Hierarchical reactive planning: Where is its limit?*

Cyril Brom
Charles University in Prague
Faculty of Mathematics and Physics
Malostranské nám. 2/25, Prague, Czech Republic
brom@ksvi.mff.cuni.cz

Abstract

Hierarchical reactive methods are very popular in the field of controlling complex artificial intelligent agents. In this paper, we argue that they cannot cope with human-like behaviour. We present a detailed analysis of behaviour of a relatively simple human-like artificial agent, an artificial gardener, whose action selection model is based on hierarchical reactive planning. It is shown that although the agent has no troubles with "survival" in a complex and dynamic environment, its behaviour is not believable in some situations. However, instead of rejecting the judged methodology, we propose how to extend it using certain features of other action selection models.

1 Introduction

There are two main characteristics that make hierarchical reactive methods of action selection popular in the field of controlling complex artificial intelligent agents. The first is a top-down recursive decomposition of top-level agents' behaviours to sub-behaviours or sequences of simple actions. This decomposition eases the design. The second is that an agent's decision procedure focuses attention only to most relevant goals, while ignoring the rest of them temporarily. All reactive methods allow for quick switching between tasks according to the changing environment and agent's internal drives. Consequently, reactive hierarchies reduce combinatorial complexity of control and still can cope with large, unpredictable, real-time environments.

We are working on a research and educational toolkit for prototyping human-like artificial agents, i.e. the agents with the objective to *imitate* behaviour of humans (*h-agents* in the following text) [Brom *et al.*, 2005]. One of our motivations on this research effort is to find an appropriate methodology for controlling h-agents. The methodology must allow an easy behavioural design and must be computationally effective. Thanks to the aforementioned advantages of reactive hierarchies, we turned our attention to this branch of methods. So far, we have prototyped several h-agents "living" in a family house utilizing reactive hierarchies in order to ascertain their applicability.

As was expected, partially owing to Bryson's analysis [2000], our h-agents had no troubles with "survival", that means with satisfying own needs. However, it was not always straightforward to design their behaviour so that it was *believable* enough. Consequently, h-agents did not behave naturally in some situations—i.e., they would not pass Turing test. Because the h-agents otherwise performed well, we aimed at isolating problems and extending pure hierarchical reactive approach, instead of rejecting it.

In this paper, we present observed limitations on believability and suggest how to overcome them. We present the results in a case-study example of an artificial gardener, whose behaviour is structured by so-called *simple hierarchical reactive planning* (S-HRP). For features that could extend this model, we seek both inside and outside of hierarchical reactive family.

In Section 2, we briefly introduce our toolkit and detail our motivation. Then, we describe morning tasks of a "natural gardener". This story represents desired behaviour. In Section 4, we formalize the S-HRP method and describe behaviour of the artificial gardener. In Section 5, we present the results, together with suggestions on extensions of S-HRP. At the end, we discuss applicability of the extended S-HRP considering related action selection models.

2 Motivation: Project Ents

Simulations of artificial humans are becoming increasingly more popular both in the academic and industrial domains. Typical applications include computer games, virtual storytelling, entertainment applications, military simulations and behavioural modelling (e.g. [Prendinger *et al.*, 2004]).

From the technical point of view, each artificial human is viewed as an autonomous intelligent agent [Wooldridge, 2002] that carries out a diverse set of goals in a large dynamic environment with the objective to simulate *believable* behaviour of humans; this agent is a so-called *h-agent*. One of the key issues in this research field is design of a mind of h-agents (i.e., a memory and a procedure that decides what to do next—an action selection algorithm).

Although various theoretical solutions of this issue have been proposed so far (e.g. [Newell, 1990]), and a lot of

* This research was partially supported by the Program "Information Society" under project 1ET100300517.

individual applications using h-agents and languages for their programming exist, it is hard to find any complex toolkit that would couple an artificial environment similar to natural world, a neat graphical user interface and a language for prototyping h-agents' minds by means of various different techniques. Such a toolkit would simplify development of h-agents, enable verifying theories and it could serve as an educational tool for students.

The project Ents [Brom *et al.*, 2005] is a first generation of a toolkit that addresses these issues. It provides:

- A customizable artificial environment similar to natural world, which allows the h-agent to carry out complex human-like tasks (among others eating, sleeping and going to toilet).
- E language, which that enables modelling of h-agents using various different techniques (including for example hierarchical reactive planning, hierarchical finite state machines, and classical planning).
- A linguistic module, which enables talking to h-agents (in Czech language).
- The tool allows for interaction between the h-agents, and between h-agent and a user-agent.

Nevertheless, a toolkit is not just a programming vehicle and a bundle of debugging tools. It is also a design methodology. Therefore, we are focused not only on how to program h-agents, but also on how to *design* their behaviour *simply*. We are now in the phase of evaluation of the first generation of the toolkit and of various models of action selection, while specifying requirements on the toolkit's second generation. Hence, we evaluate, whether hierarchical reactive planning is the methodology appropriate for h-agents domain; and that is the topic of this paper.

