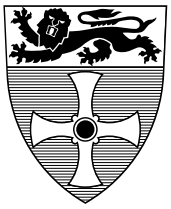
UNIVERSITY OF
NEWCASTLE



University of Newcastle upon Tyne

COMPUTING SCIENCE

Dimensions of Dynamic Coalitions

J. W. Bryans, J. S. Fitzgerald, C. B. Jones and I. Mozolevsky

TECHNICAL REPORT SERIES

No. CS-TR-963 May, 2006

TECHNICAL REPORT SERIES

No. CS-TR-963 May, 2006

Dimensions of Dynamic Coalitions

Jeremy W. Bryans, John S. Fitzgerald, Cliff B. Jones and Igor Mozolevsky

Abstract

Developments in network technology are enabling organisations to form temporary alliances to achieve specific goals. Such alliances are often referred to as "dynamic coalitions", emphasising the fluid character of their memberships. Dynamic coalitions vary widely in architecture, scale and complexity, ranging from ad hoc groupings of organisations created in order to perform a very brief transaction to long-running collaborations between allies. In many cases, there is significant sharing of information between the participants.

The term "dynamic coalitions" is often used without definition, giving rise to potential confusion and unfulfilled expectations. This paper attempts to map out a space of dynamic coalitions, using a systematic approach supported by a formal (mathematically-based) modelling language. Seven "dimensions" are identified and explored, with an emphasis on the flow of information through coalitions. A case study examines software being developed to support dynamic coalitions within the chemical engineering industry. The forms of dynamic coalitions that this software supports are positioned within the space defined by the seven dimensions.

Anticipated future work includes the development of validation techniques that exploit the formality of the models and development of more detailed knowledge/guidance about the design of dynamic coalitions. We will also use our approach to represent the Domain Based Security approach, and show how that approach interacts with dynamic coalitions.

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

BRYANS, J.W., FITZGERALD, J. S., JONES, C. B., MOZOLEVSKY, I..

Dimensions of Dynamic Coalitions
[By] J. W. Bryans, J. S. Fitzgerald, C. B. Jones and I. Mozolevsky.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2006.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-963)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-963

Abstract

Developments in network technology are enabling organisations to form temporary alliances to achieve specific goals. Such alliances are often referred to as "dynamic coalitions", emphasising the fluid character of their memberships. Dynamic coalitions vary widely in architecture, scale and complexity, ranging from ad hoc groupings of organisations created in order to perform a very brief transaction to long-running collaborations between allies. In many cases, there is significant sharing of information between the participants.

The term "dynamic coalitions" is often used without definition, giving rise to potential confusion and unfulfilled expectations. This paper attempts to map out a space of dynamic coalitions, using a systematic approach supported by a formal (mathematically-based) modelling language. Seven "dimensions" are identified and explored, with an emphasis on the flow of information through coalitions. A case study examines software being developed to support dynamic coalitions within the chemical engineering industry. The forms of dynamic coalitions that this software supports are positioned within the space defined by the seven dimensions.

Anticipated future work includes the development of validation techniques that exploit the formality of the models and development of more detailed knowledge/guidance about the design of dynamic coalitions. We will also use our approach to represent the Domain Based Security approach, and show how that approach interacts with dynamic coalitions.

About the author

Jeremy has been a post doctoral research fellow at the School of Computing Science in Newcastle since December 2002. His background is in Theoretical Computer Science, and he has held posts in Stirling and Canterbury. His work in Newcastle is involved with the security of computer-based systems, and he is employed on the DIRC and GOLD projects.

John Fitzgerald is a specialist in the engineering of dependable computing systems, particularly in rigorous analysis and design tools. He returned to the University in 2003, having established design and validation activities at Transitive, a successful new SME in the embedded processor market.

Cliff Jones is currently Professor of Computing Science and Project of the IRC on “Dependability of Computer-Based Systems”. He has spent more of his career in industry than academia. Fifteen years in IBM saw among other things the creation with colleagues in Vienna of VDM. Cliff is a fellow of the BCS, IEE and ACM. He Received a (late) Doctorate under Tony Hoare in Oxford in 1981 and immediately moved to a chair at Manchester University where he built a strong Formal Methods group which among other projects was the academic partner in the largest Alvey Software Engineering project (IPSE 2.5 created the "mural" theorem proving assistant). During his time at Manchester, Cliff had an SRC 5-year Senior Fellowship and spent a sabbatical at Cambridge with the Newton Institute event on "Semantics". Much of his research at this time focused on formal (compositional) development methods for concurrent systems. In 1996 he moved to Harlequin directing some 50 developers on Information Management projects and finally became overall Technical Director before leaving to re-join academia in 1999. Cliff's interests in formal methods have now broadened to reflect wider issues of dependability.

Igor Mozolevsky is a PhD student at the School of Computing Science at the University of Newcastle. He is a member of the Dependability research group.

Suggested keywords

DYNAMIC COALITIONS,
MODEL-BASED SPECIFICATION
INFORMATION FLOW

Dimensions of Dynamic Coalitions

Jeremy W. Bryans, John S. Fitzgerald,
Cliff B. Jones, Igor Mozolevsky

May 10, 2006

Abstract

Developments in network technology are enabling organisations to form temporary alliances to achieve specific goals. Such alliances are often referred to as "dynamic coalitions", emphasising the fluid character of their memberships. Dynamic coalitions vary widely in architecture, scale and complexity, ranging from *ad hoc* groupings of organisations created in order to perform a very brief transaction to long-running collaborations between allies. In many cases, there is significant sharing of information between the participants.

The term "dynamic coalitions" is often used without definition, giving rise to potential confusion and unfulfilled expectations. This paper attempts to map out a *space* of dynamic coalitions, using a systematic approach supported by a formal (mathematically-based) modelling language. Seven "dimensions" are identified and explored, with an emphasis on the flow of information through coalitions. A case study examines software being developed to support dynamic coalitions within the chemical engineering industry. The forms of dynamic coalitions that this software supports are positioned within the space defined by the seven dimensions.

Anticipated future work includes the development of validation techniques that exploit the formality of the models and development of more detailed knowledge/guidance about the design of dynamic coalitions. We will also use our approach to represent the Domain Based Security approach, and show how that approach interacts with dynamic coalitions.

Contents

1 Introduction

Recent improvements in the capabilities of networking, ambient computing and wireless communication enable individuals and organisations to form collaborations, driven by a desire to cooperate over a long term, or to respond to an acute need. Although each such collaboration is unique, they do have features in common such as the dynamic nature of the collaboration's membership, and the exchange of information between members.

1.1 The Space of Dynamic Coalitions

This paper concerns these forms of collaboration, which are termed "dynamic coalitions". Other terms which are used for similar concepts include "virtual organisations", "virtual enterprises" and "business alliances". Our purpose is not to present a taxonomy which prises these terms apart, but rather to explore the structural concepts that underly them. We will therefore use the term "dynamic coalitions" in its most general possible sense to cover all possible shades of meaning implied by any of its synonyms.

In the literature on management there is a notion of the *virtual breeding environment* [SZGM05] within which companies may create, join and dissolve temporary alliances (virtual organisations or dynamic coalitions). This can be observed in quite diverse domains, for example:

- Dynamic business environments, where companies may form an alliance in order to capitalise on a market opportunity, and terminate the alliance as soon as the opportunity is no longer viable. Here each partner company is motivated to contribute to the alliance by the potential reward they themselves will accrue through the success of the alliance.

- Disaster response scenarios, where groups as diverse as emergency services, military units (possibly from many countries), non-governmental organisations and civil organisations must work together to mitigate the effects of a crisis. Each partner may suspend the pursuit of its own aims, and dedicate all its energies to the immediate problem for a limited period of time.

- Health care, where a range of machines may be reporting on the condition of a patient and medical staff with many different skills must make decisions based on these reports. Here individual coalitions will form around the needs of a patient, and the environment will have been designed precisely to deal best with the kind of emergencies that can arise.

Our particular motivation in exploring this area is to provide a basis for discussing information flow, information security, privacy and trust in dynamic coalitions. Faced with the wide variety of structures that qualify as dynamic coalitions, it is necessary to chart the space of possibilities, to understand better what the character of coalitions might be. The aim of the study presented in this paper is not to provide a prescriptive definition of one notion of dynamic coalition but rather to map out and explore the space of options systematically. Our approach follows that of our colleagues who wrote about the "many meanings" of the oft-used term "open source software" [GA04]. We hope to make a

modest contribution by encouraging people who wish to discuss dynamic coalitions to explain where their use of the term sits in the multi-dimensional space that we describe.

1.2 The Model-based Approach

The tool we choose to help map out the space of dynamic coalitions is a model-based specification technique based on the Vienna Development Method (VDM) [Jon90, FL98]. Such techniques have been used extensively to model computer-based systems. Their mathematical rigour allows a significant level of machine-assisted analysis to be conducted on models, guiding and helping to verify design steps. Here, however, the modelling technique is not being used for the specification of one selected system. Rather, it is employed as a tool for systematic discussion. One of the key benefits of such modelling is *abstraction*: the choices can be understood without the detail of an implementation. Particularly revealing are data type invariants and preconditions. Showing what cannot happen is sometimes more revealing than describing events that can occur.

VDM models emphasise the structure and persistent data (called the *state*) within a computer-based system. For example, the group of members of a coalition forms part of the coalition's state. Changes to the state, such as the act of joining new members to a coalition, are described as *operations*. Similarly, the information held by coalition members forms part of the state and operations describe the transfer of information between members. Although the paper does not describe the modelling language in depth, its major features will become apparent from examples.

Although VDM has been used here, the task might have been equally undertaken with one of the other model-based specification languages such as Z [Hay93] or B [Abr96]. Indeed, model-based specifications are not the only tool that could be used to explore aspects of dynamic coalitions. However, the emphasis that such techniques place on data modelling and structure makes them suitable for describing the functionality involved in forming and operating dynamic coalitions. Other formalisms have other emphases. For example, problems of communication might be better studied using a process algebra such as the π-calculus [MPW92, SW01].

