, as it is not possible in our scenario for George to be given two task-driven goals at once (although there is no reason the system could not support this).

In George, the motivation framework is also responsible for ensuring that interactions with the robot are structured appropriately. Whilst the selection of only a single, dynamically-prioritised, goal at any time contributes to this, we have also found it necessary to add a *surfacing delay* to the filter. This mechanism is able to delay the surfacing of particular classes of goals until a given duration after the completion of the previous goal. We apply delays only to curiosity-driven goals. This ensures that George allows its interlocutor sufficient time to engage it as they desire (i.e. asking questions or issuing learning instructions) before it starts generating utterances of its own.

2.3.5 Opportunistic Goal Expansion (OGE)

The final strategy called "Opportunistic Goal Expansion (OGE)" improves on the two introduced above (i) by activating more than just one goal at a time allowing for more efficient behaviour, (ii) by its more informed selection process that takes the dependencies between goals into account, and (iii) by the additional support for temporal constraints to goals to effectively combine task-driven with curiosity-driven opportunistic behaviour. It allows the CogX robots to achieve specified tasks within the given time constraint

(deadline) while at the same time taking opportunities to self-extend their knowledge.

For this approach to work we built upon the two other strategies, but involved the planning process in the actual selection of the goals by defining activation as follows:

1. Like in "Dynamic Prioritisation" (Sec. 2.3.4) curiosity-driven goals are assigned a lower priority than task-driven goals. However all goals are surfaced and considered for activation. Again the intention is that commands given by a human overrule any self-extension drive of the robot. All curiosity-driven goals $g_c \in G^{cur}$ are assigned the lowest priority $P_{g_c} = low$.

2. Consequently, all task-driven goals $g_t \in G^{task}$ are assigned higher priorities $P_{g_t} \in \{normal, high\}$ than the curiosity-driven goals.

3. Based on the set of all surfaced task-driven goals G^{task} the same selection principle as in the dynamic prioritisation strategy (described in Sec. 2.3.4) is employed to select a subset of relatively highest priority goals. For instance, if G^{task} contains goals with *normal* and *high* priority, only the ones with *high* are further considered for activation. From this subset, a *single* goal g_t^* is selected using an *ad hoc* strategy. In our current implementation they are selected in order of appearance. For task-driven goals this an appropriate strategy if one assumes that the robot shall carry out tasks in the same order it has received them from a commanding human. Other strategies are yet to be explored. The single chosen goal g_t^* is considered as a *hard goal*, i.e. a goal that has to be accomplished by any plan generated. Formally, a hard goal has an infinite importance $i_{g_t^*} = \infty$. Additionally, the hard goal is assigned a time constraint in form of a *deadline* $T_{g_t^*}$. The actual deadline depends on the type of the goal and is currently arbitrarily determined by the user. It defines the upper boundary of the summed duration $D_\pi = \sum_{a \in \pi} d(a)$ of all planned actions a to achieve the goal (with $d(a)$ being the duration of action a in plan π that accomplishes g_t^*). A valid plan has to achieve the goal within the time constraint, i.e. $T_{g_t^*} \leq D_\pi$.

4. All curiosity-driven goals $g_c \in G^{cur}$ with priorities lower than *high* are taken as *soft goals*, i.e. goals that have a certain importance $0 < i_{g_c} < \infty$. The nature of soft goals is that they *can* be accomplished by a plan, but they can also be *forfeited* if either they cannot be accomplished given other constraints (e.g. a deadline of a hard goal), or they are not important enough given the costs their accomplishment would occur. Hence, the importance values assigned to the soft goals in relation to the incurred costs determine which of the soft goals will actually be

activated and which will be suppressed. The importance value of all the soft goals are computed similarly to the strategy introduced in the Sec. 2.3.3, taking into account the type and the estimated information gain.