The artificial gardener, whose behaviour is observed in the case-study, is prototyped in the project Ents. A model of a "family-house" is used. A screenshot from the simulation is depicted in Fig. 1. More information on the project is available at: `http://ckl.ms.mff.cuni.cz/~bojar/enti/`.

3 The challenge: natural behaviour

This section describes a story of a typical human that spends his morning gardening. Behaviour of the artificial gardener is modelled according to this "natural model" and the course of resulting behaviour is compared to it. In what follows, the artificial gardener will be denoted as the *a-gardener* and its human model as the *n-gardener* (we will use masculine for the n-gardener, neuter for the a-gardener and feminine for both an artificial and a natural neighbour).

Behaviour is observed from the moment the gardeners are going to the garden to the moment they leave it. Two tasks are intended: watering and weeding. The scenario follows:

Because n-gardener knows that a garden hose is punctured, he decides to water by a can. He goes to a chamber for tools. Because he is intended to weeding afterwards, he does not find only a can, but also a bucket, a weeder and a little scoop. Then he takes all of that (the weeder and the little scoop in the bucket) and goes to the garden. He whistles from time to time for joy.

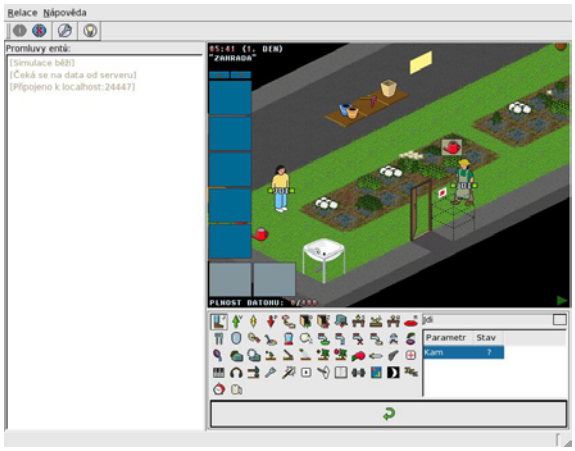

Figure 1: The GUI of the toolkit Ents. From the left: the user-agent and the artificial gardener.

In the course of watering, two events happen:

1. *A neighbour comes and asks him for a can. He promises her to bring the can after he finishes the watering.*
2. *Nature calls. He puts down the can and goes to the toilet. Then he returns and continues with the task.*

When he finishes watering, he goes to lend the can to the neighbour. Then he starts weeding.

In the course of weeding, following event happen:

3. *The neighbour returns the can. The n-gardener leaves the can as she has put it down.*

After he completes weeding, he puts all the tools into the chamber. Then he goes to eat to the dining room.

4 A-gardener: the action selection model

In this section we explain the action selection model of the a-gardener—*simple hierarchical reactive planning*, the S-HRP, and describe behaviour of the a-gardener. We remark that SHRP resembles the planning method of Bryson [2001].

4.1 Simple hierarchical reactive planning

S-HRP is a top-down, reactive method. The former means that the overall behaviour is decomposed into specific goals, which are recursively decomposed into smaller subgoals, until atomic actions are reached. The latter means that the next action an agent has to perform is not selected from a plan generated before an execution starts, but it is computed instantly by means of context-based triggers that continuously monitor an environment or the agent's internal drives. Reactive planners do not "look ahead"; instead, they compute just the next act in every instant. In S-HRP, the problem of what to do next is reduced to switching among sets of triggers associated with some subgoals according to changing circumstances.

S-HRP provides four behavioural structures: atomic actions, processes, top-level goals and sequences.

- *An atomic action* (a-action) is the primitive operation an h-agent can do. E.g. `aStep(hAgentID,placeID)`.
- *A sequence* is a simple sequence of a-actions or processes, e.g. $<a_1, a_2, p_1, a_3, p_2>$ (p denotes a process, a denotes an action).
- *A process* is a set of *process-steps* (p-steps), which are quadruples $<p, r, c, a>$, where p is a priority local to the process (such that p-steps of one process are total-ordered by their priorities), r is a releaser, c is an optional context and a is an action. Releasers and contexts are boolean conditions, an action can be an a-action, a subprocess or a sequence. In the following we will write directly "priorities of releasers" instead of "priorities of p-steps".
- *A top-level goal* is quadruple $<pr, r, c, f>$, where pr is a process associated with achieving the goal, r and c are releaser and context respectively, and f is a function of time that serves as a floating priority of the goal.

Subprocesses are nested under each top-level goal in a tree-like hierarchy. Leaves represent a-actions. The behaviour of one agent is represented by a set of such behavioural structures and this set is called a *betree* (it comes from a "behavioural tree"). The betree is always provided in advance by a behavioural programmer/designer and it is not further modified when an h-agent is running (in S-HRP).

All p-steps and top-level goals of the betree are either active or preactive or inactive or sleeping, all a-actions and processes are either executed or not-executed.