The approach taken in this paper is to propose as basic as possible a model of the state and dynamic operations of a dynamic coalition and then to identify a range of elaborations that can be made to this model, considering the range of coalition types that would result. This gives the "dimensions" along which coalitions may vary.

The state and operations describe structural changes in a coalition, such as coalition formation, adding and removing coalition members, as well as the key operations modelling the dynamic flow of information between members and between members and non-members. We will avoid deep issues such as the distinctions between "data", "information" and "knowledge". Our interest is not in epistemology but rather in the way that abstract (as opposed to physical) objects are shared. The distinguishing characteristic of information in our models is that it may, unlike physical objects, be reproduced an arbitrary number of times.

1.3 A Basic Model

Our most basic model is of a system or *global state* composed of *Agents* which may join and leave collections of agents known as *Coalitions*. This tripartite structure (Figure 1) is present in all our models. Within the global state, referred to as Σ, coalitions are identified by means of coalition identifiers (*Cid*s) and agents are identified by means of agent identifiers (*Aid*s). Note that agents do not have to be members of coalitions to exist within the global state. When we consider each aspect of a coalition (membership, information transfer, provenance, trust etc.) we have to consider where relevant data lies in the system: does membership information reside at the global level, or is it just held within coalitions, or just within agents? Similarly, the responsibility for performing operations may lie at the system level, within coalitions, or with individual agents.

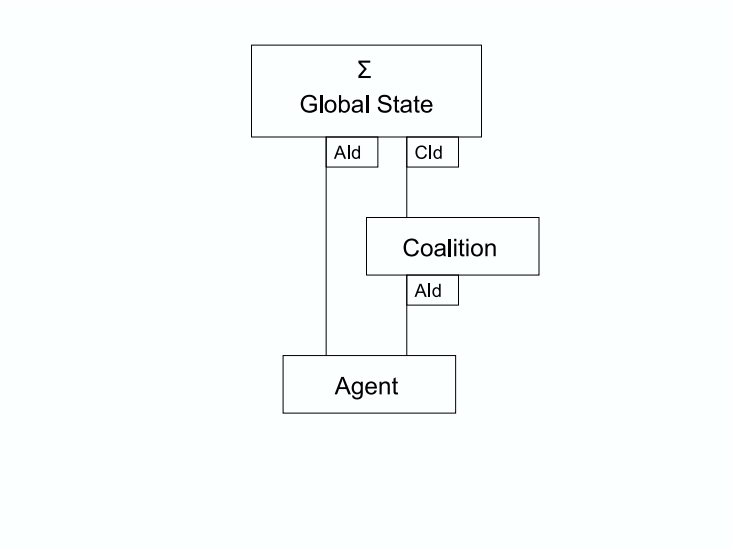


Figure 1: Components of a dynamic coalition

Formalising this structure in our modelling notation requires that we introduce some important data types. The names *Aid* and *Cid* represent the types of possible agent identifiers and coalition identifiers respectively. Elements of both types are defined as *tokens*, meaning that the details of their representations are immaterial[1]. We will introduce a type *Agent* to represent agents. At this stage, we need say nothing more about the interiors of agents, so they are left for definition later. Our basic state is defined as follows in the formal VDM notation[2]:

[1] Throughout the paper, the types that we introduce will be tokens unless explicitly defined otherwise.

[2] The notation used is a derivative of the mathematical notation for ISO Standard VDM-SL [And96]. Some simplifications have been made to ease the presentation.

$$\Sigma :: \begin{array}{rl} coals & : Cid \xrightarrow{m} Aid\text{-}\mathbf{set} \\ agents & : Aid \xrightarrow{m} Agent \end{array}$$

$$\mathbf{inv}\ (coals, agents) \triangleq \bigcup \mathbf{rng}\ coals \subseteq \mathbf{dom}\ agents$$

The definition indicates that the state consists of two components. The first, *coals*, records an association (in VDM, a mapping) between coalition identifiers and the sets of (identifiers of) agents that are members of the coalition. The second component, *agents*, relates agent identifiers to agents. The invariant is a predicate ensuring that the agent identifiers in coalitions are all genuine, i.e. they are known in the *agents* component.

The state Σ is a useful starting point: it assumes the separation of agents as distinct entities, data being stored with agents, and potentially overlapping coalitions of agents. Why have we chosen to model these aspects specifically, and not others? The purpose of the model governs the choice of abstractions made. At one level, we could simply model a collection of agents passing information around: coalition membership, information provenance, communication policies and trust levels are all 'just' pieces of information. Such a model would be simple, flat and nearly useless. As soon as we articulate a model's purpose, we begin to separate out those pieces of information that are particularly relevant to the analysis we wish to perform.

In choosing abstractions for models of dynamic coalitions, we distinguish *information* from *meta-information*. Information is the stock-in-trade of the dynamic coalition: it is the material traded between agents. Meta-information is information *about* the agents, coalitions or information itself. Generally, each of the dimensions we explore corresponds to a form of meta-information and the models that we develop make the relevant meta-information explicit. In our basic model, Σ separates coalition membership out because this is what makes a coalition dynamic. Later in the paper, we will separate out other aspects such as those mentioned above, in order to explore their interactions.

1.4 Dimensions of Dynamic Coalitions

In the body of this paper, we consider seven aspects or dimensions of dynamic coalitions that all relate to the flow of information through coalitions. In each case, we consider the alternative ways in which Σ might be extended to accommodate the relevant data and functionality. This in turn suggests ways in which to position particular dynamic coalition architectures. Each discussion of a dimension concludes with a short list of important considerations relating to that dimension, intended to be useful to system architects developing dynamic coalition environments.

We view a coalition's membership as the most fundamental aspect of its character. This is explored in Section 2. Since our goal is to be able to understand better the flow of information between coalition participants, we then explore the possible models of information storage in coalitions (Section 3) and hence the communication of information between agents (Section 4).

Having reviewed the possible mechanisms for basic membership and information flow, we examine the structures built on top of these to govern and exploit information flow. The authority structure within a coalition (Section 5) governs information transfer since it affects the rights to access information.

Given that information may be transferred, a key factor in exploiting the information itself is its provenance (Section 6). It is important to note that the value of information itself changes over time and so we examine this as a separate dimension in Section 7. Together, the decisions made about membership, information transfer, authority, provenance and time all affect the trust that is placed in information transferred (Section 8). As a case study Section 9 considers software developed for supporting dynamic coalitions within the chemical industry. We describe the range of dynamic coalitions supported by this software as a subspace within our dimensions. The VDM models for this work may be found at [Mod].

2 Coalition membership

This section aims to define basic concepts relating to the membership of dynamic coalitions. The intention is that these models should form a basis for further exploration.

In Section 1 we identified three levels of potential responsibility in our models, associated with the three main entities: agents, coalitions and the global state. Membership meta-information is a relation between agents and coalitions. It is already present in the Σ model of Figure 1.3, so this model is used as a starting point. Almost as soon as one considers modelling coalition membership, one is faced with the key question: where does responsibility lie for performing the operations that join agents to coalitions, or remove them?

In this section, we illustrate the exploratory approach based on formal modelling rather than give a "finished" set of formal definitions. We begin by examining the join/leave functionality as an extension of the Σ model. This attempt at formalisation raises further issues. In particular, it leads to an examination of alternative models in which membership responsibility is held at the system level (Section 2.2), and even at the agent level (Section 2.3). It also raises the question of membership authorisation (Section 2.4).

2.1 Membership as a Coalition Responsibility

The basic Σ model in Section 1.3 already expresses membership meta-information, so it is possible to define the joining and leaving operations over this model. It is worth noting that, at this stage, the model is neutral about who is authorised to perform membership operations.

This model takes a coalition-oriented view of membership in the sense that the agent identifiers are associated with their individual coalitions. In order to describe the act of joining an agent to a coalition, we give an operation specification:

$$
\begin{array}{l}
\mathit{Join}\ (a\colon \mathit{Aid}, c\colon \mathit{Cid}) \\
\textbf{ext}\ \textbf{wr}\ \mathit{coals}\ :\ \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\textbf{set} \\
\quad\ \ \textbf{rd}\ \mathit{agents}\ :\ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\textbf{pre}\ a \in \textbf{dom}\ \mathit{agents} \wedge c \in \textbf{dom}\ \mathit{coals} \wedge a \notin \mathit{coals}(c) \\
\textbf{post}\ \mathit{coals} = \overleftarrow{\mathit{coals}} \dagger \{c \mapsto \overleftarrow{\mathit{coals}}(c) \cup \{a\}\}
\end{array}
$$

This is an *implicit* operation definition in the sense that a postcondition is used to characterise the global state after the operation. This is in contrast to an explicit style in which an algorithm is given for performing the computation. Since we are concerned primarily with the effects of operations rather than the algorithms used to accomplish them, we will use this style throughout the paper.

The precondition in the *Join* operation is required to ensure that the agent and coalition are both known, and that the agent is not already a member of the coalition. The *Remove* operation performs the inverse:

$$
\begin{array}{l}
\mathit{Remove}\ (a\colon \mathit{Aid}, c\colon \mathit{Cid}) \\
\textbf{ext wr}\ \mathit{coals} : \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\textbf{set} \\
\textbf{pre}\ c \in \textbf{dom}\ \mathit{coals} \land a \in \mathit{coals}(c) \\
\textbf{post}\ \mathit{coals} = \overleftarrow{\mathit{coals}} \dagger \{c \mapsto \overleftarrow{\mathit{coals}}(c) \setminus \{a\}\}
\end{array}
$$

Both of these operations require that the coalition already exists prior to the addition or removal of a member. When the model is extended to encompass information (Section 3.2), it becomes possible to model alternative decisions about what happens to information within a coalition when an agent leaves.