This strategy actually selects the goals to be activated by presenting the decision on which goals to activate as an oversubscription planning problem. It defines one task-driven goal as a hard goal with a given deadline and all the surfaced curiosity-driven goals as soft goals with respectively assigned importance values. This problem can be solved by a planner that provides support for deadlines and soft goals. The planner will make the decision about which soft goals shall be accomplished by the final plan taking into account their incurred costs in relation to the importance while ensuring that the hard goal is accomplished within the time constraint. Once a plan has been found, all soft goals that are accomplished by the plan and the hard goal are activated. Instead of employing a third-party planning framework that would generally fulfil our requirements, like SAPA which is employed in some work related to ours [19], we decided to further extend the Fast-Downward planner [13] which we have already adapted for the implementation of the CogX switching planner [7] to support probabilistic belief states and assumptive actions. Adding combined deadline and soft goal support to this planner facilitates more direct applicability in CogX domains. In the following, the required extensions to the Fast-Downward planner are introduced.

Planning with deadlines We built a simple extension that can handle a subset of planning problems with deadlines upon the classical Fast Downward planner. A classical planning problem with action costs Π consists of a initial state s_0, a set of actions $a \in \mathcal{A}$ with associated costs $c(a)$ and a goal γ. The planning problem involves finding a sequence of actions $\pi = [a_0, \ldots, a_n]$ that reach the goal from the initial state. A (non-parallel) planning problem *with deadline* T assigns a *duration* $d(a)$ to each action and involves finding a plan π s.t. $D_\pi = \sum_{a \in \pi} d(a) \leq T$.

In the simplest case, action durations can be directly mapped onto action costs. In this case, planning with a deadline T reduces to finding a plan with a cost bound $C = T$. We use a heuristic forward search planner that employs a *weighted* A^* algorithm to find good plans efficiently. Weighted A^*, combined with an *admissible* heuristic, guarantees that for a given weight w any plan found by the search is at most a factor of w more expensive than the optimal plan. In general, higher weights improve planning speed at the expense of plan quality.

Algorithm 1 shows a simplified version of weighted A^* search for trees[3].

[3]In the planner, a more complicated version applicable to graphs is used, but those details are not relevant to the work described here.

Algorithm 1 WA^*-Planner

```
1: Input: Planning problem Π = ⟨s_0, A, γ⟩, weight w
2: Output: Plan π or failure
3: g(s_0) ← 0
4: Queue ← ⟨s_0, g(s_0) + wh(s_0)⟩
5: while s ← Pop(Queue) do
6:   if CheckGoal(s) then
7:     return Backtrack(s)
8:   end if
9:   for a, s' ∈ Successors(s) do
10:    g(s') ← g(s) + c(a)
11:    Queue ← ⟨s', g(s') + wh(s')⟩
12:  end for
13: end while
14: return failure
```

We initialize a priority queue with the initial state (line 4) and the priority function $f(s) = g(s) + wh(s)$. We take the lowest cost state from the queue and check whether it satisfies the goal. If yes, we backtrack and return the path that lead to this state (lines 5-7). Otherwise, we generate all states that can be reached from the current state via an action a and put them into the priority queue (lines 9-11). If the priority queue is empty and no satisfying state has been found, we return failure.

We can make two simple modifications to make sure that the found plans respect the deadline T, which are shown in Algorithm 2. First, we check if the g-value of a state exceeds T (lines 6-7): if this is true, the plan that reaches this state will already have exceeded the deadline. This alone is sufficient to make sure that the planner only returns a plan that respects T if one exists, but is throwing away information. If our heuristic function h is admissible, then $f(s) = g(s) + h(s)$ is a lower bound on the actual costs of a plan involving this state. Thus, if this value is already greater than T, we can refrain from putting it into the priority queue (lines 14-15).

The problem becomes more complicated if we include soft goals in the planning framework. A *soft goal* is a goal condition γ_i together with an *importance* i_{γ_i} and the planning task is to find a plan π that minimizes the cost of the plan plus the priorities of the unsatisfied soft goals. The easiest way to solve problems with soft goals, is to compile them back into a classical planning problem as described by Keyder and Geffner [14]. This involves the creation of *forfeit*-actions for every soft goal, that have costs equal to the goal's importance. Now, the direct correspondence between action costs $c(a)$ and action durations $d(a)$ no longer holds: The forfeit actions have non-zero costs, but zero duration: not satisfying a soft goal does not take any time. The same holds true for assumptive actions as used in our continual

