Modelling the IceCube approach to reconciliation of divergent replicas

Abstract

IceCube is a recent technology developed by Microsoft. Its purpose is to deal with the issue of reconciling divergent replicas of some shared system state, which can occur in e.g. multi-user concurrent systems. In other words, when users of a computer system each make modifications to a local copy of some global state, it is necessary at some point to merge these modifications in order to obtain some new, updated global state. Conflicts between local system updates can make the reconciliation complex or even impossible, and IceCube attempts to address this issue on an application-independent basis. Its algorithm relies heavily on the notion of constraints between modifications done by users. As these constraints have to be specified by the application programmer, the use of IceCube can be a very difficult task. This paper presents the argument that a model of the IceCube reconciliation process based on EM principles has applications both as an educational and an experimental tool.

1 Introduction

The process of reconciliation[1] is one of interplay between man (or at least between some intelligent agent, perhaps AI) and machine, as it inherently involves non-automated decision-making. To design a process to keep human interaction to a minimum requires tremendous insight into the relationships between the properties of an application's state. The IceCube project has attempted to abstract some of these relationships. The result is a generalized reconciliation process which can be used across a wide spectrum of applications, together with a complex interpretation and optimisation task for each application developer. The focus of this paper is to describe how the Eden model (see **?**) combined with EM principles can assist humans in this context, both in terms of using IceCube effectively and conducting experiments which could result in improved reconciliation algorithms in the future. We will commence by giving a short introduction to IceCube, followed by an outline of the model which was developed and a discussion of its possible uses in terms of EM principles. Finally we will give an account of the theme of this paper in a wider context and suggest future work which could enhance the model.

[1]Reconciliation is here taken to mean merging copies of some state that have developed independently

1.1 Introduction to IceCube

IceCube is a general-purpose framework for reconciliation of divergent replicas. It is highly generic in the way that it can be used with any type of system that has an explicit state. Replicas of this state evolve individually, each through a sequence of state changes, called actions, from a pre-defined set of possible actions. Such a sequence is called a log, and IceCube can subsequently be used to develop a new global system state which is consistent with all the replicas. The technical details of IceCube are not discussed further in this paper, but are described in **?**. Example applications that could potentially use IceCube include file versioning systems (e.g. source code or document development) and server-based business software (e.g. meeting planner systems).

2 Model development

2.1 Modelling approach

The modelling approach taken here is based upon agent-oriented programming aspects, the most important aspect being that of inherent focus on the processes of action and perception. Although the real-world state is abstracted in our model (we will come back to this later), we are concerned with representing the observations that can be made of this state, as well as relationships between these observations. In this context, the part of the reconciliation engine of

IceCube that we are modelling is seen as an agent. Other agents will be identified and described later, however, the artefact's own agent is special in the way that it embodies the current state of the reconciliation process, and actuates autonomous responses to other agents' actions consistent with the reconciliation algorithm.

2.2 Applicability of model

It must be noted at this point that the model developed is not suitable for use by applications that wish to utilise the IceCube reconciliation approach. The most obvious reason for this is that it lacks the ability to run simulations on a given state, which is an integral part of the reconciliation progress (see **?**). This use was, however, not an intended result of the modelling study. There are different people, each with a unique role, that provide input to the reconciliation engine in its real-world use (see figure 1). In our modelling study, these roles map onto agents each with a different set of actions that have the effect of modifying the observables within the model. Each agent also has the opportunity to perceive modifications to the state resulting from its own and other agents' actions, through the system's graphical interface. The next section discusses how the model is designed to aid the external agents' construal of its state.

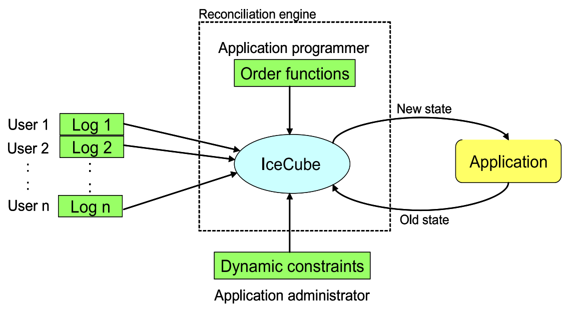

Figure 1: The figure shows the different agents involved in the reconciliation process. Integral to the process are the order functions which describe relationships between admissible log actions. The logs to be reconciled, the application state on which they are based and dynamic constraints between actions are supplied at runtime.