The *Remove* operation in particular opens the possibility that the agent is the last one left in a coalition. Does a coalition continue to exist after it has lost all its members? The basic model Σ allows for the possibility of creating initially empty coalitions. An operation to do this could be modelled as follows:

$$
\begin{array}{l}
\mathit{CreateEmptyCoal}\ (c\colon \mathit{Cid}) \\
\textbf{ext wr}\ \mathit{coals} : \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\textbf{set} \\
\textbf{pre}\ c \notin \textbf{dom}\ \mathit{coals} \\
\textbf{post}\ \mathit{coals} = \overleftarrow{\mathit{coals}} \cup \{c \mapsto \{\,\}\}
\end{array}
$$

This allows coalitions to have a separate existence from their members. Alternatively, if coalitions only have an existence in their members, it would be appropriate to exclude empty coalitions, leading to model Σ_m:

$$
\begin{array}{lll}
\Sigma_m :: & \mathit{coals} & : \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\textbf{set} \\
 & \mathit{agents} & : \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\multicolumn{3}{l}{\textbf{inv}\ (\mathit{coals}, \mathit{agents}) \triangleq \bigcup \textbf{rng}\ \mathit{coals} \subseteq \textbf{dom}\ \mathit{agents} \land \{\,\} \notin \textbf{rng}\ \mathit{coals}}
\end{array}
$$

In this case, coalition creation would require at least one member. Removing a member from a coalition could also entail taking the coalition out of existence. The operation specifications that such a change leads to are given in Appendix A. In model Σ_m coalitions can be destroyed by (individually) removing all the members and an operation is also included to explicitly dissolve a coalition.

2.2 Membership as a Global Responsibility

The following model Σ_{mg} demonstrates the effect of pulling membership meta-information out into a separate component of the global state. This reflects a decision to make joining and leaving coalitions a system-level responsibility. The additional state component *membs* is a relation between agent and coalition identifiers.

Σ_{mg} :: $coals$: Cid-**set**
$agents$: $Aid \xrightarrow{m} Agent$
$membs$: $(Aid \times Cid)$-**set**

inv $(coals, agents, membs) \triangleq membs \subseteq \{(a, c) \mid a \in \mathbf{dom}\, agents, c \in coals\}$

In such a model, the *Join* and *Remove* operations might appear as follows.[3]

Join $(a{:}\, Aid, c{:}\, Cid)$
ext rd $coals$: Cid-**set**
rd $agents$: $Aid \xrightarrow{m} Agent$
wr $membs$: $(Aid \times Cid)$-**set**
pre $a \in \mathbf{dom}\, agents \wedge c \in coals \wedge (a, c) \notin membs$
post $membs = \overleftarrow{membs} \cup \{(a, c)\}$

Remove $(a{:}\, Aid, c{:}\, Cid)$
ext rd $coals$: Cid-**set**
wr $membs$: $(Aid \times Cid)$-**set**
pre $(a, c) \in membs$
post $membs = \overleftarrow{membs} \setminus \{(a, c)\}$

2.3 Membership as an Agent Responsibility

For completeness, we also consider the possibility of coalition membership being a property of the agents themselves. The corresponding model pushes the membership information into the agents:

Σ_{ma} :: $coals$: Cid-**set**
$agents$: $Aid \xrightarrow{m} Cid$-**set**

inv $(coals, agents) \triangleq \bigcup \mathbf{rng}\, agents \subseteq coals$

In this model, agents can be seen as "voting themselves" into membership by simply declaring themselves members of coalitions. The corresponding joining and removal operations are as follows:

Join $(a{:}\, Aid, c{:}\, Cid)$
ext wr $coals$: Cid-**set**
rd $agents$: $Aid \xrightarrow{m} Cid$-**set**
pre $a \in \mathbf{dom}\, agents \wedge c \in \mathbf{dom}\, coals \wedge c \notin coals(a)$
post $agents = \overleftarrow{agents} \dagger \{a \mapsto \overleftarrow{agents}(a) \cup \{c\}\}$

Remove $(a{:}\, Aid, c{:}\, Cid)$
ext rd $coals$: Cid-**set**
wr $agents$: $Aid \xrightarrow{m} Cid$-**set**
pre $c \in \mathbf{dom}\, coals \wedge a \in coals(c)$
post $agents = \overleftarrow{agents} \dagger \{a \mapsto \overleftarrow{agents}(a) \setminus \{c\}\}$

[3] For this model, we retain the view that coalitions have an existence even when they have no members.

This model, and the two previous ones, permit agents to disagree about the membership of coalitions. If an agent is not able to access the *coals* component in the global state, then all it can know about who else is in a coalition is what is communicated to it from other agents.

2.4 Membership Authorisation Schemes

We have considered membership responsibility at different levels in the agent-coalition-system structure. However, more sophisticated models are possible. For example, support for joining and leaving decisions may have to be gathered from more than a certain threshold of existing coalition members. This threshold value must be recorded within the coalition structure, leading to model Σ_{auth}:

$$\begin{array}{l} \mathit{Coalition} :: \mathit{members} : \mathit{Aid}\text{-}\mathbf{set} \\ \qquad\qquad\quad \mathit{threshold} : \mathbb{R} \\ \mathbf{inv}\ (\text{-}, \mathit{threshold}) \triangleq 0 \leq \mathit{threshold} \wedge \mathit{threshold} \leq 1 \end{array}$$

$$\begin{array}{l} \Sigma_{auth} :: \quad \mathit{coals} : \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\ \qquad\qquad\ \ \mathit{agents} : \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\ \mathbf{inv}\ (\mathit{coals}, \mathit{agents}) \triangleq \bigcup\{c.\mathit{members} \mid c \in \mathbf{rng}\ \mathit{coals}\} \subseteq \mathbf{dom}\ \mathit{agents} \end{array}$$

The full model, including operations to create coalitions, and join and remove agents is provided in Appendix B.

Yet more elaborate membership authorisation schemes could be envisaged and modelled. For example, the model could itself be generic with a set of parameters governing membership determination. Many schemes for joining and leaving groups are to be found in the literature on group membership in distributed systems, particularly dealing with limitations on the decidability of membership [Cri91, CHTCB96].

A different view of coalition formation would be to allow agents to have initial knowledge of the existence of all the other agents in the environment. This could be done by assuming an inclusive coalition in which the only shared information is the identities of all the agents. Agents can then evolve more intimate coalitions by forming subsets of this set.

2.5 The Dimension of Dynamic Coalition Membership

Through the modelling work discussed in this section, we have identified several issues relevant to positioning a dynamic coalition architecture in the membership dimension. We might expect the architect of a system of dynamic coalitions to ask the following important questions:

- Does the existence of a coalition depend on there being one or more members?
- Should membership (joining and leaving) be an agent-level responsibility, so that agents can vote themselves in to coalitions?
- Should membership be a coalition responsibility so that a coalition records in some place the current membership roll and admits or ejects members?
- Should membership be a system-level responsibility, with agents effectively being placed in coalitions by *force majeure*?

- Are there preconditions on the membership joining and leaving operations?

We do not intend to imply any exclusiveness between the answers to these questions. Indeed, membership management capabilities may reside at more than one level in the agent-coalition-system hierarchy.

3 Information

In order to model the flow of information in dynamic coalitions, it is necessary to decide on a representation for information, its storage and creation within the model. This section explores these dimensions, while Section 4 deals with information flow itself.

Several significant abstraction decisions have to be made regarding information. It has already been noted that we use the term information in a general sense to describe the data traded between agents in a coalition, and between coalitions and their environment. Given that the purpose of the model to analyse information flow, rather than accuracy of information with respect to some external "real world", we will also refrain from attempting to model this semantic relationship.

The choice of a representation for information is addressed in Section 3.1. The location of information at various levels in our agent-coalition-system hierarchy is then presented (Section 3.2). The model must necessarily address the ways in which information may be created within the model (Section 3.3). One semantic issue, namely that of information consistency, is also dealt with at this level.

3.1 Representing and Identifying Information

There are numerous ways of representing information. A naïve representation might be a string of bits. This captures no context and allows almost no semantic comparisons between two pieces of information (apart from string equality and subsetting). A more structured way of representing information is to consider it as a series of n-tuples. Such tuples can impose structure on information. For example, the RDF framework can be represented as a simple (subject, object, predicate) triple which can encode a statement such as 'car speed is 30 mph' as $(car, speed, 30mph)$. The use of triples as the basis for information storage is well established as an underpinning for knowledge base technology[4]. A VDM development of a 3-tuple database is described by Welsh [Wel84]. Epistemic logic [FHMV95] represents information as Boolean predicates, and provides powerful operators for reasoning about these predicates in a distributed context.

Models of specific coalitions may choose common information representation frameworks. However, for the purposes of modelling information flow, the models we develop here are neutral about the particular representation chosen. We will use the data type *Information* to stand for the chosen representation, and treat this as a collection of unstructured tokens. The only semantic operator on these tokens is the test for equality.

[4]See, for example, `http://threestore.sourceforge.net`.

Information is an unusual kind of resource in that it may be copied arbitrarily, as well as transfered. If an information item is copied, and we wish to distinguish the copy from the original, it is necessary to identify each item by means of a key. In this case, where individual items have unique keys as identifiers, it is necessary to maintain a mapping from the keys to the information values (formally, $\mathit{InfoKey} \xrightarrow{m} \mathit{Information}$). An alternative is simply to regard *Information* items as unkeyed values and so to regard the collection of information as *Information*-**set**. There are advantages for the mapping model because one can discuss, for example, the visibility of information in terms of the sets of *InfoKey*. However, this approach makes the discussion of copying more difficult. Another reason for preferring the set structure in what follows is that is is always possible to embed a "key" within *Information*, although this has to be done with an awareness of "normal forms" and any requirements for uniqueness. The decision about whether to use a mapping-based or set-based model is likely to vary between applications. Indeed, in Section 9, we discuss a specific coalition architecture in which the mapping approach is preferred.

