A language for functional interpretation of model based simulation

Jonathan Bell, Neal Snooke and Chris Price
Department of Computer Science, University of Wales Aberystwyth
Penglais, Aberystwyth, Ceredigion, SY23 3DB, U. K.
jpb, nns, cjp@aber.ac.uk

Abstract

Functional modeling is in use for the interpretation of the results of model based simulation of engineered systems for design analysis, enabling the automatic generation of a textual design analysis report that expresses the results of the simulation in terms of the system's purpose. We present a novel functional description language that increases the expressiveness of this approach, allowing a system function to be decomposed in terms of subsidiary functions as well as required effects, increasing the range both of systems and design analysis tasks for which the approach can be used.

1 Introduction

The automation of the design analysis of engineered systems requires both simulation of the behavior of the system and interpretation of the results, to enable the automatic generation of a draft report outlining the results of the design analysis. This report is cast in terms of the system's purpose and as system function can be regarded as relating its behavior to purpose, this interpretation of simulation for automatic report generation is a useful rôle of functional description.

Functional modeling has been used for deriving the behavior of a system from knowledge of its structure and component functions [Sticklen and Chandrasekaran, 1989], supporting design by functional refinement [Iwasaki *et al.*, 1993], and interpretation of simulation for automatic report generation [Price, 1998] where system functions are associated with significant behaviors (outputs or goal states). Here we present a novel language for the description of system function that increases the expressive power of that approach by allowing partial fulfilment of a system's purpose to be described. Novel features of the language also increase both the range of systems that can be described and the range of tasks for which the language is useful.

2 Describing and decomposing function

The definition of function that underlies the language presented here might be given informally as "how a device achieves its purpose" or more formally as

> Function: An object O has a function F if it achieves an intended goal by virtue of some external trigger T resulting in the achievement of an external effect E.

This is distinct from notions of behavior and purpose, which is a problem with some definitions of function used in model based reasoning. It follows that a representation of function has three elements; a description of purpose and the function's trigger and effect. The trigger and effect are Boolean expressions and act as the recognizer of achievement of the function. A function can fail in two ways. Either the trigger can fail to result in the expected effect or the effect can occur without the trigger, as summarized in Table 1. The four

Trigger	**Effect**	**Function**
false	false	**I**noperative
true	false	**Fa**iled
false	true	**U**nexpected
true	true	**A**chieved

Table 1: Achievement of function using trigger and effect.

states of a function are defined in terms of the truth of the trigger and effect, so if a function is triggered but the effect is not present it is said to be failed, for example. Agreement in value between the trigger and effect is consistent with correct behavior of the system. We cannot, however, pair off these cases as the consequences of the two inconsistent behaviors will differ and the resulting report must reflect this.

The description of purpose is separate from the functional description itself. This is consistent with the idea that function is concerned with how a purpose is fulfilled rather than the purpose itself and also encourages model reuse as similar purposes might be fulfilled by systems with different functional requirements, such as cars' and motorbikes' lighting systems.

A function might depend on more than one trigger and effect so logical operators can be used to describe the required combinations. Alternatively, a function can be composed of subfunctions, each with its own trigger, effect and purpose. This distinction allows cases where the achievement of one subfunction mitigates failure of the top level function, such as where a warning system has visual and audible signals and the presence of either one means at least some warning is given.

Decomposition in terms of functions means that the top level function's states must be derived from the four possible states of each of the subfunctions. This is shown in Table 2. The rule is that triggering of the top level function is derived from the triggering of the children and achievement of its effect from those of the children. So, for example, in the sixth line of the table, where Child 1 is achieved (both trigger and effect are true) and Child 2 failed (trigger true but effect false) a top level function that depends on Child 1 AND Child 2 has failed as it is triggered (Child 1 trigger AND Child 2 trigger is true) but the effect is not achieved as Child 1 effect AND Child 2 effect resolves to false. Where the state of the function is in brackets, the report will ignore the top level function and include the associated failure of the child function. In addition to the logical operators, a functional description might use the sequential operators described in [Bell and Snooke, 2004] to allow systems whose function depends on intermittent or sequential behaviour to be described.