At every instant, at least on of top-level goals' releasers must hold. The highest priority goal with the holding releaser is *active*, the other goals with holding releasers are *preactive*; the rest is *inactive*.

A sequence, which is a child of an active node (i.e. a p-step), is *executed*; other sequences are *not-executed*. A process or an a-action, which is a child of an active node (i.e. a p-step or a top-level goal), is *executed*. Exactly one process or exactly one a-action from an executed sequence is also *executed* (the one just being performed). The other processes and the a-actions are *not-executed*.

Each executed process is associated with several p-steps. At every instant, at least one of their releasers must fire. The p-step (under an executed process) with holding releaser and with the highest priority is called *active*. Its siblings with holding releasers (and lower priority) are called *preactive*. Its siblings without a holding releaser (both with higher and lower priority) are called *inactive*. All p-steps under a not-executed process are *sleeping*. In the following, we will often write directly "active/ inactive/... releasers" instead of "active/inactive/... p-steps/goals".

An example of a betree is depicted in Figure 2. Figure 3 shows the action selection algorithm. This algorithm is performed in every time-step by a control unit of an h-agent. When the simulation starts, all nodes of the given betree are marked as not-executed, or sleeping, except for top-level goals, which are inactive.

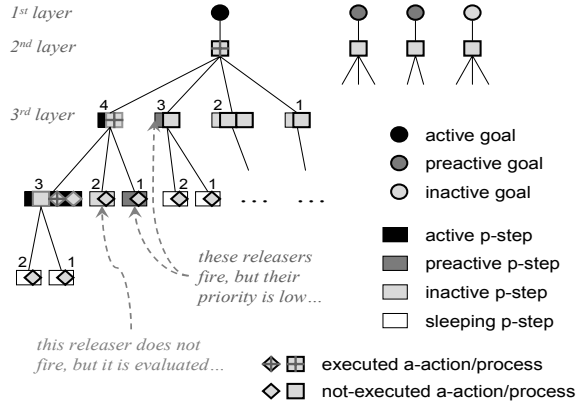

Fig. 2: An example of a betree. The priorities are written above the p-steps.

When the algorithm starts, it evaluates all releasers of top-level goals and identifies an executed process (SELECT-GOAL). Then it recursively finds in a breath-first manner all active, preactive and inactive releasers in the upper layer of the subtree of the executed element (A-S). The active releaser in a leaf triggers an a-action and finishes the evaluation (26). A non-leaf active releaser expands the evaluation

```
SELECT-GOAL( betree ):
(1)  eval ← all releasers of top-level goals of betree  % they "never sleep"
(2)  evaluated ← evaluate eval
(3)  set active/preactive/inactive top-level goals based on evaluated
(4)  set executed/not-executed processes of top-level goals
(5)  p-steps ← all p-steps of executed process
(6)  A-S( third layer of betree, p-steps, betree ) % 3rd layer – see Fig. 2
A-S( layer, p-steps, betree ):
(7)  releasers ← all releasers of p-steps
(8)  eval ← evaluate releasers
(9)  set active/preactive/inactive p-steps from p-steps based on eval
(10) set all other nodes % i.e. p-steps % in the layer as sleeping
(11) act ← the action of the active p-step of p-steps
(12) if act differs from previously executed action then
(13)    set previously executed flag as not-executed
(14) if act is "a-action" or "process" then
(15)    EXEC( act, layer, betree )
(16) else if act is "sequence" then
(17)    if act is not executed then set act as executed
(18)    if act already contains executed element then
(19)       set this element as not-executed        % element = process or
(20)    if this element is not the last element of act then      % a-action
(21)       EXEC( the next element of act, layer, betree )
(22)    otherwise    % restart the sequence:
(23)       EXEC( the first element of act, layer, betree )
EXEC( act, layer, betree ):
(24) set act as executed
(25) if act is "a-action" then
(26)    execute( Act )
(27) otherwise % act is now a "process"
(28)    A-S( next layer of betree, all p-steps of act, betree )
```

Fig. 3: The S-HRP evaluation algorithm.

to the lower layer of the betree (28). Because there is exactly one active releaser in each layer, there is also exactly one active p-step in each layer. Subsequently, exactly one a-action can be performed in the given time step. This a-action can be performed several times, until another leaf releaser becomes active. An agent's attention is switched to another branch of the betree, i.e. to another subtask, when previously preactive or inactive releaser is activated. That happens either when an executed process is finished (i.e. its releaser holds not more), or when external/internal circumstances are changed. As exactly one branch of the betree can be executed, there is no parallelism in S-HRP.

The asymptotic complexity of the S-HRP-evaluation is $O(p.l.a)$, where l is the depth of the betree, p an average number of p-steps of one executed process and a a constant that limits the time for evaluation of one releaser; provided that releasers are re-evaluated in every time step. (Complexity can be reduced significantly by utilizing a variant of RETE algorithm [Forgy, 1982].)

The purpose of a context is to interrupt currently executed action, even if the releaser holds. As contexts are typically conjunctions of releasers with higher priorities, they are omitted from the description of the algorithm for the sake of simplicity (it is possible to express for example timeouts or numbers of retries by means of them). Scheduling of top-level goals is enabled by their floating priorities.