3.2 Locating Information in the Model

Where should information reside? Taking the basic model Σ as a starting point again, we can envisage information being held at agent, coalition and global levels, as was the case for membership. At the agent level, the *Agent* data type can be augmented with an information store. In keeping with the discussion in Section 3.1, this would be a set of *Information* tokens (or a mapping from information identifiers to tokens if appropriate). The resulting definition of *Agent* would be:

$$\mathit{Agent} :: \mathit{info} : \mathit{Information}\text{-}\mathbf{set}$$

Shared information, common to members of a coalition, would reside at the coalition level. The coalition model is therefore more than just the set of member identifiers, and has its own information set. We introduce the type *Coalition* to model this:

$$\begin{aligned} \mathit{Coalition} :: \mathit{info} &: \mathit{Information}\text{-}\mathbf{set} \\ \cdots &: \cdots \end{aligned}$$

Global common knowledge, shared across the space of all coalitions, is at the outermost level, leading to a new version of Σ:

$$\begin{aligned} \Sigma_{loc} :: \quad \mathit{info} &: \mathit{Information}\text{-}\mathbf{set} \\ \mathit{coals} &: \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\ \mathit{agents} &: \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{aligned}$$

We have opted for what is intended to be an intuitive model here. It is worth noting that other models are possible. For example, we might have a single place for storing information at the coalition level. Agent-specific information would be modelled by artificially constructing single member coalitions each with their own information store; global information would be modelled by an artificial universal coalition.

3.3 Creating and Sharing Information

The layered model Σ_{loc}, with information repositories at agent, coalition and global levels, supports analysis of the ways in which information can be ini-

tially created, moved between layers, and even lost. For example, one model allows for information to be created by individual agents and then shared by moving it to the coalition-level store and even to the global level. At the bottom of the hierarchy, information is "created" through direct acquisition from empirical observation, for example through sensors, by deduction from existing information, or by transfer between agents. Section 4 addresses the wide range of models for (authorised) information transfer. In this section, we consider the basic operations of information creation and sharing "up the hierarchy" that can be described over the Σ_{loc} model.

Acquiring Information

Information can be acquired directly from sensors or by interaction with agents in the environment. If we choose not to model the source of information explicitly, we must include within agents an operation such as *discover*, which adds new set of *Information* tokens to the agent's knowledge base[5]:

$$
\begin{array}{l}
\mathit{Discover}\ (a{:}\ \mathit{Aid}, \mathit{is}{:}\ \mathit{Information}\text{-}\mathbf{set}) \\
\mathbf{ext\ wr}\ \mathit{agents}\ :\ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\mathbf{pre}\ a \in \mathbf{dom}\ \mathit{agents} \\
\mathbf{post}\ \mathit{agents} = \overleftarrow{\mathit{agents}} \dagger \{a \mapsto \mu(\overleftarrow{\mathit{agents}}(a), \mathit{info} \mapsto \overleftarrow{\mathit{agents}}(a).\mathit{info} \cup \mathit{is})\}
\end{array}
$$

Since the model is intended for the analysis of information transfer and is not concerned with modelling the external environment, this operation is almost trivial, simply allowing information to appear in the model. The operation is defined at the agent level (it works on the information store of a specific agent). One might envisage a similar operation being available at the coalition level, if the coalition has its own ability to acquire information independent of the participating agents. It is somewhat harder to motivate a version of the operation at the global level.

Inferring Information

Agents can have the ability to perform some basic inference on their own information sets. As with information acquisition, so far as the present model is concerned, this is just another source of new information tokens. A single inference step may be modelled as a function that generates a set of additional tokens:

$$
\begin{array}{l}
\mathit{Infer}\ (\mathit{info}_{in}{:}\ \mathit{Information}\text{-}\mathbf{set})\ \mathit{info}_{out}{:}\ \mathit{Information}\text{-}\mathbf{set} \\
\mathbf{post}\ \mathit{info}_{in} \subseteq \mathit{info}_{out}
\end{array}
$$

The transitive closure of the function represents the process of inferring all the possible facts that can be deduced from an existing *Information*-**set**.

[5]The μ operator used in the postcondition describes a change to a single component of a record structure. In the postcondition of this operation, it refers to the *info* component of the agent a.

Sharing Information

The layered model permits the definition of operations describing the movement of information from the agent level to coalition level, a form of sharing:

$$
\begin{array}{l}
\mathit{Share}\ (a{:}\ \mathit{Aid}, c{:}\ \mathit{Cid}, \mathit{is}{:}\ \mathit{Information}\text{-}\mathbf{set}) \\
\mathbf{ext\ wr}\ \mathit{coals}\ \ :\ \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\
\quad\ \mathbf{rd}\ \mathit{agents}\ :\ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\mathbf{pre}\ a \in \mathbf{dom}\ \mathit{agents} \wedge \mathit{is} \subseteq \mathit{agents}(a).\mathit{info} \wedge c \in \mathbf{dom}\ \mathit{coals} \\
\mathbf{post}\ \mathit{coals} = \overleftarrow{\mathit{coals}} \dagger \{c \mapsto \mu(\overleftarrow{\mathit{coals}}(c), \mathit{info} \mapsto \overleftarrow{\mathit{coals}}(c).\mathit{info} \cup \mathit{is})\}
\end{array}
$$

A similar operation could promote information from coalition to global level, placing it in the *info* field of Σ_{loc}. In both of these cases, there is a question of granting authorisation to perform these kinds of operation. These topics are explored in Section 4.

Losing Information

The layered model of information storage represented in Σ_{loc} leads us to consider the fate of information when agents leave coalitions or coalitions are themselves dissolved. When an agent leaves a coalition, the information held within the agent may be lost to the coalition. A protocol might be employed which requires the sharing of certain information (placing at the coalition level) before permission for departure is granted. In the opposite direction, the agent may be able to copy coalition-level information into its individual store prior to departure from the coalition. At coalition dissolution, information stored at the coalition level could be deleted or migrated up to the global level, or copied or distributed among the agents in the former coalition.

If the information stores are shared repositories, it is possible to model agents losing access to information but this raises questions about which part of the store the departing agent had accessed and when. These questions would be particularly pertinent if the departing agent may have hostile intent. Countermeasures here include changing information so that the knowledge itself is altered so that the departing agent's knowledge is no longer meaningful? This is practiced regularly with such secrets as group keys. In general, this brings up the question of validity of information over time which is discussed in Section 7.

3.4 The Dimension of Information

The layered model of information storage has allowed us to identify issues relating to the representation and storage of information, the creation, acquisition and loss. The developer of a dynamic coalition scheme might wish to consider the following questions:

- At what level of detail is information represented?
- Are units of information identified by unique keys?
- Can information be stored at the agent, coalition and/or global levels?
- How is information acquired and added to information stores?

- Can information be moved between agent, coalition and global levels? In either direction?
- What happens to information when an agent leaves a coalition, or a coalition is dissolved?

4 Information Transfer

An underlying purpose of our modelling activity is to be able to analyse the flow of information through a coalition. In Section 3.3 we identified the options for moving information between levels in the agent-coalition-global structure. In this section, we consider the options for movement of information between agents. The models developed in this section describe the functionality of information transfer on the basis of the meta-information about membership and information discussed in previous sections. At this level, the model is not concerned with mechanisms of transmission, so much as the preconditions: who can participate in an information transfer, and what can be transmitted?

As a starting point, consider the original base model Σ. We will treat the information stored in each agent as a set of *Information* tokens.

$$\begin{array}{lrcl} \Sigma_{simp} :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{array}$$

$$\mathbf{inv}\ (\mathit{coals}, \mathit{agents}) \triangleq \bigcup \mathbf{rng}\ \mathit{coals} \subseteq \mathbf{dom}\ \mathit{agents}$$

$$\mathit{Agent} = \mathit{Information}\text{-}\mathbf{set}$$

4.1 Naïve Information Transfer

The most basic operation is copying a set of *Information* from one agent to another:

$$\mathit{InfoTransfer}\ (\mathit{from}, \mathit{to}{:}\ \mathit{Aid}, \mathit{is}{:}\ \mathit{Information}\text{-}\mathbf{set})$$

$$\mathbf{ext\ wr}\ \mathit{agents} : \mathit{Aid} \xrightarrow{m} \mathit{Agent}$$

$$\mathbf{pre}\ \{\mathit{from}, \mathit{to}\} \subseteq \mathbf{dom}\ \mathit{agents} \wedge \mathit{is} \subseteq \mathit{agents}(\mathit{from})$$

$$\mathbf{post}\ \mathit{agents} = \overleftarrow{\mathit{agents}} \dagger \{\mathit{to} \mapsto \overleftarrow{\mathit{agents}}(\mathit{to}) \cup \mathit{is}\}$$

The second conjunct of the precondition requires that the information to be transferred is known by the "from" agent.

4.2 Agent Clearance of Information

It is likely that individual agents will operate policies regarding the clearing of information for transfer. One would wish to incorporate within an agent a decision-making procedure for granting clearance. At a simple level, this could be a function mapping each piece of *Information* to the set of potential recipients. Holding such a function complicates the *Agent* type:

$$\begin{array}{lrcl} \mathit{Agent} :: & \mathit{info} & : & \mathit{Information}\text{-}\mathbf{set} \\ & \mathit{clearance} & : & \mathit{Information} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \end{array}$$

$$\mathbf{inv}\ (\mathit{info}, \mathit{clearance}) \triangleq \forall i \in \mathbf{dom}\ \mathit{clearance} \cdot i \in \mathit{info}$$

$\Sigma_{a\text{-}tr}$ :: $coals$: $Cid \xrightarrow{m} Aid\text{-}\mathbf{set}$
$agents$: $Aid \xrightarrow{m} Agent$

inv $(coals, agents) \triangleq \bigcup \mathbf{rng}\ coals \subseteq \mathbf{dom}\ agents \wedge$
$\bigcup\{\bigcup \mathbf{rng}\ ags.clearance \mid ags \in \mathbf{rng}\ agents\} \subseteq \mathbf{dom}\ agents \wedge$
$\ldots$

The extra invariant clause on the type *Agent* asserts that all the information which may be revealed by an agent is known by that agent. The second conjunct of the invariant on $\Sigma_{a\text{-}tr}$ asserts that all the agents to whom information may be revealed are known.