Algorithm 2 Deadline-Planner-Simple

```
1: Input: Planning problem Π = ⟨s_0, A, γ⟩, weight w, deadline T.
2: Output: Plan π or failure
3: g(s_0) ← 0
4: Queue ← ⟨s_0, g(s_0) + wh(s_0)⟩
5: while s ← Pop(Queue) do
6:   if g(s) > T then
7:     continue
8:   end if
9:   if CheckGoal(s) then
10:    return Backtrack(s)
11:  end if
12:  for a, s' ∈ Successors(s) do
13:    g(s') ← g(s) + c(a)
14:    if g(s') + h(s') ≤ T then
15:      Queue ← ⟨s', g(s') + wh(s')⟩
16:    end if
17:  end for
18: end while
19: return failure
```

planner [7, 2].

Listing 3 shows the modifications to our search process that are required to deal with these differences. In addition to the costs g, each state s also keeps track of the elapsed time $t(s)$ (lines 4, 15) and uses this to abort search if it exceeds the deadline (line 7). In addition, we can no longer use the heuristic $h(s)$ to get an lower bound estimate on the remaining time. We use a modified heuristic function h^t that uses durations instead of costs to compute the heuristic value of a state and use this function for the early pruning of states (line 16). While this doubles in the worst case the number of heuristic calculations that are required, experiments suggest that the savings from pruning states that are known to exceed the deadline easily makes up for this.

Validation Runs We have tested the OGE strategy in the context of a modified Dora scenario using the standard planning domain and system instantiation described in detail in **DR 7.3**, including the planning domain in DTPDDL. Here, we gave Dora a number of *patrolling tasks* which basically involve travelling around a *partially explored* environment to repeatedly inspect, i.e. visit, a number of designated places within a certain time constraint. This task is informed by typical security applications that require a (human) guard to "check in" at specific spots according to a given schedule. Accordingly, Dora will always have one task-driven goal g_t^* at a time

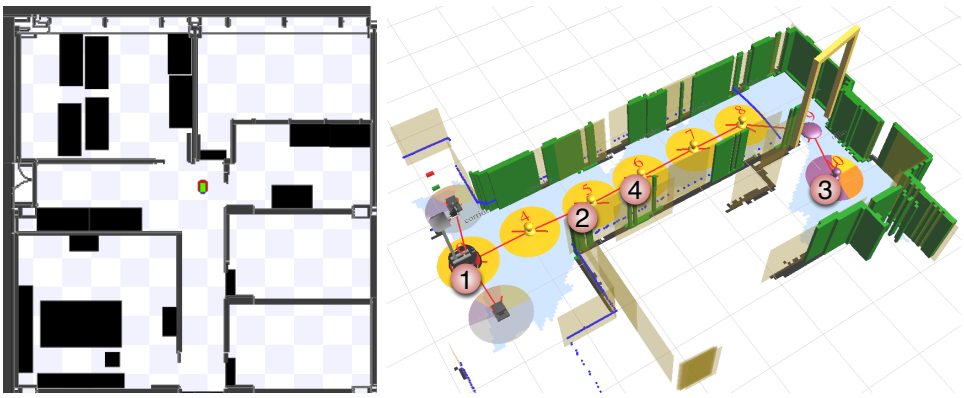
Algorithm 3 Deadline-Planner

```
1: Input: Planning problem Π = ⟨s_0, A, γ⟩, weight w, deadline T.
2: Output: Plan π or failure
3: g(s_0) ← 0
4: t(s_0) ← 0
5: Queue ← ⟨s_0, g(s_0) + wh(s_0)⟩
6: while s ← Pop(Queue) do
7:    if t(s) > T then
8:       continue
9:    end if
10:   if CheckGoal(s) then
11:      return Backtrack(s)
12:   end if
13:   for a, s' ∈ Successors(s) do
14:      g(s') ← g(s) + c(a)
15:      t(s') ← t(s) + d(a)
16:      if t(s') + h^t(s') ≤ T then
17:         Queue ← ⟨s', g(s') + wh(s')⟩
18:      end if
19:   end for
20: end while
21: return failure
```

to reach and inspect a given place and it will have a deadline $T_{g_t^*}$ assigned to this task that it has to respect. However, as the environment is partially unexplored, Dora has the opportunity to extend her knowledge by pursuing other curiosity-driven goals $g_c \in G^{cur}$, which will eventually lead to a map extended to the one previously known and prepare her for future tasks, e.g. to patrol a larger area or finding objects in newly discovered rooms [9].