Child 1	**Child 2**	**AND**	**OR**	**XOR**
I	**I**	**I**	**I**	**I**
I	**A**	**I**	**A**	**A**
I	**Fa**	(**I**)	**Fa**	**Fa**
I	**U**	(**I**)	**U**	**U**
A	**A**	**A**	**A**	**I**
A	**Fa**	**Fa**	(**A**)	(**U**)
A	**U**	**U**	(**A**)	**Fa**
Fa	**Fa**	**Fa**	**Fa**	(**I**)
Fa	**U**	(**I**)	(**A**)	(**A**)
U	**U**	**U**	**U**	(**I**)

Table 2: States of functions and sub-functions.

As there are three elements in a complete functional representation, there are three possible pairs that can be used as incomplete subfunctions to simplify the functional decomposition. As pairing trigger and purpose (with no effect) is unnecessary, this leaves two classes of incomplete function that can usefully be incorporated into the language. These are

Purposive incomplete function (PIF) maps effect to purpose. This can be used when several subfunctions share a trigger, as in the warning example mentioned above.

Operational incomplete function (OIF) maps trigger to effect with no distinct purpose. It ensures a trigger is associated with a specific effect where alternative combinations of trigger and effect can achieve some purpose.

These are only used as subsidiary (child) functions.

3 Example of functional description

The following is (part of) the functional description of a car's seat belt reminder system that warns the driver if either front seat is occupied with its seat belt unbuckled.

```
FUNCTION belt_warning
  ACHIEVES unbuckled_warning
  BY
    vehicle moving
    AND (driver_unbuckled OR
      (passenger_present AND
       passenger_unbuckled))
    TRIGGERS
      PIF warning_lamp AND PIF chimer

PURPOSE unbuckled_warning
  DESCRIPTION "Warn seat belt is unbuckled"
  FAILURE_CONSEQUENCE
    "No warning given of dangerous state"
```

The PIFs and their associated purpose descriptions are, like the description of purpose shown, separate entities. The decomposition in this example allows the language to distinguish the case where one of the PIFs fails from the more serious case where both fail so no warning is given. In addition to this increased expressiveness the explicit inclusion of the trigger differs from the language in [Price, 1998] and allows unambiguous description of functions that are triggered by the state of some other system function, increasing the range of systems that can be modeled. The trigger and the use of labels for trigger and effect allow a more precise specification of required system behavior than the earlier language used for functional labeling. They also enable the functional description to be built independently of a target system, so the language for can be used for functional refinement of design in addition to interpretation of model based simulation.

4 Conclusion

The functional description language introduced here increases both the expressiveness and range of the language used for functional labeling in [Price, 1998]. It shares the advantages of simplicity, reusability and capability claimed for that language and in addition allows the functional model to be constructed independently of the system, so supporting the use of the language in functional refinement of a design as well as for interpretation of simulation.

References

[Bell and Snooke, 2004] Jonathan Bell and Neal A. Snooke. Describing system functions that depend on intermittent and sequential behavior. In *Proceedings 18th International Workshop on Qualitative Reasoning, QR2004*, 2004.

[Iwasaki *et al.*, 1993] Yumi Iwasaki, R. Fikes, M. Vescovi, and B. Chandrasekaran. How things are intended to work: Capturing functional knowledge in device design. In *Proceedings of 13th International Joint Conference on Artificial Intelligence*, pages 1516–1522, 1993.

[Price, 1998] Christopher J. Price. Function-directed electrical design analysis. *Artificial Intelligence in Engineering*, 12(4):445–456, 1998.

[Sticklen and Chandrasekaran, 1989] Jon Sticklen and B. Chandrasekaran. Integrating classification-based compiled level reasoning with function-based deep level reasoning. In Werner Horn, editor, *Causal AI Models, Steps Toward Applications*, pages 191–220. Hemisphere Publishing Corporation, 1989.