The *clearance* component of the agent state is not very practical — it needs to be updated every time a new piece of information is added — but serves to illustrate the basic point. The new component can be used to govern information transfer:

InfoTransfer $(from, to$: Aid, is: $Information\text{-}\mathbf{set})$
ext wr $agents$: $Aid \xrightarrow{m} Agent$
pre $\{from, to\} \subseteq \mathbf{dom}\ agents \wedge$
$\forall i \in is \cdot to \in agents(from).clearance(i) \wedge$
$is \subseteq agents(from)$
post $agents = \overleftarrow{agents} \dagger \{to \mapsto \mu(\overleftarrow{agents}(to), info \mapsto \overleftarrow{agents}(to).info \cup is)\}$

Alternative policies may be pursued, for example returning an explicit error if the clearance precondition is not satisfied, or transferring the cleared subset of is: $is \setminus \{i \in is \mid to \notin \overleftarrow{agents}(from).clearance(i)\}$.

4.3 Coalition-based Clearance of Information

While individual agents may pursue their own information clearance policies, coalitions are likely to have their own rules. For the present, ignore the agent clearance notion introduced in Section 4.2 and revert to the basic model Σ_{simp} introduced at the beginning of this section.

If it is required that a transfer is conditional on the "to" agent being in a common coalition with the "from" agent, the relevant precondition can be written:

InfoTransfer $(from, to$: Aid, is: $Information\text{-}\mathbf{set})$
ext rd $coals$: $Cid \xrightarrow{m} Aid\text{-}\mathbf{set}$
wr $agents$: $Aid \xrightarrow{m} Agent$
pre $\{from, to\} \subseteq \mathbf{dom}\ agents \wedge$
$\exists c \in \mathbf{dom}\ coals \cdot \{from, to\} \subseteq coals(c) \wedge$
$is \subseteq agents(from)$
post $agents = \overleftarrow{agents} \dagger \{to \mapsto \overleftarrow{agents}(to) \cup is\}$

The precondition of this function is rather weak, merely requiring that there exists a coalition containing the two agents. In practice, membership of different coalitions is likely to confer different privileges, so it is more appropriate that a specific coalition is identified as the one under which the transfer is taking place. This would be supplied as an input to the information transfer operation:

$$
\begin{array}{l}
\mathit{InfoTransfer}\ (\mathit{from}, \mathit{to}\colon \mathit{Aid}, \mathit{is}\colon \mathit{Information}\text{-}\mathbf{set}, c\colon \mathit{Cid}) \\
\mathbf{ext}\ \mathbf{rd}\ \mathit{coals} \quad : \ \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\
\quad\ \mathbf{wr}\ \mathit{agents}\ : \ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\mathbf{pre}\ \{\mathit{from}, \mathit{to}\} \subseteq \mathbf{dom}\ \mathit{agents} \wedge \\
\quad c \in \mathbf{dom}\ \mathit{coals} \wedge \{\mathit{from}, \mathit{to}\} \subseteq \mathit{coals}(c) \wedge \\
\quad \mathit{is} \subseteq \mathit{agents}(\mathit{from}) \\
\mathbf{post}\ \mathit{agents} = \overleftarrow{\mathit{agents}} \dagger \{\mathit{to} \mapsto \overleftarrow{\mathit{agents}}(\mathit{to}) \cup \mathit{is}\}
\end{array}
$$

This allows the information to be tagged with the coalition under which it was transferred (see Section 6). It is reasonable to suppose that coalition membership should play a role in the clearance decision. There should be a place for coalition-level policies to be expressed under which an information transfer takes place. As with agent-level clearance, in that it is necessary to define a place for recording the policy, this time at the coalition level. A type to represent the coalition state might be introduced:

$$
\begin{array}{lll}
\mathit{Coalition} & :: & \mathit{members} \ : \ \mathit{Aid}\text{-}\mathbf{set} \\
& & \mathit{clearance} \ : \ \mathit{Information} \xrightarrow{m} \{\textsc{MembsOnly} \mid \textsc{Everyone}\}
\end{array}
$$

This involves a mapping of individual atoms to enumerated values indicating who may receive the information: in this case either to members of the coalition or to anyone. More sophisticated policies are possible. Again, the abstraction in the model means that it is necessary to update the coalition every time a new unit *Information* comes into existence.

With this model of a coalition, the overall state needs to be revised:

$$
\begin{array}{lll}
\Sigma_{c\text{-}tr} & :: & \mathit{coals} \ : \ \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\
& & \mathit{agents} \ : \ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\multicolumn{3}{l}{\mathbf{inv}\ (\mathit{coals}, \mathit{agents}) \triangleq \bigcup\{\mathit{cs}.\mathit{members} \mid \mathit{cs} \in \mathbf{rng}\ \mathit{coals}\} \subseteq \mathbf{dom}\ \mathit{agents} \wedge \ldots}
\end{array}
$$

and the information transfer operation becomes:

$$
\begin{array}{l}
\mathit{InfoTransfer}\ (\mathit{from}, \mathit{to}\colon \mathit{Aid}, \mathit{is}\colon \mathit{Information}\text{-}\mathbf{set}, c\colon \mathit{Cid}) \\
\mathbf{ext}\ \mathbf{rd}\ \mathit{coals} \quad : \ \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\
\quad\ \mathbf{wr}\ \mathit{agents}\ : \ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\
\mathbf{pre}\ \{\mathit{from}, \mathit{to}\} \subseteq \mathbf{dom}\ \mathit{agents} \wedge \\
\quad c \in \mathbf{dom}\ \mathit{coals} \wedge \mathit{from} \in \mathit{coals}(c).\mathit{members} \wedge \\
\quad (\mathit{to} \notin \mathit{coals}(c).\mathit{members} \\
\quad \Rightarrow\ \forall i \in \mathit{is} \cdot \mathit{coals}(c).\mathit{clearance}(i) = \textsc{Everyone}) \wedge \\
\quad \mathit{is} \subseteq \mathit{agents}(\mathit{from}) \\
\mathbf{post}\ \mathit{agents} = \overleftarrow{\mathit{agents}} \dagger \{\mathit{to} \mapsto \overleftarrow{\mathit{agents}}(\mathit{to}) \cup \mathit{is}\}
\end{array}
$$

Again, various policies could be described if the precondition is unfulfilled: returning an error message or transferring a subset of *is*.

4.4 The Dimension of Information Transfer

The modelling activity in this section allows us to identify more questions that the developer of a coalition scheme may wish to ask regarding the flow of information through the coalition structure:

- Who can participate in a transfer of information?

- Who may initiate a transfer of information?
- What information may be transfered?
- Where in the structure does control of the clearance mechanisms reside?

More subtle problems may be brought to light as well. For example, if the two information transfer models above are combined, we have the basis of a layered model for information transfer. Setting aside the question of how policies are to be described (the richness of the policy language), there are issues relating to the potential for conflict between policies at different levels.

5 Authorisation Structure

Various "structures" of dynamic coalitions have been proposed in the literature, mostly limited to information transfer (e.g. [Let01]). For example, a *star* structure requires that a single agent acts as the nexus for all communications in the coalition. A *tree* structure has a collection of local star structures with communications passing up from centres to the next level. A *peer to peer* structure may prescribe no communication routes at all. One may imagine a range of hybrid structures combining these options such as a peer-to-peer collection of agents, each of which is at the centre of its own local star structure.

Our modelling work suggests that information transfer structures describe just part of a coalition's character. To us, a key aspect of a coalition's structure is its governance: which agents may authorise specific acts such as information transfer or membership operations. In particular, our use of pre/postcondition specifications for operations governing membership and information transfer emphasises that these operations may require permission. For example, in Section 2.4, we considered the possibility that coalition members might have to vote on the acceptance of a new member, and in Section 4 we considered several levels at which clearance might be given for information transfer. In both cases, some form of authorisation structure was built into the model, rather implicitly, to cater for this possibility. This is rather like coalitions as they are implemented today, with authorisation structures existing only as a consequence of rights, obligations and privileges which are agreed, known and held within the coalition. The explicit, early consideration of these structures is likely to have a significant effect on the design of a robust coalition.

In this section, we consider authorisation structure more explicitly and in a more general setting, independent of the operations being authorised. The meta-information expressed in the models developed in this section explicitly concerns the authorisation structures which may be added to any of the models developed so far, allowing forms such as star and tree structures to be represented for authorisation as well as communication (Section 5.1). We show how this structure is linked to membership management and information transfer operations (Section 5.2) and finally consider the effect of making the authorisation structure dynamic (Section 5.3).

5.1 Representing the Authorisation Structure

In constructing a formal model, we are forced to consider exactly what is meant by *authorisation structure*. For the moment, consider a single operation or

group of operations, such as those performing information transfer, which share a common authorisation structure. Each operation may require authorisation from one or more specific agents. We thus consider an authorisation structure as a relation between agents. Formally, we could define a data type representing such a relation:

$$AuthRel = (Aid \times Aid)\textbf{-set}$$

Given a specific authorisation relation *auth*, we will write a_1 *auth* a_2 to indicate that (a_1, a_2) is in the relation, so a_1 is capable of authorizing an operation by a_2 from the group of operations under consideration.

The definition of such a data type in a modelling language naturally leads one to ask about the structure's properties: could it be reflexive, symmetric, transitive? Should it be acyclic? If any of these properties are required, they could be described in an invariant. For example, we may require an authorisation relation to be acyclic[6]:

$$AuthRel = (Aid \times Aid)\textbf{-set}$$

$$\textbf{inv-}\ auth\ (a) \triangleq \forall as \subseteq \textbf{dom}\ auth \cdot as \neq \{\} \Rightarrow \exists a \in as \cdot \neg(\exists a' \in as \cdot a\ auth\ a')$$

where **dom** *auth* denotes the domain of a relation[7].