The four elements of S-HRP are similar to the elements of Byson's POSH action selection plans [2001] (primitive actions, action patterns, competences and drive collections). The whole S-HRP-betree resembles to AND-OR tree with only AND branches, which are used for example in total-order simple task network planning [Ghallab *et al.*, 2004].

4.2 Behaviour of the artificial gardener

The behavioural structure of the a-gardener is depicted in Figure 5. For simplicity, library functions like searching for an object or drinking are not detailed. Figure 4 shows the priorities of top-level goals and the whole course of the behaviour. The behaviour of the a-gardener is programmed in the language E of the project Ent [Brom *et al.*, 2005].

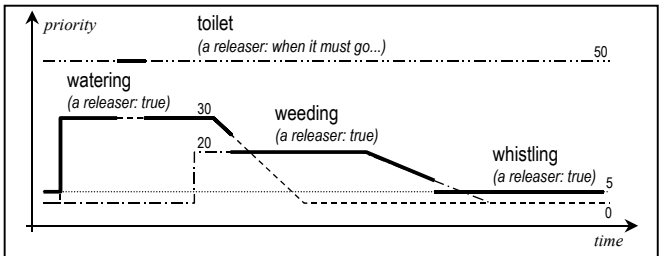

Figure 4: Priorities of top-level goals of the a-gardener. The resulting course of behaviour (without interaction with the user-agent) is indicated by the bold line. The time of switching from *watering* to *toilet* is based on the incremental rate of the internal drive underlying "when it must go" releaser.

5 Results: observed limitations

In this section, we present observed limitations on believability of behaviour of the artificial gardener. The results clearly reveal four main flaws of the a-gardener and thus of the S-HRP method. First, the S-HRP betree does not allow for intentions, the best one could do in S-HRP is to associate intentions with top-level goals. Second, concurrent processes are not allowed—just one a-action can be performed in each time step. Third, some situations require planning, at least to some extent, but S-HRP avoids classical planning. And finally, strong need for transition behaviours, i.e. small processes that applies during task switching, has been recognised. Unfortunately, it is not straightforward to express them in S-HRP. We anticipate that some of these limitations can be avoided by utilizing a different reactive hierarchical method, but others are more fundamental.

5.1 Intentions

Case 1. Choosing an alternative:

N-gardener: Before he starts watering, he decides whether to hose or to water by a can.

A-gardener: When the watering goal becomes active (i.e. *what* to do), the a-gardener is not able to choose between alternatives (i.e. *how* to do), because exactly one process is associated with the top-level goal.

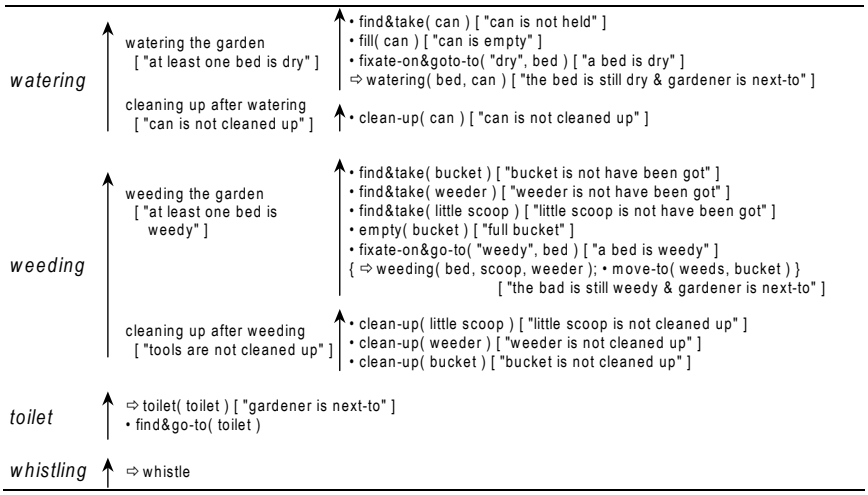

Figure 5: The schema of behaviour of the a-gardener. Parenthesis denotes parameters, brackets denote releasers, angle brackets stand for a sequence. Arrows denote priorities, the p-step with the highest priority is at the top. A-actions are marked with (⇨), subgoals are marked with (•). Contexts are omitted for simplicity.

It is noteworthy, that contrary to other hierarchical reactive methods (e.g. [Bryson, 2001]), the p-steps are written in the reversed-order (the first thing done is on the top) and the releasers are expressed in a negative form. The reason for this is to make the programming simpler.

Case 2. Adding a new intention:

N-gardener: When a neighbour asks for a can, the n-gardener promises her to bring her the can after he finishes watering and then, he becomes *intended* to this task.
A-gardener: It completely lacks capability to add itself a new intention, that means a new top-level goal.

