GENERATING WARNING INSTRUCTIONS BY PLANNING ACCIDENTS AND INJURIES

Daniel Ansari and Graeme Hirst[1]
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 3G4, Canada

Abstract

We present a system for the generation of natural language instructions, as are found in instruction manuals for household appliances, that is able to automatically generate safety warnings to the user at appropriate points. Situations in which accidents and injuries to the user can occur are considered at every step in the planning of the *normal* operation of the device, and these "injury sub-plans" are then used to instruct the user to *avoid* these situations.

1 Introduction

We present a system for the generation of natural language instructions, as are found in instruction manuals for household appliances, that is able to automatically generate safety warnings to the user at appropriate points. Situations in which accidents and injuries to the user can occur are considered at every step in the planning of the *normal* operation of the device, and these "injury sub-plans" are then used to instruct the user to *avoid* these situations. Thus, unlike other instruction generation systems, our system tells the user what *not* to do as well as what to do. We will show how knowledge about a device that is assumed to already exist as part of the engineering effort, together with adequate, domain-independent knowledge about the environment, can be used for this. We also put forth the notion that actions are performed on the materials that the device operates upon, that the states of these materials may change as a result of these actions, and that the goal of the system should be defined in terms of the final states of the materials.

We take the stand that a complete natural language instruction generation system for a device should have, at the top level, knowledge of the device (as suggested by Delin et al. (1993)). This is one facet of instruction generation that many NLG systems have largely ignored by instead incorporating the *knowledge of the task* at their top level, i.e., the basic content of the instructions is assumed to already exist and does not need to be planned for. In our approach, all the knowledge necessary for the planning stage of a system is contained (possibly in a more abstract form) in the knowledge of the artifact together with the world knowledge. The kinds of knowledge that should be sufficient for this planning are device knowledge (topological, kinematic, electrical, thermodynamic, and electronic) and world knowledge.

The IDAS project of Reiter et al. (1992; 1995) served as a key motivation for this work. One of the primary goals of the IDAS project was to automatically generate technical documentation

[1]Address correspondence to the second author. E-mail: gh@cs.toronto.edu

from a domain knowledge base containing design information (such as that produced by an advanced computer-aided design tool) using NLG techniques. IDAS turned out to be successful in demonstrating the usefulness, from a cost and benefits perspective, of applying NLG technology to partially automate the generation of documentation. If work in qualitative process theory, using functional specifications such as those in e.g., (Iwasaki et al., 1993), can yield the device and world knowledge that are required for text planning, then the need for cost effectiveness would be met.

2 A situation calculus approach to the generation of instructions

2.1 Overview

In this section we shall present some of the planning knowledge for a toaster domain, in the form of axioms in the *situation calculus*[2] (see (Reiter, 1991)). This planning knowledge formally characterizes the behaviour of the artifact, and it is used to produce a basic plan of actions that both the device and user take to accomplish a given goal. The axioms together with the goal are the input to our system. We will explain how the instructions are generated from the basic plan. This plan is then used to derive further plans for states to be avoided, and warning instructions about these situations.

We shall use the term *device–environment system* to refer to the device, the user, and any objects or materials used by the device.

We can conceptually divide the actions that are performed in the device–environment system into *user actions* and *non-user actions*, the latter of which are actions that are carried out either by the device on its components and the materials it uses, or by some other agent. Because the majority of non-user actions are actions performed by the device, we shall only consider device actions henceforth. Natural language instructions are directed to the user of a device, and usually they mainly describe the actions that are executed by the user.

A device action may be carried out by a component of the device on another component; for example, the heating element of a toaster may carry out a *heating* action (i.e., a continuous, physical process) on the bread slot, which in turn may heat the inserted bread slice.

Instead of using a qualitative or quantitative simulation system, such as the Device Modelling Environment (Iwasaki and Low, 1991), we have used device actions to discretely model the continuous processes, for simplicity.

Table 1 shows the components of our toaster and the materials used for its operation. Table 2 shows the user actions, device actions, and fluents.