Fig. 3(a) visualises the simulation environment we set up to test the OGE extension with the integrated Dora system. A simulation environment has been chosen to ease reproducibility and controllability for the tasks at hand. The figure also indicates the starting position of the robot in each of the runs. This position corresponds to the location of the robot in Fig. 3(b) which visualised the robot's knowledge at start time, i.e. the partially explored map comprised of places (disks with numbers up to 10 in the figure), connection between those places (red arrows), a door, walls, and placeholders as introduced in Fig 2.

As said before, all the placeholders give rise to curiosity-driven soft goals with a low priority to form G^{cur}. Additionally, the one task-driven goal g_t^* is given to subsequently patrol to places 0, 5, 10, and 6, an order also indicated by the coloured superimposed balls in Fig. 3(b). This is implemented using a specialised goal generator that emits goals of type $t_{g_t^*} = patrol$. Whenever



(a) The stage simulation map used including the robot's starting position.

(b) The robot's representation about the partially explored environment. The superimposed balls with numbers (1-4) indicate the order of patrolling tasks issued to the system to be accomplished sequentially.

Figure 3: The situation at start-up time for experiments on opportunistic goal expansion in the Dora patrolling scenario.

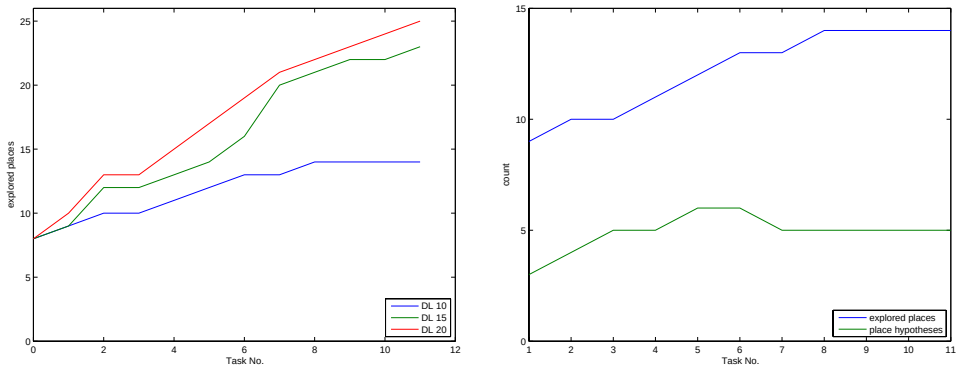
the current task to patrol a given place has successfully been accomplished, this generator emits a new goal to patrol the next place in the given order, looping through the list of given goals over and over again, leading to an assignment of places to patrol to respective goals shown in Tab. 1.

The deadline $T_{g_t^*}$ assigned to the goal is varied to study its effect. In our experiments we chose three different parameterisations $T_{g_t^*} \in \{10, 15, 20\}$ resulting in three different conditions DL10, DL15, and DL20, respectively. The importance of all soft goals $g_c \in G^{cur}$ is fixed to $i_{g_c} = 500$. This high value effectively enforces the OGE strategy to activate as many soft goals as possible under the given deadline constraint $T_{g_t^*}$.

To study the effect of the different conditions DL10, DL15, and DL20, the size of the explored map has been measured in "number of places explored". In each run, the system starts off in a partially explored map comprised of a total of 8 places (of which the place with ID 9 is a door, see Fig. 3(b)). Then it is set to accomplish its patrolling tasks and take the opportunities to explore placeholders. It is expected that higher values for the deadline of the patrolling goals give the robot more time to exploit exploration oppor-

Order	Place	Task no.										
see Fig. 3(b)	ID	1	2	3	4	5	6	7	8	9	10	11
1	0	X				X				X		
2	5		X				X				X	
3	10			X				X				X
4	6				X				X			

Table 1: The places to patrol in the respective task corresponding to the results in Fig. 4(a).