Comment: Changes of intentions and feasibility of choosing alternative ways how to accomplish a goal are basic requirements on action selection model for h-agents. These features are not built-ins of S-HRP, but are well-defined in others hierarchical reactive architectures, namely in the Belief Desire Intention (BDI) [Wooldridge, 2002]. S-HRP could be pushed towards BDI by specifying *simple goal driven hierarchical reactive planning* (S-GHRP):

1. A new structure of *goal* is introduced. It is a quintuple $<\{pr\}, me, r, c, f>$, where $\{pr\}$ is a set of processes that can accomplish the goal, *me* is a procedure for means-ends reasoning among the processes, and *r*, *c* and *f* are a releaser, a context and a priority, a function of time, respectively. A *top-level goal* is just an ordinary goal.
2. An *extended sequence* is a simple sequence of a-actions, processes or goals.
3. A *p-step* is augmented as follows: it is a quadruple $<p, r, c, {}^{g}a>$, where ${}^{g}a$ is an extended action, i.e. a subprocess or an a-action or a goal or an extended sequence.
4. A S-GHRP betree is partially *modifiable on-line*. When an h-agent is running, a new goal can be added to the betree, both top-level one and a subgoal.
5. In S-GHRP, we say that a goal is active, preactive, inactive or sleeping *iff* the p-step encapsulating the goal (or a sequence with the goal) is active, preactive, inactive or sleeping, respectively. Top-level goals are never sleeping, but all top-level goals that are not intended (i.e. not a part of the betree) can be considered as such. From the other hand all not-sleeping goals can be regarded as intentions.
6. The EXEC procedure of the algorithm in Fig. 2 is called also when the *act* variable contains a goal (line (14)) and it performs *me* reasoning in this case (line (25)).

Notice, that the set of all active and preactive elements of the S-GHRP betree is similar to so-called *intention structure* of JAM architecture [Marcus, 1999], which is an implementation of BDI. We suggest that implementations of BDI can be exploited in solving the issue on intentions.

5.2 Concurrent processes and interleaving

Case 3. Pure parallelism:

N-gardener: When he is watering, he whistles from time to time for joy.
A-gardener: It lacks capability to do two tasks simultaneously.

Comment: A simulated body of a believable h-agent should be viewed as a group of semi-independent resources that can perform a-actions concurrently, and S-GHRP should be applied in parallel version, where more active nodes can co-exist in one betree-layer. This idea is hardly surprising and, in fact, a lot of reactive hierarchies address this issue (e.g. [Blumberg, 1996; Bryson, 2001]). It is also noteworthy, that modelling of preferences' combination may be required. That means choosing a compromise action when two (or more) concurrent tasks compete for the same resource (for a discussion on this topic see [Tyrrell, 1993, p. 185-187]).

Case 4. Preparation:

N-gardener: When he is beginning watering, he goes to a chamber and finds a can. Because he knows that he will weed afterwards, he finds also a bucket, and puts a weeder and a little scoop into it. Then he takes the bucket and the can and goes to the garden.
A-gardener: When it is beginning watering, it takes a can from a chamber and goes to the garden. When it finishes the watering, it returns to the chamber for a bucket, a weeder and a little scoop.

Comment: Two goals conflict, active watering and inactive weeding, but even though inactive, weeding has to manifest itself in order to save the second trip to the chamber. Goals have to be *interleaved*. The question is how to give "losers" chances to influence overall behaviour out of their time-slots. Classical solution is to use a planning technique, in this case partial-order simple task network planning (STN) would be appropriate. Unfortunately, it is not straightforward (and perhaps even not biologically plausible) to combine reactive methods with this kind of planning. Therefore, we suggest another solution based on semi-autonomous fuzzy triggers:

1. The priority function of a goal, *f*, is replaced with the set of fuzzy-triggers $\{t^{+}\}$.
2. A *fuzzy trigger* is like a releaser in that it continuously monitors an environment, an agent's body or its mind in order to recognise some relevant situations. Unlike a releaser, the trigger is able to invoke resource negotiation procedure *ne*(*pow*) between an active goal (or goals) and an inactive/preactive applicant. *pow* is the actual power of the trigger (a value <0, 1>).
3. Based on the result of *ne* the applicant can either subsume the active goal, or the active goal can let the "loser" manifest itself shortly, or the *me* procedure of the active goal can switch to another process[1].

The challenging issue is to identify situations that should invoke negotiation. One of such situations is: *an h-agent is attracted by an object that is supposed to be use later*. This notion of semi-autonomous triggers puts S-GHRP a bit towards Minsky's Society of Mind [1985]. A fundamental question on efficiency of this method rises. What is more

[1] We are working on a prototype implementation of negotiation procedure using Soar [Newell, 1990].

computationally effective? STN planning or a bundle of reactive triggers and negotiation procedures?

Case 5. Task inhibition:

(In this case, we assume that a goal of lending a can to the neighbour is intended also by the a-gardener.)
N-gardener: When he finishes watering, he goes to the neighbour and lends her the can.
A-gardener: When it finishes watering, it puts the can to the chamber, then it picks it immediately up and goes to lend it to the neighbour.