[2] In the situation calculus, the initial state is denoted by the constant S_0, and the result of performing an action a in situation s is represented by the term $do(a,s)$. Certain properties of the world may change depending upon the situation. These are called *fluents*, and they are denoted by predicate symbols which take a situation term as the last argument. *Positive (negative) effect axioms* describe the conditions under which performing a in situation s causes a fluent to become true (false) in $do(a,s)$. Action precondition axioms describe the conditions under which a can be performed in s. We use these axiomatic forms in order to avoid the frame problem. Following Pinto (1994), we shall abbreviate terms of the form $do(a_n,(do(\ldots,do(a_1,s)\ldots))$ as $do([a_1,\ldots,a_n],s)$.

Components
ON lever
bread slot

Materials
bread slice

Table 1: Components and materials of the toaster system

User actions
insert
remove
press
touch
get_burned

Device actions
raise_temp
pop_up

Fluents
pressed
contains
removed
temperature
touching
burned
toasted
exposed

Table 2: User actions, device actions, and fluents used in the toaster example

2.2 Some axioms for the toaster system

The following are some of the more important axioms for our toaster example (see Ansari (1995) for the complete set). Some of them are essentially domain-independent, whereas the others relate specifically to the appliance. Where free variables appear in formulas, they are assumed to be universally quantified from the outside.

2.2.1 Action precondition axioms

$$Poss(insert(x, y), s) \equiv three\_d\_location(y) \wedge fits(x, y) \wedge exposed(y, s) \tag{1}$$

$$Poss(touch(x), s) \equiv physical\_object(x) \wedge exposed(x, s) \tag{2}$$

$$Poss(get\_burned, s) \equiv \exists x, t.(touching(x, s) \wedge temperature(x, t, s) \wedge t \geq 70) \tag{3}$$

$$Poss(raise\_temp(x), s) \equiv (x = bread\_slot \vee contains(bread\_slot, x, s)) \wedge \exists t.(temperature(x, t, s) \wedge t < 200) \wedge pressed(on\_lever, s) \tag{4}$$

$$Poss(pop\_up, s) \equiv \exists t.(temperature(bread\_slot, t, s) \wedge t \geq 200) \tag{5}$$

These axioms state that:

- an action by the agent of inserting x into y is possible in state s if y is a *three_d_location*, i.e., a spatial volume, x fits into y, and y is exposed;

- an agent can touch an object if it is exposed;
- the agent can get burned by touching something with a temperature of at least 70°C; and
- the device can cause the bread slot to pop up its contents if the temperature of the bread slot reaches 200°C.

2.2.2 Positive effect axioms

$$Poss(a, s) \wedge a = insert(x, y) \rightarrow contains(y, x, do(a, s)) \tag{6}$$

$$Poss(a, s) \wedge a = get\_burned \rightarrow burned(do(a, s)) \tag{7}$$

$$Poss(a, s) \wedge a = pop\_up \wedge contains(bread\_slot, x, s) \rightarrow exposed(x, do(a, s)) \tag{8}$$

These axioms state that:

- inserting x into y in state s results in y containing x in state $do(a, s)$;
- if it is possible for the agent to get burned (by the *get_burned* action), then the agent might be burned in the new state; and
- if the device causes x to pop up in state s, then x becomes exposed in the next state.

2.2.3 Negative effect axioms

$$Poss(a, s) \wedge a = press(on\_lever) \wedge contains(bread\_slot, x) \rightarrow \neg exposed(x, do(a, s)) \tag{9}$$

This axiom states that an action of the user pressing the ON lever causes anything in the bread slot to become unexposed; this happens because the object in the bread slot gets "pushed down".

3 Generating instructions with warnings

3.1 Deriving instruction plans from the axioms

We wish to derive a sequence of actions (by the user and the device) that, when performed, cause a slice of bread to become toasted. Ideally, this sequence would begin with the act of the user inserting a slice of bread into the toaster and end with the act of the user removing the toasted bread from the toaster. The goal will be described in terms of the final state of the material (bread, in this case). Thus, the plan will describe a sequence of actions which cause the transformation of the material from its initial to its desired state.

$$temperature(bread\_slot, 20, S_0)$$
$$temperature(bread\_slice, 20, S_0)$$
$$exposed(bread\_slot, 20, S_0)$$
$$exposed(bread\_slice, 20, S_0)$$

Figure 1: Fluents that hold in the initial state, S_0

We could, as a reasonable approximation, model the state changes of the bread in terms of the temperature of the bread. Using *temperature*(x, t, s) as a fluent describing that object x has a temperature of t°C in state s, we could define toast as a slice of bread that has reached a temperature of 200°C:

$$toasted(bread\_slice, do(a, s)) \leftarrow$$
$$temperature(bread\_slice, t, s) \wedge t \geq 200 \vee toasted(bread\_slice, s) \quad (10)$$

Note that using this definition, *toasted(bread_slice)* holds for all states after *do(a,s)*.