(a) Number of explored places in the combined patrolling exploration task under varied deadline conditions.

(b) Number of explored places and still existing place hypotheses in condition DL10.

Figure 4: Progress of exploration in opportunistic goal expansion.

tunities, effectively leading to a faster exploration of its environment. This hypothesis is confirmed by the results shown in Fig. 4(a) which shows results for a total of 11 iterations of patrolling tasks given to the system according to Tab. 1.

In all three conditions, the robot adhered to the deadline, i.e. it generated plans to patrol the places according to the schedule. However, it is apparent that after 11 iterations in the different conditions the amount of explored space explored map is higher the higher the deadline in the respective condition. The map contains only 13 explored places in condition DL10, whereas the strategy yielded 23 and 25 explored places in conditions DL15 and DL20, respectively.

It can also be seen in Fig. 4(a) that in condition DL10 the size of the map has reached a plateau from task 8 onwards. The number of explored places does not increase anymore, even if the system is left running for more iterations, because all placeholders that can be opportunistically explored with the tight deadline have been explored, and the constraint simply does not allow the system to extend the map any further as it is bound by its high-priority task-driven goal. Fig. 4(b) shows that indeed there are still placeholders left to be explored, i.e. soft goals to be accomplished, but OGE does not include any of them due to the deadline.

On the contrary, in conditions DL15 and DL20 the plateau has not yet been reached and the opportunistic exploration of the robot still continues even after iteration 11 because there are still a number of exploration goals that can be accomplished while pursuing an assigned task without violating the deadline. The final map in condition DL20 is visualised in Fig. 5 which contains a total of 25 places. Therefore the OGE strategy allowed the robot to opportunistically add another 17 places to its topological map and to discover two new rooms whilst accomplishing all of its patrolling duties.

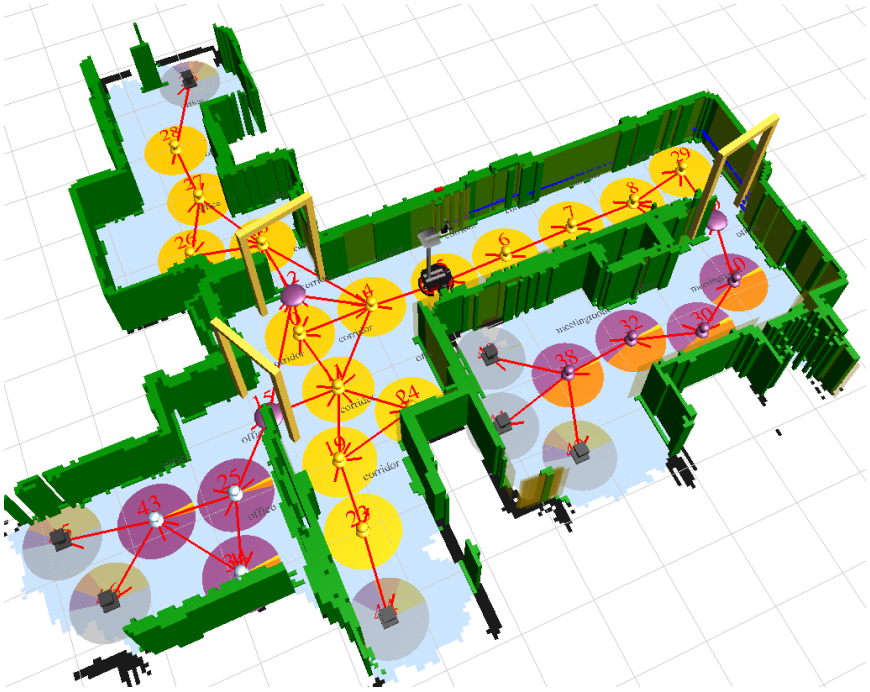


Figure 5: Final map after interleaved task- and curiosity-driven goal management in condition DL20.