Comment: There is another kind of situation that should be recognised by a trigger: *the h-agent attempts to do a task whose effect will be later cancelled*. This situation is unlike Case 4 because the result of negotiation is temporal *inhibition* of a subgoal. To describe this, a new type of trigger is useful: *an inhibition trigger*. It temporarily inhibits a releaser of a p-step that would invoke a conflicting goal (i.e. putting the can to the chamber).

Extended definition of a goal is: the goal is a quintuple $<\{pr\}, me, r, c, \{t^+, t^-\}>$. t^+ and t^- are tuples $<t, ne>$, where t is the trigger and ne is the negotiation procedure. The difference between t^+ and t^- is that t^+ starts negotiation in order to activate the goal, while t^- starts negotiation in order to inhibit a releaser of a p-step with an undesirable goal.

Inhibition is a fundamental feature of architectures like of Maes [1991] (an inhibition link) or Minsky [1985] (a suppressor and a censor agent). Its need is also mention for example in [Charles *et al.*, 2002]. Nevertheless, it is typically not a built-in of reactive hierarchies. We will call this kind of extended S-HRP *negotiatory goal driven hierarchical reactive planning*, the N-GHRP.

5.3 Stop and think

Case 6. Seeing a distance:

(This case extends the scenario from Section 2 as it describes one situation more precisely.)
N-gardener: When he is preparing tools for weeding, he first looks around and then chooses almost optimal way how to pick up a bucket, a weeder and a little scoop.
A-gardener: When it is preparing tools for weeding, it follows the priorities of the p-steps and no matter how the objects are far it always picks up the bucket, then the weeder and finally the little scoop (see Fig. 6).

Comment: This is an observation of a task with complex appetitive behaviour. A question is what we humans do in these situations. We think that two cases should be distinguished. The first is when a human perceive all objects of the interest at once and there are less then three or four of these objects. The second case encompasses more complicated situations with hidden objects or more objects on the scene. We think that in the former case the human *directly* perceive the order of how to pick the objects up[2], while in the latter case the human *consciously* stops and thinks a bit about what to do next.

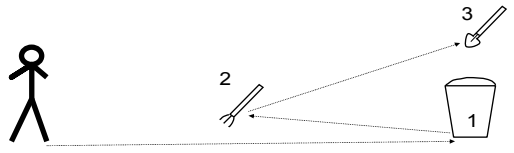

Fig. 6: The order in which the n-gardener picks the objects up is fixed.

For an a-gardener: The latter case calls for a conventional planner that should be invoked by the first subprocess of the process with complex appetitive behaviour. The purpose is to re-arrange the order of appetitive subtasks (i.e. to change priorities of p-steps—it corresponds to stop and think activity). The former, direct perceiving, can be simulated by using releasers and triggers for each possible ordering.

Case 7. A sharp timeout:

(We assume a can is not in the chamber and must be looked for.)
N-gardener: He remembers that the can is often in the chamber. When he does not find it there, he starts searching it within the whole house. As time is passing, he becomes more and more angry. After a while, he wants to give it up, but then he suddenly spot the can in the garage. He picks it up and returns to the garden.
A-gardener: It remembers that the can is often in the chamber. When it does not find it there, it starts searching it within the whole house. After 14.55 minutes of searching, it catches sight of the can in the dining room. It makes two steps towards the can, but just before it reaches the can, the timeout (15 min.) expires. The task of watering is failed.

Comment: A sharp timeout can be expressed directly in S-HRP by a context of a p-step. We suggest that instead of the sharp timeout, a soft one should be used. It incorporates not only time, but also appropriate environmental/body/mental states. N-GHRP triggers serve this purpose better than contexts, because they can facilitate negotiation.

5.4 Transition behaviour

Case 8. No transition:

N-gardener: When nature calls if he is watering, he puts down the can and goes to the toilet.
A-gardener: If it needs to go to the toilet during watering, it goes with the can in its hands and puts it down when in the toilet.

Comment: We have stumbled on so-called *cleaning behaviour*, which is in the case of humans performed in some of its form as a consequent of a consumatory behaviour almost ever. An example of pure cleaning is cleaning up the can. This behaviour can be simply described in S-HRP by adding one p-step after the consumatory act. However, special kind of cleaning behaviours, *transitions*, that mean short behaviours that should automatically apply

[2] Here, we refer to the concept of an *affordance* and *direct perceiving* of James J. Gibson [1979]. However, the discussion on this topic is out of the scope of this paper.

when two "major" ones are being switched, complicate the situation. An example of transition is putting away the can. The need for transitions is noted for example in [Blumberg, 1996; Mateas, 2002]. The problem is that they can not be simply expressed in S-HRP.

We suggest that both for pure cleaning and for transitions negotiation from N-GHRP could be utilised as follows:

1. As the result of negotiation, the incoming goal should give a small amount of time to perform cleaning or transition process to an outgoing goal. The amount of time should be proportional to the ratio of the necessities of behaviours.
2. If an incoming goal is very urgent, transition may be also performed as a part of it. As an example, assume a case of an attack—the gardener might throw the currently holding object at the aggressor, instead of putting it down.