Figure 1 shows the fluents that hold in the initial state.

We can define the goal G to be the following:

$$G = toasted(bread\_slice) \wedge removed(bread\_slice, bread\_slot) \quad (11)$$

A plan derived by our system to cause G to become true is this:

$$do([insert(bread\_slice, bread\_slot), press(on\_lever), raise\_temp(bread\_slice),$$
$$raise\_temp(bread\_slice), raise\_temp(bread\_slice), raise\_temp(bread\_slice),$$
$$pop\_up, remove(bread\_slice, bread\_slot)], S_0) \quad (12)$$

The *raise_temp* action is carried out four times, since each time it raises the temperature of something by 50°C.

Note that we do not model the perception actions of the user watching for the bread slice to pop up. In our simple domain, we have avoided the need for these by assuming that the user knows when a salient observable change occurs in the system. In this case, the salient change is the popping up of the bread slice.

3.2 Deriving plans for warning instructions

Now that we have seen how plans for basic instructions can be obtained, we can describe how warning instructions can be derived.

In order to generate warning instructions, the system must be able to derive plans, using the available actions and fluents, in which the user can become harmed. There are many ways in which this can happen: by burning, electric shock, laceration, crushing, etc. We shall concentrate on examining the conditions under which burns to the user can occur.

We can derive a plan in which the user gets burned by setting the goal G to be this:

$$G = burned \quad (13)$$

References

Ansari, Daniel. 1995. Deriving procedural and warning instructions from device and environment models. Technical report CSRI-329, Department of Computer Science, University of Toronto. *ftp://ftp.cs.toronto.edu/csri-technical-reports/329/*

Delin, Judy , D. Scott, and T. Hartley. 1993. Knowledge, intention, rhetoric: Levels of variation in multilingual instructions. In *Association for Computational Linguistics Workshop on Intentionality and Structure in Discourse Relations*, pages 7–10.

Di Eugenio, Barbara. 1992. Understanding natural language instructions: The case of purpose clauses. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 120–127.

Iwasaki, Yumi and Che Ming Low. Model generation and simulation of device behavior with continuous and discrete change. Technical report KSL-91-69, Knowledge Systems Laboratory, Stanford University.

Iwasaki, Yumi, Richard Fikes, Marcos Vescovi, and B. Chandrasekaran. How things are *intended* to work: Capturing functional knowledge in device design. *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambéry, France, 1516–1522.

Kosseim, Leila and Guy Lapalme. 1994. Content and rhetorical status selection in instructional texts. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 53–60.

Moore, Johanna D. and Cécile L. Paris. 1989. Planning text for advisory dialogues. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 203–211.

Penman Natural Language Group. 1989. *The Penman Documentation.* University of Southern California, Information Sciences Institute,

Pinto, Javier A. 1994. *Temporal Reasoning in the Situation Calculus.* Ph.D. thesis, University of Toronto. Also available as Technical Report KRR-TR-94-1.

Reiter, Ehud, Chris Mellish, and John Levine. 1992. Automatic generation of on-line documentation in the IDAS project. In *Third Conference on Applied Natural Language Processing*, Trento, pages 64–71.

Reiter, Ehud, Chris Mellish, and John Levine. 1995. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9: 259–287.

Reiter, Raymond. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* Academic Press, San Diego, CA, pages 359–380.

Vander Linden, Keith. 1993. *Speaking of Actions: Choosing Rhetorical Status and Grammatical Form in Instructional Text Generation.* Ph.D. thesis, University of Colorado. Also available as Technical Report CU-CS-654-93.

Wahlster, Wolfgang, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, and Thomas Rist. 1993. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63: 387–427.