

# A SNePS Agent for Unexploded Ordnance Disposal Progress Report

Stuart C. Shapiro and Haythem O. Ismail

Department of Computer Science and Engineering  
State University of New York at Buffalo  
226 Bell Hall

Buffalo, NY 14260-2000

Phone: (716) 645-3180

E-mail: {shapiro | hismail}@cse.buffalo.edu

August 27, 1999

---

## 1 Introduction

In this document, we present recent developments in the design of the knowledge level of the UXO disposal agent. Those developments enhance the interrupt handling model that was presented in our 5/31/1999 report. Although there are still problems to be solved, we have managed to solve two of the problems pointed out in the last report in addition to discovering and solving other problems. Section 2 provides a brief review of the main ideas underlying the interrupt handling system. Section 3 describes changes to the SNePS act scheduler. Section 4 discusses enhancements of the priority system and the prioritized acting procedure. Section 5 introduces a problem that we discovered with mental acts and outlines our current solution. Finally, Section 6 discusses problems and future work. Appendix A contains a demonstration of the system.

## 2 Interrupt Handling

In this section, we briefly review the main contents of our last (5/31/1999) report. In that report we explained in detail how the SNePS system was used to allow the UXO disposal agent to handle interrupts. By outlining the main ideas presented therein, we lay the ground to a detailed discussion of recent enhancements of the system.

1. A theory of interrupt handling subsumes one for error recovery. An error is just a special kind of an interrupt, one resulting from the unexpected situation of the failure of one of the agent's acts.

### 3 The New Scheduler

Given the above discussion, there seems to be three unsatisfying issues regarding the proposed interrupt handling mechanism.

1. There are two different mechanisms for handling primitive and composite acts.
2. The agent always stops everything that it is doing when an interrupt occurs. Those on-going acts with high priorities should be allowed to continue to completion.<sup>2</sup>
3. It is not clear when exactly to use *interrupt*. The act is used as a reaction to some situation that requires the agent to stop everything that it is doing and take some appropriate action. But then all situations that require the agent to react should make use of *interrupt*. In general, whenever the agent is doing something and it decides that it is appropriate to perform some other act (whether because it was told to do so or because it needs to react to some situation) it should *reason* about whether to interrupt what it is currently doing. Since this is a general requirement, it would be better to build interrupt handling into the acting system rather than require explicit use of *interrupt* in the knowledge base.

Solving the above problems was achieved by revising both PFI and the procedural semantics of *p-do-all*. In this section, we discuss the revised PFI procedure and defer the discussion of *p-do-all* to the next section. As pointed out in Section 2, PFI results in all scheduled acts being performed in order of their priorities. More specifically, the acts scheduled on the act stack,  $\Sigma$ , are replaced by a *p-do-all* of those acts. That is:

$$\Sigma \leftarrow \{\text{p-do-all}(\Sigma)\}$$

To build interrupt handling into the act scheduler, we revised PFI so that the contents of  $\Sigma$  are replaced by a *p-do-all* of all the scheduled acts in addition to those that are currently being performed.

$$\Sigma \leftarrow \{\text{p-do-all}(\Sigma \cup \Pi)\}, \text{ where } \Pi \text{ is the set of on-going acts (processes).}$$

This way, whenever the agent is about to act (while in PFI mode), it has to reason about (i) what to do first and (ii) whether to interrupt its on-going activities.

---

<sup>2</sup>This is the fourth problem pointed out in our last report (page 12).

2.  $a_2 = \text{cascade}(a_j, \dots, a_{j+m})$  and  $a_1 >_p a_j$ , or

3.  $\text{Holds}(\text{p-higher}(a_1, a_2), \text{*NOW})$  is deducible.

Accordingly, the relation  $>_p$  holds between two cascades,  $c_1$  and  $c_2$ , if it holds between their first elements. The base case of the recursion is the explicit assertion of priorities (in terms of *p-higher*) among acts. However, the above does not say that  $c_1$  should be completed before starting to perform  $c_2$ , it only defines the  $>_p$  relation. What should be done when this relation holds between two acts is a different issue that we now turn to.

**Definition 4.2** Let  $A$  be a set of acts. Define the set  $A_\top$  as follows. An act  $a \in A_\top$  iff:

1.  $a \in A$ ,  $a$  is a cascade, and for every  $\tilde{a} \in A$ ,  $\text{Holds}(\text{p-higher}(a, \tilde{a}), \text{*NOW})$  is deducible;
2.  $a \in A$ ,  $a$  is not a cascade, and there is no  $\tilde{a} \in A$  such that  $\tilde{a} >_p a$ ; or
3.  $c = \text{cascade}(a, \dots, a_n) \in A$  and there is no  $\tilde{a} \in A$  such that  $\tilde{a} >_p c$ .

Intuitively,  $A_\top$  is the set of acts in  $A$ , or embedded within cascades in  $A$ , that should be performed first, i.e, those with *top* priorities. A complementary set contains whatever remains.