3 Relation to the state-of-the-art

Although the problem of generating and managing goals for an integrated system has not been studied widely by the AI community, there is a body of work to which our work relates. The work of Coddington [5] supports the link between motivation and planning complexity. In a limited setting she compared two approaches to generating goals for an agent: reactive generation, and encoding all the system's goals as resources in its planning domain. This work demonstrated that by only using reactive goal generators the system could not guarantee to satisfy all of its desires, as the effects of actions in current or future plans were not reasoned about by the generators. Coddington views using a planning process to decide which goals should be pursued as a solution to this. As a planning approach would consider all interactions between possible goals and current actions it prevents potentially deleterious situations occurring. However, Coddington demonstrated that the computational cost of encoding all possible goals in a planning problem prevented the system tackling problems beyond a certain size. This is clear motivation for filtering and prioritisation mechanisms prior to activation, provided that they are appropriately designed to ensure mission-critical goals surface appropriately.

The problems and benefits of autonomous goal generation in an integrated system setting are demonstrated by the work of Schermerhorn et al. [19]. They present a system that can use the preconditions and effects of planning actions to generate new goals for a system at run-time. This approach has clear benefits in unpredictable worlds and would fit cleanly within a generator in our framework. [20] highlights the problems of treating goal activation as a rational decision making problem. The primary difficulty is getting reliable, comparable models of actions and the environment. Our systems face this problem when attempting to compare measures of information gain from different types of curiosity-driven goals. In [24] the same team also show how, in an open-world domain, new observations can be the source of new goals. They distinguish between hard and soft goals, and associate goals with deadlines, but they do not consider other management criteria.

Klenk et al.'s [15, 18] framework for goal-driven autonomy is the work that most closely aligns with our framework. One difference between the designs is that they introduce an additional explanation component which can be used to generate goals from discrepancies between expectations and observations. For us this is just an example of the kind of behaviour a goal generator could demonstrate, and does not warrant special attention. They also do not address the difference between task- and curiosity-driven goals, which we see as key to the use of such a framework in the real world. Their framework has also not been tested outside of a naval warfare simulation, where it is used to control autonomous vehicles in large-scale deployments. This different application perhaps best explains our differences in focus.

Other frameworks for goal generation or management have been proposed previously. These approaches typically fail to make the distinction between generation, surfacing and activation, instead assuming that generation implies activation (ignoring the requirement to deliberate about possible goals). We can consider such approaches (in terms of goal generation and management) as implementations of a subset of our framework. Examples can be found in belief-desire-intention systems (e.g. [6]), behaviour-based systems (e.g. [4]), and reactive planners with goal management extensions (which represents perhaps the largest body of work on this subject) (e.g. [8, 17]). For a full review of related literature, see [11].

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] Francesco Amigoni and Vincenzo Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

[2] A. Aydemir, M. Göbelbecker, A. Pronobis, K. Sjöö, and P. Jensfelt. Plan-based object search and exploration using semantic spatial knowledge in the real world. In *Proc. of the European Conference on Mobile Robotics (ECMR 2011)*, Örebro, Sweden, sep 2011.

[3] L. P. Beaudoin and A. Sloman. A study of motive processing and attention. In A. Sloman, D. Hogg, G. Humphreys, A. Ramsay, and D. Partridge, editors, *Prospects for Artificial Intelligence: Proc. of AISB-93*, pages 229–238. IOS Press, Amsterdam, 1993.

[4] Joanna J. Bryson. The Behavior-Oriented Design of modular agent intelligence. In R. Kowalszyk, Jörg P. Müller, H. Tianfield, and R. Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, pages 61–76. Springer, Berlin, 2003.

[5] A. M. Coddington. Integrating motivations with planning. In *Proc. of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '07)*, pages 850–852, 2007.

[6] Michael P. Georgeff and François Felix Ingrand. Decision-making in an embedded reasoning system. In *Proc. of the 11th International Joint Conference on Artificial Intelligence (IJCAI '89)*, pages 972–978, 1989.

[7] M. Göbelbecker, C. Gretton, and R. Dearden. A switching planner for combined task and observation planning. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI 2011)*, page NA, 2011.