5.5 Other lessons learned

Here, we briefly mention the remnant of observations.

Postponement. When the task *A* is aimed to suspend the task *B* (e.g. eating watering) *postponement* could be negotiated if *B* is almost finished. This is similar to Case 7—when a-gardener is finishing watering, so-called *small variant* of eating could be performed (e.g. eating a tomato from the garden instead of lunching), or watering should not be interrupted at all.

Quantities. Serious problem appears when an h-agent is confronted with huge amount of objects it is potentially interested in. Consider an h-agent aimed to eat a carrot, which needs to be pulled out from a garden bed first—there are hundreds of such carrots in the garden and typical cognitive h-agent's perception system that is designed to perceive all of them, will push this pile into the h-agents' memory. We suggest that in such a situation, the h-agent should instead of perceiving some concrete objects rather see a container, e.g. the garden bed.

Blocking behaviour. The common problem is a situation in which a process *A* shortly corrupts its own context. A correction process *B* (typically a sibling from the betree) could fix the situation, but then *A* corrupts the context again—that only leads to an infinite loop. Consider the a-gardener who must first hold the can to be able to fill it, but in order to turn water on, it must temporarily put it down. It is the same problem as with Herbert, the robot retrieving cans, that blocked by its arm its camera focused on the can, when it had begun to pick the can up [Connell, 1990]. An h-agent must use a memory to remember that it has to avoid execution of the correction process.

6 Discussion: S-HRP vs. related AI models

We have shown several situations in which S-HRP fails as the action selection model for believable h-agents. We conclude that this does not mean the methodology has to be discarded, but rather reviewed instead. Considering the fact that h-agents carry out large number of complex goals in unpredictable and dynamic environments, the hierarchies together with reactive approach must be utilised anyway. There are two main reasons for this. First, hierarchies reduce design complexity. Second, because believable h-agents are aimed for real simulations with several h-agents running on a single PC, their action selection model must be computationally effective (h-agents belong to the field of applied AI, rather then computational psychology or ethology). Therefore, we think that models based on spreading activation in a flat network (e.g. [Maes, 1991]) or in a hierarchical network (e.g. [Tyrrell, 1993; Negatu, 2003]) would not fit, because they generally suffer from combinatorial complexity.

What does it mean to review S-HRP? S-HRP is partially based on Bryson's [2001] basic reactive plans. We suggest that it can be simply extended into S-GHRP by adding the concept of goals. S-GHRP is in fact BDI architecture (e.g. [Wooldridge, 2002]), nevertheless, we suggest that GHRP can be pushed further towards another approaches. Namely, we suggest adding "stop-and-think" planner (but not conventional planning in general) and semi-autonomous triggers that are able to cause resource negotiation, and inhibit an undesirable goal. The second concept is inspired by Minsky's Society of Mind [1986] and Maes [1991]. We have called such architecture "negotiatory goal driven hierarchical reactive planning", the N-GHRP, and we have recommended applying its parallel version. As N-GHRP combines reactive approach with conventional planning, it might be viewed as a *hybrid* architecture representative.

What is the contribution? We see the main contribution of the architecture in that the triggers are able to break the monolithical reasoning procedure into relatively independent modules. Notice, that decomposition of the reasoning procedure is neither decomposition of the body (e.g. [Blumberg, 1996]) nor decomposition of overall behaviour into independent behavioural-modules (e.g. [Bryson, 2003]). It is yet another kind of decomposition.

The decomposition of reasoning blurs the borders between behaviours and makes the alternation among prescripted plans more "smooth" and thus natural and believable (contrary to "rigid 'artificial' switching" in S-HRP and S-GHRP/BDI). For example, sharp timeouts can be avoided, undesirable tasks can be inhibited, preactive behaviour can be demonstrated shortly without its timeslots, transitions can be expressed and postponement can be negotiated.

There is yet another branch of methods that is suitable for believable h-agents. It is any-time planning. For example, Nareyek uses any-time planning based on *structural constrained satisfaction* [2005] and Charles *et al.* exploit a variant of *hierarchical task network planning* and *heuristic search planning* [2002]. We think that anytime planning do not allow for as easy design as reactive hierarchies do. However, the correct comparison between N-GHRP and any-time planning methods is a question for future research.

S-HRP and similar methodologies belongs to the branch of so-called *forward-chaining* methods. To complete the

picture, we must mention Soar architecture [Newell, 1990], which is one of the most known cognitive forward-chaining architectures. Soar is also exploited in h-agents simulations. However, it is rather a powerful programming vehicle, not a design methodology. For example, S-GHRP as well as simple task network planning can be programmed in it.

7 Conclusion

In this paper, we have argued that hierarchical reactive planning is not able to cope with human-like behaviour. We have shown several limitations of this branch of methods through behavioural analysis of an artificial gardener, whose behaviour have been designed according to simple hierarchical reactive planning, the S-HRP.