**Definition 4.3** Let  $A$  be a set of acts. Define  $A_\perp = (A - A_\top) \cup \{\text{cascade}(a_{2_i}, \dots, a_{n_i}) \mid \text{cascade}(a_{1_i}, a_{2_i}, \dots, a_{n_i}) \in A \text{ and } a_{1_i} \in A_\top\}$

Informally, a p-do-all with a set of acts  $A$  reduces to a cascade of  $\text{do-all}(A_\top)$  followed by  $\text{p-do-all}(A_\perp)$ .<sup>4</sup> Going back to the problem discussed at the beginning of this section, and assuming that  $\text{Holds}(\text{p-higher}(\text{Recharge-battery}, \text{Search}), \text{*NOW})$  is deducible, we have the following situation:

- $A = \{\text{cascade}(\text{Recharge-battery}), \text{cascade}(\text{Search}, \text{Pick-up}, \text{Drop})\}$ .
- $A_\top = \{\text{Recharge-battery}\}$ .
- $A_\perp = \{\text{cascade}(\text{Search}, \text{Pick-up}, \text{Drop})\}$ .<sup>5</sup>

Thus the agent would exhibit the correct behavior; recharging the battery and then resuming what it was doing before.

<sup>4</sup>But see below for some subtle differences.

<sup>5</sup>Note that empty cascades are not allowed by Definition 4.3.

as  $c_1$ . Assume that the agent starts performing  $c_1$  while it is in the field (Z1). To drop the UXO it needs to achieve the precondition for dropping; being in the drop-off zone (Z4). This is done by performing the cascade `cascade(Goto(Z4), Drop)`.<sup>6</sup> Let us call this cascade  $c_2$ . Before starting to perform  $c_2$ , the agent senses that the battery is low. This results in scheduling a p-do-all of  $c_2$  and `Recharge-battery`. Assuming that  $c_2$  has higher priority over recharging the battery, the agent sets out to perform `cascade( $c_2$ , Recharge-battery)`.<sup>7</sup> Let us refer to that as  $c_3$ . To perform  $c_3$ , the agent needs to start performing  $c_2$  and to form the belief that when the goal of  $c_2$  has been achieved it should perform `Recharge-battery`. What is the goal of  $c_2$ ? According to Definition 3.1 in our last report (page 4), it is achieving the state `complete( $c_2$ )`. This state gets asserted by appending the mental act `believe(complete( $c_2$ ))` to the end of  $c_2$ . That is,  $c_2$  expands to `cascade(Goto(Z4), Drop, believe(complete( $c_2$ )))`.

Now the agent reaches Z4. At this point, it would be useful to inspect the agent's intentions. In particular, it has the following beliefs.

- `when-do(Empty-handed, cascade(Goto(Safe-zone), Talk-to(Bill)))`.
- `when-do(Empty-handed, believe(complete( $c_2$ )))`.

The first form represents the agent's intention to continue  $c_1$  after dropping the UXO. The second represents its intention to continue  $c_2$ . Once the agent finishes dropping the UXO and becomes empty-handed, the two acts `cascade(Goto(Safe-zone), Talk-to(Bill))` and `believe(complete( $c_2$ ))` get scheduled. These two acts are then performed in order of their priorities. However, this seems very unsatisfying. The only reason the mental act was introduced is to enable scheduling `Recharge-battery` once the agent has finished dropping the UXO. It is the priority of `Recharge-battery` that should be compared to that of the rest of  $c_1$ , not the priority of the mental act.

One way out of this problem is to introduce a rule to the effect that mental acts have higher priority over all other acts. This might indeed solve the problem. However, it seems awkward for the agent to *reason* whether it should *believe* before, say, *search*. If the agent has the intention to believe some proposition, it should do so once the appropriate conditions are met (in the above example, having dropped the UXO). This requires us to build the higher priority of mental acts into the acting system rather than explicitly represent it in the knowledge base. This was done by introducing a special queue,  $Q$ , for mental acts. The acting system only starts processing the contents of  $\Sigma$ , the non-mental acts stack, when it has processed all the contents of  $Q$ . This way, mental acts will be performed before

<sup>6</sup>See Section 5 of our 5/31/1999 report.

<sup>7</sup>A do-all or a p-do-all with one act reduces to that act.

## A Ascii World Demonstration

The following is a demonstration of the high level behaviors Gather-Objects and Blow-up Objects in the ASCII world. Sentences surrounded by asterisks are the output of the world simulation. Sentences preceded by semi-colons are explanatory comments. Otherwise, it is the agent explaining what it is doing. In both cases, the first sentence, following the prompt (:), is the natural language input to the system. For this demonstration, we manually set the field to contain only four objects.