Given such a general model, it is possible to describe common structures by means of combinations of conditions on the relation. For example, a *star* authorisation structure is one in which a single agent permits or denies actions. A tree-based authorisation structure is characterised by a constraint that every agent that is subject to authorisation is subject to only one authoriser.

5.2 Adding Authorisation to a DC Model

If authorisation is to be treated explicitly in coalition development, the authorisation structure should be represented in the system state. It is then referenced by operation preconditions. For the moment, we continue to consider a single authorisation structure for a single group of operations, bearing in mind that one could potentially have several such (orthogonal) structures for different kinds of operation.

As an example, we will consider the addition of an authorisation structure to a model derived from the model for information storage, Σ_{loc}, defined in Section 3.2. The basic model is:

$$\begin{array}{l}\Sigma_{auth} :: \quad coals : Cid \xrightarrow{m} Coalition \\ \qquad\qquad\ agents : Aid \xrightarrow{m} Agent \\ \textbf{inv}\ (coals, agents) \triangleq \bigcup\{c.members \mid c \in \textbf{rng}\ coals\} \subseteq agents \\ \\ Coalition :: members : Aid\textbf{-set} \\ \qquad\qquad\qquad\ info : Information\textbf{-set}\end{array}$$

[6]This is just an example of a possible invariant. Other invariants may be required for other forms of authorisation relation. For example, this invariant does not permit an agent to authorise itself to perform a specific act.

[7]The domain (**dom**) and range(**rng**) operators in VDM-SL are normally confined to mappings, but here we extend them to general relations.

$Agent = Information\text{-}\mathbf{set}$

The first question to address is whether the authorisation structure is to be defined globally, as a single relation at the outermost level of the state, or on a per-coalition basis as a component of a coalition, or even on a per-agent basis. Here, we illustrate the issues based on an authorisation structure defined at the coalition level, as this seems to be a likely scenario. The *Coalition* data type would be extended with the extra component as follows:

$$\begin{array}{rrcl} Coalition :: & members & : & Aid\text{-}\mathbf{set} \\ & info & : & Information\text{-}\mathbf{set} \\ & auth & : & AuthRel \end{array}$$

$$\mathbf{inv}\ (members, \text{-}, auth) \triangleq \mathbf{dom}\ auth \cup \mathbf{rng}\ auth \subseteq members$$

The invariant contains two clauses to ensure consistency with the existing components of the coalition. Other conditions, such as acyclicity of the authorisation relation within the coalition, are inherited from the *AuthRel* type. It would be possible to enforce other properties such as transitivity as required.

The link from the authorisation structure to operations is made via their preconditions. For example, consider an information transfer operation such as that given in Section 4.1, operating on the Σ_{auth} state. It is important to know which coalition this transfer falls under. This is significant, as the two members may be included in a number of coalitions, which would lead to an ambiguity in selecting the correct authorisation relationship.

$$\begin{array}{l} InfoTransferAuth\ (from, to{:}\ Aid, cid{:}\ Cid, is{:}\ Information\text{-}\mathbf{set}) \\ \mathbf{ext}\ \mathbf{wr}\ agents\ :\ Aid \xrightarrow{m} Agent \\ \quad \mathbf{rd}\ coals\ \ :\ Cid \xrightarrow{m} Coalition \\ \mathbf{pre}\ \{from, to\} \subseteq coals(cid).members \wedge is \subseteq agents(from) \wedge \\ \quad \mathbf{let}\ ar = coals(cid).auth\ \mathbf{in} \\ \quad \exists a \in \mathbf{dom}\ agents \cdot a\ ar\ from \wedge authorises(a, from, to, is) \\ \mathbf{post}\ agents = \overleftarrow{agents} \dagger \{to \mapsto \overleftarrow{agents}(to) \cup is\} \end{array}$$

where *authorises* is a predicate describing the criteria used to decide if the transfer of the particular information set *is* may proceed. In general, this expression may have to refer to additional state components to check, for example, clearance of the recipient *to*.

5.3 Changing the Authorisation Structure: Delegation

Including the authorisation structure in the coalition state opens up the possibility of modelling dynamic changes in the structure. The operations that permit this change might be performed (at least) every time a member joins or leaves a coalition, and perhaps at other times in response to changes of policy.

As an example, consider the delegation of authority by one agent to another. Let agent a_1 delegate its authority over agent a_3 to an agent a_2. This can be described by a function over the authorisation relation *auth* . The core part of the function definition is the change to *auth* by adding the new authority:

$$\begin{array}{l} Delegate\ (\ auth : AuthRel, a_1, a_2, a_3{:}\ Aid)\ auth{:}\ AuthRel \\ \mathbf{pre}\ a_1\ auth\ a_2 \wedge a_1\ auth\ a_3 \\ \mathbf{post}\ auth = \ auth \cup \{(a_2, a_3)\} \end{array}$$

In this particular example, a_1 retains his authority over a_3, but this need not always be the case. In modelling the function, we naturally think of its precondition and in this case we require the necessary initial authorities to be in place.

In defining functions and operations, we always have an eye to any relevant data type invariants. In VDM, there is an obligation on the writer of a function or operation to ensure that any invariants are respected. In this case, the risk is that the function may cause the invariant on *AuthRel* to be broken, e.g. by introducing a cycle where the *AuthRel* invariant forbids it. Discharging such a proof obligation is a core part of design using formal modelling.

The function as defined above works on a single authorisation relation. In our model in which these relations are defined on a per-coalition basis, the function would be "promoted" to work on the system state:

$$
\begin{array}{l}
\mathit{DelegateInCoal}\ (c\colon \mathit{Cid}, a_1, a_2, a_3\colon \mathit{Aid}) \\
\textbf{ext wr}\ \mathit{coals}\ :\ \mathit{Cid} \xrightarrow{m} \mathit{Coalition} \\
\textbf{pre}\ \{a_1, a_2, a_3\} \subseteq \mathit{coals}(c).\mathit{members} \wedge \\
\quad \mathit{pre\text{-}Delegate}(\mathit{coals}(c).\ \mathit{auth}\ , a_1, a_2, a_3) \\
\textbf{post let}\ \mathit{oc} = \overleftarrow{\mathit{coals}}(c)\ \textbf{in} \\
\quad \mathit{coals} = \overleftarrow{\mathit{coals}} \dagger \{c \mapsto \mu(\mathit{oc},\ \mathit{auth}\ \mapsto \mathit{Delegate}(\mathit{oc}.\ \mathit{auth}\ , a_1, a_2, a_3))\}
\end{array}
$$

Here the precondition is extended with additional consistency conditions that hold at the coalition level and the postcondition raises the results of the delegation function to the global state.

5.4 Extensions to the Authorisation Structure

It may be that an agent is permitted to give different authorisations to different agents, depending on their various skill sets, for example. In this case several authorisation structures would co-exist. The construction of a formal model such as that illustrated above provides an opportunity to explore the interactions of several such authorisation structures. A further refinement of this model explicitly records agents who are entitled to alter the authorisation structure.

We have presented in this section a straightforward way of representing authority to conduct operations within a coalition. This could be extended to authority over resources, and a similar set of relations could describe the roles that agents hold within a coalition.

5.5 The Dimension of Authorisation Structure

Unless security is an important factor in the coalition, the authority relation is unlikely to be implemented directly. It will most likely be an *emergent* relation, coming from, for example, the decision making and communications mechanisms within the coalition. We would argue that this relation, although it is emergent, is important to many understandings of virtual organisations. For example, it forms an important part of the basis of the taxonomies in [Let01] and [SZGM05]. An explicit description such as the one outlined above will allow him to check that any proposed specifications do in fact represent this relation.

Questions to consider include:

- Is authorisation a significant part of the dynamic coalition?
- Which (types of) actions will require authorisation?
- Will the authorisation structure vary across types of actions?
- Will the authorisation structure vary over time?
- Can the right to authorise actions be delegated?

6 Provenance

6.1 Modelling Provenance

In a dynamic coalition, it will often be important for an agent to know the source of the information it holds, as well as the information itself. In this section we consider some basic models of provenance (i.e. what meta-information is here teased out to record provenance — we could of course say that this is also "just" information). The simplest of these is for each agent to associate with each item of information the agent from whom it was received.

To simplify the presentation, we again assume that the knowledge base (*info*) within each agent is a set of information. The information record is altered, to associate a single providing agent with each information token.

This model would require some changes to the definitions in Section 4. Agents need only transfer the *token* part of an information record, and the receiving agent will add his own provenance component.

$$\begin{array}{lrcl} \Sigma_p :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{array}$$

$$\mathit{Agent} :: \mathit{agentinfo} : \mathit{TrackedInformation}\text{-}\mathbf{set}$$

$$\begin{array}{lrcl} \mathit{TrackedInformation} :: & \mathit{item} & : & \mathit{Information} \\ & \mathit{prov} & : & \mathit{Aid} \end{array}$$

If, when information is transferred, the provenance information is passed with it, then agents could build up a list of agents, representing the path that the information has taken to them. This immediately raises the possibility of an agent lying about the provenance information it passes on, but leaving that aside, the resulting model will be:

$$\begin{array}{lrcl} \Sigma_{p^*} :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{array}$$

$$\mathit{Agent} :: \mathit{agentinfo} : \mathit{TrackedInformation}\text{-}\mathbf{set}$$

$$\begin{array}{lrcl} \mathit{TrackedInformation} :: & \mathit{item} & : & \mathit{Information} \\ & \mathit{prov} & : & \mathit{Aid}^* \end{array}$$

We could include an invariant to stipulate that provenance information is passed on accurately and in full, if this was appropriate to the coalition. This would ensure that each agent's view of provenance was consistent. We would also have to take account of the situation where an agent receives a piece of information that it has previously passed on: whether or not the loop was removed from the provenance list would depend on whether the agent was interested in

recording the origin and path of the information, or in knowing as accurately as possible those agents that also know the information.

