A Structural Framework for the Formal Representation of Cooperation

Anthony Finkelstein
Imperial College of Science, Technology & Medicine(University of London)
180 Queens Gate, London SW7.
acwf@doc.ic.ac.uk

0 *Introduction*

This paper presents a structural framework for the formal representation of cooperation and illustrates this framework with an outline example. The paper builds upon a basic argument:

> that cooperation is a characteristic feature of software development;
>
> that much of the observed complexity of software development results from the operation of the underlying mechanisms of cooperation.

It follows from this that a prerequisite for modelling software development is a detailed understanding of cooperation. This understanding must of necessity be backed up by appropriate means for formally representing that cooperation.

Before discussing how such a formal representation might be organised it is necessary to explain precisely what we mean by cooperation. Cooperation is the activity through which agents (autonomous and loosely coupled) contribute to the achievement of a joint understanding on which action can be based. Cooperation can thus be seen as a form of reasoning.

1 *Structural Framework*

Cooperation rests on interaction between agents. It follows from this that we need to have some idea of the structure of that interaction. This problem is closely analogous to that encountered in user interface specification where it is necessary to separate concerns in the analysis of computer-human interaction (Moran 1981). The computer-human interaction is divided into levels. At the top level are the tasks set by the user (eg to draw a specific diagram) and at the lowest level the management of presentation (eg spatial layout and device handling). Between these levels are:

> the semantic level, the conceptual objects and the methods for achieving tasks in terms of these objects (eg box, Place, Link, Unlink);
>
> the syntactic level, the syntactic rules and structure of conceptual objects and methods (eg Link connects two specified boxes);
>
> the lexical level, the rules governing the actions associated with syntactic elements (eg connection causes a line to be drawn between the centres of the two boxes).

Each level has a representation scheme associated with it. At the semantic level this might be logical relations, at the syntactic level it might be transition networks or grammars, at the lexical level it might be a graphical description language.

This structure is directly mirrored in cooperative interaction. In other words we can divide such interactions into similar levels. At the top level we can place the software development tasks (eg program validation) and at the bottom the physical details of communication between agents (eg network protocols). Between these levels are the levels concerned with the mechanisms of cooperation. In the example we develop below we focus on these intermediate levels.

2 *Example*

An obvious set of software development tasks which we might want to support are encountered in configuration management - a classical setting in which cooperation comes into play. Typical of such tasks is the construction and maintenance of a baseline of software development objects. This involves the "publication" of objects to the baseline, the removal of objects from the baseline and so on. What might the levels underlying these tasks look like?

Cooperation, viewed as a form of reasoning, is particularly amenable to logical representation and analysis. One area of cooperation in which such representation and analysis has been developed is that of logics of dialogue (or conversation). These provide a powerful repertoire of constructs and rules which can be used to describe how cooperation works.

Our illustration is based on a dialogue logic, DC, which is fully presented in Finkelstein & Fuks (1989). A fragment of the scheme is given to assist in understanding the framework. To reduce the scope of our example we will concern ourselves with the minimal case of two cooperating (configuring) agents.

2.1 Semantic level

The conceptual objects and methods, defined as as follows:

> statements - constructed in a propositional language which includes negation, conditional and conjunction of statements.
>
> acts- basic operations, represented by a statement and a modifier, examples of acts are assertions represented **asserts**(statement), withdrawals represented **withdraws**(statement), questions represented **questions**(statement) and challenges represented **why**(statement);
>
> commitments - public engagements to statements (in effect, holding yourself out as liable for the consequences of a statement), represented **committed**(Stage,Agent);
>
> events -a cooperative task consists of a sequence of events each of which is represented by a triple of the form <Stage, Agent, Act>, Stage marks the progress of the interaction, stage, stage+1 and so on, Agent indicates the current active agent (speaker).

The task level, in this example, can be mapped onto the semantic level in a relatively straightforward way:

> the statements describe the software development object configuration;
>
> the view each agent has of the baseline is represented in terms of commitments;

tasks are built up from sequences of events, thus publication is simply assertion, the removal of an inconsistent version might involve a sequence such as challenge-assertion-resolution demand-withdrawal-withdrawal;

and so on.

2.2 Syntactic level

The syntactic organisation and structure of the conceptual methods and objects, given in rule form, for example:

> After the questioning of a statement (**questions**(Statement)), the next event must be either the assertion (confirmation) of that statement, it's withdrawal or it's denial (**asserts**(Statement), **withdraws**(Statement) or **denies**(Statement)).
>
> No legal interaction of length stage+1 contains an event
> <stage-1,hearer,**questions**(Statement)> unless it also contains an event
> <stage,speaker,**asserts**(Statement)> ∨ <stage,speaker,**withdraws**(Statement)>
> ∨ <stage,speaker,**denies**(Statement)>.

2.3 Lexical level

The rules governing the actions associated with syntactic elements. The actions at this level are updates and deletions from the commitment record of an agent. For example:

> After a withdrawal the statement is removed from the speaker's commitment record, the hearer's record remains unchanged.
>
> After <stage,speaker,**withdraws**(Statement)>
> **committed**(stage+1,speaker)=**committed**(stage,speaker) - {Statement}
> **committed**(stage+1,hearer)=**committed**(stage,hearer)

3 *Conclusion*

This paper has outlined a structural framework for the formal representation of cooperation and illustrated this framework with a small example. The framework can be used to analyse, select and develop formal descriptions of cooperation. It appears that relatively simple semantic, syntactic and lexical levels can, in conjunction, support relatively complex cooperative tasks. We are currently looking at a variety of different sets of rules and constructs, largely drawn from logics of dialogue, to substantiate this. Our particular interest is in the cooperative underpinning of "multi-party" specification - more specifically the cooperative mechanisms that underlie requirements elicitation. We hope to build, around the structural framework, tools and environments to support these tasks.

References

Moran, T. (1981); The Command Language Grammar: a representationof the user interface of interactive computer systems; International Journal of Man-Machine Studies; 15 pp 3-50.

Finkelstein, A. & Fuks, H. (1989); Multi-party Specification; Proc. 5th International Workshop on Software Specification & Design; IEEE CS Press.