The main limitations of S-HRP include: 1) impossibility of adding new goals/intentions during execution, 2) the shortage of parallel execution and task interleaving, 3) the impossibility of inhibition an undesirable subtask, 4) fixed-ordered steps in appetitive behaviour, 5) rigid "unnatural" switching between behaviours, which disable for example expressing of transition behaviours and postponement. The first one is the limitation only of the S-HRP method. The second one is the limitation of all the methods that do not allow parallel execution and/or preactive behaviours. The third is the limitation of methods that cannot express inhibition. The last two are limitations of the whole branch of reactive hierarchical family.

We have suggested a solution to overcome these by extending S-HRP to N-GHRP, negotiatory goal driven hierarchical reactive planning. It is a hybrid architecture representative. The precise comparison between N-GHRP and other hybrid approaches, namely any-time planning, is a question for future research.

Acknowledgement

The application Ents was developed as a student project at Faculty of Mathematics—Physics, Charles University, Prague. Thanks to Vladislav Kuboň for supervising the project and to Ondřej Bojar, Milan Hladík, Vojtěch Toman, David Voňka and Mikuláš Vejlupek for their contribution. Thanks also to Rudolf Kryl, Iveta Mrázová, Kamamúra Ryšlink, Terezka Slunečnice, Matěj Hoffmann, Tomáš Bureš and three anonymous referees for their suggestions and comments.

References

[Blumberg, 1996] Bruce M. Blumberg. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD thesis, MIT Media Laboratory, Learning and Common Sense Selection, 1996

[Brom et al., 2005] Ondřej Bojar, Cyril Brom, Milan Hladík, Vojtěch Toman. The Project ENTs: Towards Modeling Human-like Artificial Agents. In *SOFSEM 2005 Communications*, pages 111–122, Liptovský Ján, Slovak Republic, January 2005.

[Bryson, 2000] Joanna Bryson. Hierarchy and Sequence vs. Full Parallelism in Action Selection. In: *The Sixth International Conference on the Simulation of Adaptive Behaviour (SAB00)*, pages 147–156. MIT Press, Ma, Cambridge, USA, 2000.

[Bryson, 2001] Joanna Bryson. *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. PhD thesis, Massachusetts Institute of Technology, 2001.

[Bryson, 2003] Joanna Bryson. The Behaviour-Oriented Design of Modular Agent Intelligence. In: *Proceedings of Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pages 61-79, Springer LNCS 2592, Germany, 2003.

[Charles et al., 2002] Fred Charles, Miguel Lozano, Steven J. Mead, Alicia F. Bisquerra, Marc Cavazza. Planning Formalisms and Authoring in Interactive Storytelling. In: *First International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Germany, 2002.

[Connell, 1990] Jonathan H. Connell. *Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot*. Academic Press, Cambridge, Ma, 1990.

[Forgy, 1982] Charles L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In: *Artificial Intelligence*, 19, pages 17-37, 1982.

[Ghallab et al., 2004] Malik Ghallab, Dana Nau, Paolo Traverso. Hierarchical Task Network Planning. In: *Automated Planning: Theory and Practice*, chapter 11, Morgan Kaufmann Publishers, San Francisco, Ca, USA, 2004

[Gibson, 1979] James J. Gibson. *The Ecological Approach to Visual Perception*. Boston: Houghton Miffin, 1979.

[Maes, 1991] Pattie Maes. The agent network architecture (ANA). In: *SIGART Bulletin*, 2 (4), pages 115–120, 1991.

[Marcus, 1999] Marcus J. Huber. JAM: A BDI-theoretic mobile agent architecture. In: *Proc. of 3rd International Conference on Autonomous Agents*, pages 236-243, Seatle, USA, 1999.

[Mateas, 2002] Michael Mateas. *Interactive Drama, Art and Artificial Intelligence.* Ph.D. Dissertation. Department of Computer Science, Carnegie Mellon University, 2002.

[Minsky, 1985] Marvin Minsky. *The Society of Mind*. Simon and Schuster Inc., 1985

[Nareyek, 2005] Alexander Nareyek. Project Excalibur. An on-going project on agents for computer games. Homepage: http://www.ai-center.com/projects/ excalibur/

[Negatu, 2003] Aregahegn S. Negatu, Stan Franklin. An Action Selection Mechanism for "Conscious" Software Agents. In: *Cognitive Science Quarterly*, 2, pages 363-386, 2002.

[Newel, 1990] Alan Newell. *Unified Theories of Cognition.* Harward University Press, Cambridge, Masachusetts, 1990.

[Prendinger et al., 2004] Predigner, H., Ishizuka, M. Introducing the cast for social computing: Life-like characters. In: *Life-like Characters. Tools, Affective Functions and Applications*, Cognitive Technologies Series, Springer, Berlin, p. 3-16, 2004.

[Tyrrell, 1993] Toby Tyrrell. *Computational Mechanisms for Action Selection*. Ph.D. Dissertation. Centre for Cognitive Science, University of Edinburgh, 1993.

[Wooldridge, 2002] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.