3 Agency, observables and dependencies

In the context of this modelling study, one of the main focus points is how observables and dependencies within the model relate to agents. This section will discuss this in relation to the human agents.

3.1 The application programmer

As mentioned earlier, it is up to the application programmer to supply the so-called ”order functions”, which really are constraints on the ordering of pairs of actions in the output of the algorithm, i.e. the final log. There are many factors which influence the programmer's choice of order functions. One is strict rules, such as the fact that two meetings cannot be planned in the same slot (cf. example in **?**). Another is qualitative reasoning, like ”it makes sense to carry out meeting cancellations before new requests” (this could for example imply that the order function for ”request” followed by ”cancellation” ought to be set to unsafe, as in the example in **?**). To experiment with different order functions in order to find an optimal one, the programmer needs to be able to interact with the reconciliation process. The IceCube model offers such interaction in the form of buttons for input, a graphical display and text output.

As this model is intended to be an experimental tool rather than an ”application” in the more mainstream sense, we will not try to enumerate the ways in which the programmer could approach this model. However, one possibility is to try out different order functions for a specific pair of input logs. He could then use the automation functionality to obtain the successive output actions that the algorithm suggests. By assuming that dynamic conflicts occur often, the programmer could perceive the alternatives that the algorithm suggests and thus assess the success of his choice of order function in restricting the search space for an admissible output. Another possibly fruitful approach might be to inspect the directed graph that shows the dependency (red edges) and safe (blue edges) relations: Many red edges could be a positive sign because it restricts the possible orderings of actions and thus constricts the search space, but loops in the red graph makes the reconciliation problem impossible to solve. Conversely, a limited number of blue edges could reduce the search space for the next output action, while both no blue edges leaving the current node and edges going to all remaining nodes will leave the search space unrestrained. Through experience with the model, the author believes that many more such relationships between dependencies and observables could be found.

Of course, the programmer also needs to make sure that the order function he uses always finds a solution to the reconciliation problem if such a solution does in fact exist. In this context, it might be useful for him to use a pen and a notepad to sketch solutions to specific problems and their successive states after each action. The Empirical Modelling concept of ”thinking with computers” is evident here: As the programmer enters order functions through the model's interface, he can observe the results of the dependencies within the artifact and connect these perceptions with the perhaps more concrete state representations on his notepad. In this way, the combination of the model and notepad complement each other, and should help improve the programmer's construal of the concrete state of the reconciliation process.

3.2 Other potential agents

As stated earlier, the model abstracts the concrete state of the process. In other words, it only takes into account the relationships between successive actions on a state; it does not consider these actions in relation to the state itself. During reconciliation, dynamic conflicts which the stateless algorithm cannot possibly foresee take place, and this is the reason for the dialogue following each step (see **?** for hints about how to use the model). An application administrator wants answers to such questions as ”is a certain set of logs reconcilable based on the current state and order functions” and ”what sort of application-specific information does the reconciliation engine require”. Answering the latter of these questions could help in creating AI agents to supply dynamic information to IceCube when implemented. Experiments with the output dialogue could help give the application administrator a better understanding of answers to both these questions.

Other human agents who could potentially benefit from experimenting with the model include system programmers with an interest in reconciliation algorithms. For instance, such a person could observe the dependencies between different inputs and the graph relations, and use his understanding to invent new relations to improve the algorithm. Such understanding could also be used to improve the artefact itself, and this will be further discussed in section 4.

4 Evaluation

Rather than trying to embed the whole reconciliation process through a closed-world assumption, the definitive script only captures the dependencies between the underlying mathematical structures and observables that attempt to assist the human experimenter's construal. The model takes into account only a part of IceCube's functionality; it does not, for example, consider the process of simulation, which links the logs with the state upon which they are based. This is one aspect which could foreseeably be included in the model in the future. As simulation is less directly related to human input to the process, it is perhaps of less importance in this context. Another future extension could be expansion in terms of input size, i.e. the number of order functions, logs, and permissible actions. Finally, observables that currently have to be modified indirectly through the interpreter could be brought into the graphic display of the model in accordance with the ”thinking with computers” principle.

5 Conclusion

The author believes that the model discussed in the paper is a good starting point for anyone who wants to experiment with the functionality of IceCube. This paper has discussed the construction and important properties of the model, and identified some of the agents that might have an interest in its use. It has argued that through EM principles, people in different roles can get insight into IceCube's reconciliation algorithm and how to use it effectively.