As well as recording the provenance of a piece of information, an agent might be interested in recording which other agents know the information. A suitable model for information would then be

$$\begin{array}{lrcl} \mathit{TrackedInformation} :: & \mathit{item} & : & \mathit{Information} \\ & \mathit{known\text{-}by} & : & \mathit{Aid}\text{-}\mathbf{set} \end{array}$$

If information is received, an agent can modify the 'known-by' field by recording the source of the message.

Coalitions may vary in the forms of communication by which information is transferred. Examples are agents disseminating information by unicast, multicast or broadcast. Agents can also take this into account. If something was received by broadcast, the recipient may then only assume that this is known by every member of the coalition. If by multicast (and the recipient set is known), the recipient set may be assumed to have received the information. These are really only weak conclusions as they refer to a single act of information transfer, and not to previous transfers.

It is not always appropriate or possible for individual agents to retain this kind of information. Some dynamic coalitions provide an *archival* service (see for example the extended case study in Section 9.) This could be modelled as an agent whose role is to record this kind of information for every transaction within the dynamic coalition. This can be used, for example, if non-repudiation of messages is important.

If we allow coalitions to communicate with each other as coalitions, then we need to include a similar provenance recording mechanism at the coalition level.

An analogy here is with briefings given under different terms by politicians. An "on the record" briefing will typically contain less sensitive information than an "off the record" one. Note that, in this example, terms also govern provenance meta-data. Information gathered at an "on the record" briefing is passed on with its provenance attached, while "off the record" briefings contain wild card provenances, e.g. "sources close to the Minister".

6.2 The Dimension of Provenance

When thinking about provenance, a coalition designer might ask

- What provenance information will be recorded?
- How will it be acquired?
- How detailed will the information be?
- Will agents record their own provenance information?
- Will provenance information be recorded centrally?
- If there are differing records of provenance, what efforts will be made to keep them consistent?

7 Time

Within different forms of dynamic coalitions, significant events may take place at vastly differing rates. For example, a commercial virtual enterprise may operate in terms of years, months, weeks and days. In a hospital the seconds or minutes taken to get hold of the right doctor may be of significance, and in an automated intensive care ward the milliseconds taken to communicate between different machines may be crucial.

Consistent with this view of timed behaviour is the notion of *time bands*, initially developed by Newell within the context of human cognition work [New90]. This approach has been applied to complex socio-technical systems in [BHBF05], on which we base much of the following presentation. Time is not seen as a single flat dimension, but is represented as a series of *time bands*. These are characterised by their *granularity* and *precision*. System activities are placed within a particular time band if they engage in significant events within the time scale represented by the time band. For example, one band may capture operating system level activity, and another may capture social activity (phone calls, etc.) related to the same system.

The specification of a system requires the definition of these bands (usually separated by approximately a factor of 10). Within a band, *activities* have duration while *events* are instantaneous. Instantaneous events in one band may map into durational activities in a lower one. A description of the timely behaviour of a dynamic coalition will therefore (we suggest) need to specify the time bands relevant to the dynamic coalition, and identify all the significant events and activities at each time band, as well as their relation to linked events and activities in other bands.

Time bands can also be used to provide a way of recording the history of the behaviour of a coalition. This record would be external to the model.

In the example specifications below, we assume that a single agent operates in a single time band, but may communicate with agents in any other time band.

As in previous dimensions, we have three possible distinct choices for enriching our model with meta-information about time — we can place knowledge of time at the global, coalition or agent level.

The first choice is shown in the $\Sigma_{time\text{-}g}$ model, and in the $\Sigma_{time\text{-}a}$ model each agent has its own clock. We do not present the model of a coalition with a common clock. In each model we put no constraints on the type *Time*. (Later, the operations we define assume a $\geq$ operator.)

$$\begin{array}{lrcl} \Sigma_{time\text{-}g} :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent} \\ & \mathit{current\text{-}time} & : & \mathit{Time} \end{array}$$

In the $\Sigma_{time\text{-}a}$ model, each agent has its own clock.

$$\begin{array}{lrcl} \Sigma_{time\text{-}a} :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{array}$$

$$\begin{array}{lrcl} \mathit{Agent} :: & \mathit{current\text{-}time} & : & \mathit{Time} \\ & \ldots & : & \ldots \end{array}$$

It is possible to include awareness of time at different levels in the same model. This would allow some agents to have access to the global clock and others to

be restricted to their own clock.

The dimensions we have described so far (membership, knowledge, etc.) are orthogonal to each other. Each may exist and describe meaningful situations without the presence of the others. This is not true for the time dimension. In isolation, it describes little of worth or relevance, but in combination with the other dimensions it enriches them and provides valuable new insights.

We have already introduced the possibility of meta-information. In the next two sections we allow this meta-information to record time-values. We demonstrate this in the rest of this section by combining time with two example domains: *information* in Section 3 and *provenance* in Section 6.

7.1 Time-valued Information

The simplest way to introduce a time component to meta-information is to associate a single time value with each information token. This value would record a significant instant for that information. For example, it could record the time at which an agent learned a piece of information.

7.1.1 Expiry Times

In this section, we consider the case where information expires at a certain point in time. In the simple agent model below, a single time is associated with each piece of information. At the recorded time, the value of the information changes in some quantifiable way: for example, a document may move from "classified" to "unclassified", or meteorological data may change from "current" to "out-of-date".

$$\begin{array}{rrcl} \mathit{Agent} :: & \mathit{agentinfo} & : & \mathit{Information}\text{-}\mathbf{set} \\ & \mathit{current\text{-}time} & : & \mathit{Time} \end{array}$$

$$\begin{array}{rrcl} \mathit{Information} :: & \mathit{item} & : & \mathit{token} \\ & \mathit{expire} & : & [\mathit{Time}] \end{array}$$

In this model, information items may have a single "expiry date". If this is not present, we assume that information is valid indefinitely. Using *expire*, an agent can check if a piece of information is still valid at a particular point in time.

$$\mathit{still\text{-}valid} : \mathit{Information} \times \mathit{Time} \to \mathbb{B}$$

$$\mathit{still\text{-}valid}(\mathit{info}, \mathit{time}) \quad \triangleq \quad \mathit{info}.\mathit{expire} = \mathbf{nil} \lor \mathit{info}.\mathit{expire} \geq \mathit{time}$$

It is possible that a set of information may remain valuable for longer than any of its elements. This could happen if two pieces of information were about the same external thing – for example location and strength of armed forces. To allow for this, we must allow the agent to group the information that he has.

$$\begin{array}{rrcl} \mathit{Agent} :: & \mathit{agentinfo} & : & \mathit{TimedInfo}\text{-}\mathbf{set} \\ & \mathit{current\text{-}time} & : & \mathit{Time} \end{array}$$

$$\begin{array}{rrcl} \mathit{TimedInfo} :: & \mathit{info} & : & \mathit{Information}\text{-}\mathbf{set} \\ & \mathit{set\text{-}expire} & : & \big[\mathit{Time}\big] \end{array}$$

$$\mathbf{inv}\ (\mathit{info}, \mathit{set\text{-}expire}) \triangleq \mathit{set\text{-}expire} \neq \mathbf{nil} \ \Leftrightarrow\ \forall i \in \mathit{info} \cdot i.\mathit{expire} \neq \mathbf{nil} \land i.\mathit{expire} \leq \mathit{set\text{-}expire}$$

The invariant states that the expiry of a set of information must be later than the expiry of any of its elements.

It is straightforward to define an operation to determine the expiry of a set *info* not explicitly grouped into a *TimedInfo*, simply by taking the maximum expiry time of its elements.

7.1.2 Degrading the Value of Information Over Time

Rather than information being either "expired" or "not expired" with respect to a certain time, we could model information degrading more slowly over time. For example, if the security classification of information was relevant, agents could categorise all their information according to some *security-type*, as in the model below. Here, agents record the first time at which a certain security level applies, and the assumption (captured by the invariant) is that the security level decreases monotonically with time. For ease of presentation, we will assume a single *KnowledgeBase* for each agent.

$$
\begin{array}{lrcl}
\mathit{Agent} :: & \mathit{security\text{-}level} & : & \textsc{High} \mid \textsc{Medium} \mid \textsc{Low} \mid \textsc{None} \\
& \mathit{KnowledgeBase} & : & \mathit{TimedInformation}\text{-}\mathbf{set} \\
& \mathit{current\text{-}time} & : & \mathit{Time}
\end{array}
$$

$$
\begin{array}{lrcl}
\mathit{TimedInformation} :: & \mathit{item} & : & \mathit{Information} \\
& \mathit{security} & : & \mathit{security\text{-}level} \xrightarrow{m} \mathit{Time}
\end{array}
$$

7.2 Time and Provenance

In Section 6, we presented two models of information provenance, Σ_p and Σ_{p^*}. Here we enrich both of these models by adding time. In the first model the extension is straightforward: an agent merely records the time at which each item of information was received, as well as the provider.

$$
\begin{array}{lrcl}
\Sigma_{p\text{-}time} :: & \mathit{coals} & : & \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\
& \mathit{agents} & : & \mathit{Aid} \xrightarrow{m} \mathit{Agent}
\end{array}
$$

$$
\begin{array}{lrcl}
\mathit{Agent} :: & \mathit{KnowledgeBase} & : & \mathit{TimedInformation}\text{-}\mathbf{set} \\
& \mathit{current\text{-}time} & : & \mathit{Time}
\end{array}
$$

$$
\begin{array}{lrcl}
\mathit{TimedInformation} :: & \mathit{item} & : & \mathit{Information} \\
& \mathit{prov} & : & \mathit{Aid} \\
& \mathit{time} & : & [\mathit{Time}]
\end{array}
$$

In this model, an agent records the source of a *item* in *prov* (here just the name of the communicating agent), and the time it was communicated. This would be enough to establish some measure of the trustworthiness of the *item*, but no attempt is made to keep a full audit trail.