[8] Elizabeth Gordon and Brian Logan. Managing goals and resources in dynamic environments. In Darryl N. Davis, editor, *Visions of Mind: Architectures for Cognition and Affect*, chapter 11, pages 225–253. Idea Group, 2005.

[9] Marc Hanheide, Charles Gretton, Richard W Dearden, Nick A Hawes, Jeremy L Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Göbelbecker, and Hendrik Zender. Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011.

[10] Marc Hanheide, Nick Hawes, Jeremy L Wyatt, Moritz Göbelbecker, Michael Brenner, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Hen-

drik Zender, and Geert-Jan M Kruijff. A Framework for Goal Generation and Management. In *Proceedings of the AAAI Workshop on Goal-Directed Autonomy*, 2010.

[11] Nick Hawes. A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5-6):1020 – 1036, 2011. Special Review Issue.

[12] Nick Hawes, Marc Hanheide, Jack Hargreaves, Ben Page, Hendrik Zender, and Patric Jensfelt. Home alone: Autonomous extension and correction of spatial representations. In *2011 IEEE International Conference on Robotics and Automation*, pages 3907–3914. IEEE, May 2011.

[13] M. Helmert. The fast downward planning system. *J. Artif. Intell. Research*, 26:191–246, 2006.

[14] Emil Keyder and Hector Geffner. Soft goals can be compiled away. *J. Artif. Intell. Res. (JAIR)*, 36:547–556, 2009.

[15] Matthew Klenk, Matthew Molineaux, and David W. Aha. Goal-driven autonomy for responding to unexpected events in complex environments. *Computational Intelligence*, In Press.

[16] Geert-Jan M. Kruijff, Miroslav Janiček, and Pierre Lison. Continual processing of situated dialogue in human-robot collaborative activities. In *19th International Symposium in Robot and Human Interactive Communication*, pages 594–599. IEEE, September 2010.

[17] F. Michaud, C. Côté, D. Létourneau, Y. Brosseau, J. M. Valin, É. Beaudry, C. Raïevsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguère, M. A. Roux, S. Caron, P. Frenette, and F. Kabanza. Spartacus attending the 2005 AAAI conference. *Auton. Robots*, 22(4):369–383, 2007.

[18] Matthew Molineaux, Matthew Klenk, and David W. Aha. Goal-driven autonomy in a navy strategy simulation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[19] Paul W. Schermerhorn, J. Benton, Matthias Scheutz, Kartik Talamadupula, and Subbarao Kambhampati. Finding and exploiting goal opportunities in real-time during plan execution. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, pages 3912–3917, October 2009.

[20] Paul W. Schermerhorn and Matthias Scheutz. The utility of affect in the selection of actions and goals under real-world constraints. In *Proceedings of the 2009 International Conference on Artificial Intelligence (ICAI '09)*, pages 948–853, 2009.

[21] D Skočaj, M Kristan, A Leonardis, M Mahnič, A Vrečko, M Janiček, G.-J. M Kruijff, P Lison, M Zillich, C Gretton, M Hanheide, and M Göbelbecker. A system approach to interactive learning of visual concepts. In *Tenth International Conference on Epigenetic Robotics EPIROB 2010*, Örenäs Slott, Sweden, November 2010.

[22] Danijel Skočaj, Matej Kristan, Alen Vrečko, Marko Mahnič, Miroslav Janiček, Geert-Jan M Kruijff, Marc Hanheide, Nick Hawes, Thomas Keller, Michael Zillich, and Kai Zhou. A system for interactive learning in dialogue with a tutor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*, San Francisco, CA, USA, 2011.

[23] Danijel Skočaj, Matej Kristan, Alen Vrečko, Marko Mahnič, Miroslav Janíček, Geert-Jan M. Kruijff, Marc Hanheide, Nick Hawes, Thomas Keller, Michael Zillich, and Kai Zhou. A system for interactive learning in dialogue with a tutor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*, San Francisco, CA, USA, 25-30 September 2011.

[24] Kartik Talamadupula, J. Benton, Paul Schermerhorn, Subbarao Kambhampati, and Matthias Scheutz. Integrating a closed world planner with an open world robot: A case study. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, Atlanta, GA, July 2010.