;;;-----The High-Level Behavior: Gather-Objects-----

```
: Clear the field.
I am going to Z1.
**The robot is going to WORLD:Z1.**
**The robot is in WORLD:Z1 at the point: (150.00, 5.00).**
I went to Z1.
I am in Z1.
I am searching.
**The robot is searching for a UXO . . .**
**Object found at: (204.06, 22.04).**
**The robot is going near the object (WORLD::ORANGE 204.0638 22.035358) . . .**
**The robot is looking at the object (WORLD::ORANGE 204.0638 22.035358).**
**The robot is near the object (WORLD::ORANGE 204.0638 22.035358).**
**The robot is going to examine object.**
**OBJECT FOUND IS A UXO.**
I searched.
I am near a UXO.
I am picking up the UXO.
**The robot is picking up the UXO.**
I picked up the UXO.
I am holding the UXO.
I am turning to Z4.
**The robot is looking towards WORLD:Z4.**
I turned to Z4.
I am looking at Z4.
I am going to Z4.
**The robot is going to WORLD:Z4.**
**The robot is in WORLD:Z4 at the point: (35.81, 3.69).**
I went to Z4.
```

**\*\*<<<THE BATTERY IS LOW>>>.\*\***

;;;The battery goes low while the agent is about to drop the UXO.  
;;;This is similar to the situation discussed in Section 4.1.  
;;;Also similar to that discussed in Section 5.

I went to Z4.  
I am in Z4  
and a battery is low.  
I am dropping the UXO.  
**\*\*The robot is dropping the UXO.\*\***  
I dropped the UXO.  
I am empty handed.

;;;Dropping the UXO has higher priority over recharging the battery if in Z4.

I am turning to Z3.  
**\*\*The robot is looking towards WORLD:Z3.\*\***  
I turned to Z3.  
I am looking at Z3.  
I am going to Z3.  
**\*\*The robot is going to WORLD:Z3.\*\***  
**\*\*The robot is in WORLD:Z3 at the point: (19.69, -5.00).\*\***  
I went to Z3.  
I am in Z3.  
I am recharging the battery.  
**\*\*The robot is recharging the battery.\*\***  
I recharged the battery.  
the battery is full.  
I am talking to you.

;;;Correctly prioritized the two cascades of recharging the battery and of the main  
;;; gathering loop (see Sections 4.1 and 5).  
;;;The on-going talking was stopped while recharging and now resumes.

I am turning to Z1.  
**\*\*The robot is looking towards WORLD:Z1.\*\***  
I turned to Z1.  
I am looking at Z1.  
I am going to Z1.

```

**The robot is going near the object (WORLD::WHITE 187.23366 327.44772) . . .**
**The robot is looking at the object (WORLD::WHITE 187.23366 327.44772).**
**The robot is near the object (WORLD::WHITE 187.23366 327.44772).**
**The robot is going to examine object.**
**OBJECT FOUND IS NOT A UXO.**
**The robot is searching for a UXO . . .**
**Object found at: (36.72, 347.09).**
**The robot is going near the object (WORLD::ORANGE 36.72171 347.08948) . . .**
**The robot is looking at the object (WORLD::ORANGE 36.72171 347.08948).**
**The robot is near the object (WORLD::ORANGE 36.72171 347.08948).**
**The robot is going to examine object.**
**OBJECT FOUND IS A UXO.**
  I searched.
  I am near a UXO.
**The robot has a charge.**
  I am placing a charge.
**The robot is placing a charge on the UXO (WORLD::ORANGE
                                     36.72171
                                     347.08948).**

**The charge is on the UXO.**
  I am empty handed.
  I placed the charge.
  the charge is on the UXO.
  I am hiding.
**The robot is looking for a safe place. . .**
**The robot is going to the point: (141.67, 346.41). . .**
**The robot is at a safe place.**
**<<<THE BATTERY IS LOW>>>.**

```

```

;;;While waiting for the explosion, the battery goes low.
;;;Note that continuing to wait should have higher priority.

```

```

  I hid.
  I am safe
and the battery is low.
  I am turning to Z3.
**The robot is looking towards WORLD:Z3.**
  I turned to Z3.
  I am looking at Z3.
**!!BANG!!**
  I am going to Z3.
**The robot is going to WORLD:Z3.**

```

\*\*The robot is looking at the object (WORLD::WHITE 172.33784 368.4796).\*\*  
\*\*The robot is near the object (WORLD::WHITE 172.33784 368.4796).\*\*  
\*\*The robot is going to examine object.\*\*  
\*\*OBJECT FOUND IS NOT A UXO.\*\*  
\*\*The robot is searching for a UXO . . .\*\*  
Z1 is cleared.  
I am turning to Z2.  
\*\*The robot is looking towards WORLD:Z2.\*\*  
\*\*<<<THE BATTERY IS LOW>>>.\*\*  
I turned to Z2.  
I am looking at Z2.  
I am going to Z2.  
\*\*The robot is going to WORLD:Z2.\*\*  
\*\*The robot is in WORLD:Z2 at the point: (170.60, -5.00).\*\*  
I went to Z2.  
I am in Z2  
and the battery is low.  
I am turning to Z3.  
\*\*The robot is looking towards WORLD:Z3.\*\*  
I turned to Z3.  
I am looking at Z3.  
I am going to Z3.  
\*\*The robot is going to WORLD:Z3.\*\*  
\*\*The robot is in WORLD:Z3 at the point: (32.00, -5.00).\*\*  
I went to Z3.  
I am in Z3.  
I am recharging the battery.  
\*\*The robot is recharging the battery.\*\*  
I recharged the battery.  
the battery is full.  
I am talking to you.