In the second model, where an agent passes on its own provenance information with a *item*, the information component in the model becomes:

$$
\begin{array}{lrcl}
\mathit{TimedInformation} :: & \mathit{item} & : & \mathit{Information} \\
& \mathit{prov} & : & (\mathit{Aid} \times \mathit{Time})^*
\end{array}
$$

$$
\mathbf{inv}\ (\_, \mathit{prov}) \triangleq \\
\quad \forall i < \mathit{len}(\mathit{prov}) \cdot \mathit{snd}(\mathit{prov}(i+1)) \geq \mathit{snd}(\mathit{prov}(i))
$$

The invariant says that *time* is non-decreasing as we move towards the head of any *prov* sequence[8].

7.3 The Dimension of Time

Time is a dimension which enriches and sheds new light on other dimensions, rather than one that is valuable of itself. A coalition designer may ask

- What other dimensions have some time component?
- How do they use time?
- Which components may vary with time?
- What access do agents and coalitions have to clocks?

8 Trust

8.1 Trust-related Meta-information

The problem of how to infer trust from meta-information is still open, and solutions are necessarily context dependent. After trust has been computed the question of how to use it has many answers, again context dependent. We therefore do not consider these questions, and begin by assuming that trust values have been obtained. These trust values may represent an agent's trust in other agents, or an agent's trust in information.

An agent's trust in other agents can be represented as

$$\begin{array}{lll} \Sigma_{trust\text{-}a} :: & coals & : Cid \xrightarrow{m} Aid\text{-}\mathbf{set} \\ & agents & : Aid \xrightarrow{m} Agent \end{array}$$

$$\mathbf{inv}\ (coals, agents) \triangleq \bigcup \mathbf{rng}\ coals \subseteq \mathbf{dom}\ agents$$

$$Agent :: aTrust : Aid \xrightarrow{m} trustvalue$$

where

$$trustvalue = \mathbb{R}$$

Over time, the *aTrust* mapping will be updated, according to some set of rules that an agent has. For example, if an agent a applies a "Friend-of-a-friend" rule, and agent b (whom a trusts) itself trusts agent c, then agent a may be inclined to trust agent c as well.

Coalition members might be expected to trust the other members of the coalition to some degree. If this were the case, it could be mandated by a suitable invariant:

$$\begin{array}{lll} \Sigma_{trust\text{-}c} :: & coals & : Cid \xrightarrow{m} Coalition \\ & agents & : Aid \xrightarrow{m} Agent \end{array}$$

$$\begin{array}{l} \mathbf{inv}\ (coals, \text{-}) \triangleq \\ \quad \forall c \in \mathbf{dom}\ coals \cdot \\ \quad \forall m, m' \in coals(c).members \cdot\ m.aTrust(m') \geq coals(c).cTrust \end{array}$$

[8] The *snd* operator gets the second element of a pair.

$$\begin{array}{rcl} \mathit{Coalition} & :: & \mathit{members} \ : \ \mathit{Aid}\text{-}\mathbf{set} \\ & & \mathit{cTrust} \ : \ \mathit{trustvalue} \end{array}$$

Trust in information is most simply recorded as a meta-information component within *Information*.

$$\begin{array}{rcl} \Sigma_{\mathit{trust\text{-}info}} & :: & \mathit{coals} \ : \ \mathit{Cid} \xrightarrow{m} \mathit{Aid}\text{-}\mathbf{set} \\ & & \mathit{agents} \ : \ \mathit{Aid} \xrightarrow{m} \mathit{Agent} \end{array}$$

$$\mathit{Agent} \ :: \ \mathit{KnowledgeBase} \ : \ \mathit{Information}\text{-}\mathbf{set}$$

$$\begin{array}{rcl} \mathit{Information} & :: & \mathit{item} \ : \ \mathit{token} \\ & & \mathit{iTrust} \ : \ \mathit{trustvalue} \end{array}$$

8.2 The Dimension of Trust

Many models for deciding trust values and acting on them may be found in the literature. We have not attempted to characterise these, but rather considered how agents may record trust values, and what these values might relate to.

Following on from this, some questions to consider are:

- Which elements of will agents need to have trust in?
- How will they record these values?
- How will trust values influence an agents behaviour?
- How will trust values vary with the behaviour of agents?

9 Case Study: The GOLD VO Architecture

In this section we will use as an example the Virtual Organisation architecture under development within the GOLD project [GOL, CCH+05]. We show how the architectural choices made in [CCH+05] can be positioned within the space of dynamic coalitions outlined in the paper.

The GOLD project is seeking to build a software architecture that will support the formation, operation and termination of a number of business coalitions, within the high-value chemicals industry. The production of a chemical involves a number of stages, including initial laboratory experiments, building and running of industrial scale plant, safety analysis of any by-products. A coalition of companies often forms around the production of a particular chemical, since very few companies have the resources to see one chemical right through from inception to marketing. These coalitions are loosely bound together with members joining as necessary and leaving when their part of the process is complete.

In [CCH+05], an environment is described where a number of overlapping Virtual Organisations can be formed. We will call this environment the *VO breeding ground*. In a sense, this breeding ground forms a single global coalition, whose purpose is to form smaller and more constrained VOs around particular development processes. A single company may be in many such coalitions.

The GOLD architecture is service-based, and allows companies to communicate information, transfer documents and access each others resources. The intention is that each company in the VO breeding ground will offer a standard

set of services in a uniform way which will allow companies to form and operate these coalitions at a much greater rate than is currently possible.

We take each relevant dimension in turn and show where the VOs to be supported by the GOLD architecture fit along these dimensions.

Our purpose is not to provide a full VDM specification of the architecture, but rather to demonstrate that early use of formality in VO design (as in any design) can help identify important decisions quickly.

9.1 Coalition Membership

In [CCH+05], coalition membership is realised by a *membership management function* (MMF) which contains a list of the members of each VO in the VO breeding ground, and mechanisms for joining and leaving coalitions.

The MMF also contains role descriptions for individuals and organisations within each VO. A possible way of organising this is shown below. The *Agents* are the companies involved. The *Uid* are individual users of GOLD within companies. The mappings *aroles* and *uroles* map agents and users to *agent* (*aRole*) and *user* (*uRole*) roles.

$$aRole = \text{LEADER} \mid token$$

$$uRole = \text{CHEMIST} \mid token$$

$$\begin{array}{lrcl} \Sigma_{gold} :: & coals & : & Cid \xrightarrow{m} Coalition \\ & agents & : & Aid \xrightarrow{m} Agent \\ & users & : & Uid \xrightarrow{m} User \end{array}$$

$$\begin{array}{lrcl} Coalition :: & members & : & Aid\text{-}\mathbf{set} \\ & aroles & : & Aid \xrightarrow{m} aRole\text{-}\mathbf{set} \\ & uroles & : & Uid \xrightarrow{m} uRole\text{-}\mathbf{set} \end{array}$$

$$\mathbf{inv}\ (\text{-}, aroles, \text{-}) \triangleq \exists!\, aid \in \mathbf{dom}\, aroles \cdot \text{LEADER} \in aroles(aid)$$

$$Agent :: employees : Uid\text{-}\mathbf{set}$$

A distinction is made here between the companies (*Agents*) and the company employees (*Users*). As an aside, it would be possible to model companies as coalitions of users. However our task here is to tailor a model to describe the functionality provided by the GOLD platform, which explicitly recognises both companies and agents as first class objects. It is clearly necessary that the information about a user include the company for which he or she works. We must also ask here ask the question "Must a user in a coalition be an employee of a member company?" This is not stipulated by [CCH+05], but if in a particular VO the answer is yes, it can be enforced by an invariant.

Every GOLD coalition will be initiated by a single leader, and this is enforced by the invariant. Membership of the coalition will be at the discretion of this leader, but companies will be able to leave unilaterally. The *Join* operation will therefore be (where b is authorising a to join:)

Join $(a, b\colon Aid, c\colon Cid)$

ext wr $coals \;:\; Cid \xrightarrow{m} Coalition$
rd $agents \;:\; Aid \xrightarrow{m} Agent$

pre $a \in \mathbf{dom}\, agents \wedge c \in \mathbf{dom}\, coals \wedge a \notin coals(c).members \wedge$
$\textsc{leader} \in coals(c).aroles(b) \wedge authorise\text{-}join(b, a, c)$

post $coals(c).members = \overleftarrow{coals}(c).members \cup \{a\}$

Join does not update the *aroles* or *uroles* components of the *Coalition*. We envisage separate operations to do this, but do not present them here.

Remove $(a\colon Aid, c\colon Cid)$

ext wr $coals \;:\; Cid \xrightarrow{m} Coalition$
rd $agents \;:\; Aid \xrightarrow{m} Agent$

pre $c \in \mathbf{dom}\, coals \wedge a \in coals(c).members$

post $coals =$
$\overleftarrow{coals} \dagger \{c \mapsto \mu(\overleftarrow{coals}(c),$
$members \mapsto \overleftarrow{coals}(c).members \setminus \{a\},$
$aroles \mapsto \{a\} \lhd \overleftarrow{coals}(c).aroles,$
$uroles \mapsto \{u \mid u \in agents(a).employees\} \lhd$
$\overleftarrow{coals}(c).uroles)\}$

With *Remove* we explicitly remove a departing agent from the *aroles* map, and remove all their employees from the *uroles* map. This is a basic security precaution. However, it raises the question "Can an employee be a member of two companies (perhaps through a consultancy arrangement?)" If this is the case, we must be careful not to remove him just because one of his employers leaves.

9.2 Information

Within GOLD, a *document* "is the fundamental unit of information exchanged between VO members" [CCH+05]. The VO breeding ground contains certain *service discovery functions* as architectural elements, which are external to any VO or company. These include a *Document Type Registry (DTR)*, which defines a common understanding of the classes of document which may be exchanged between companies.

This is very close to our basic centralised database Σ_{ig} in Section 3, with a pre-defined set of document types. To model the *DTR* we would include the following in the global state component of the model.

$\Sigma_{gold} \;::\; \ldots : \ldots$
$doc\text{-}type \;:\; \textsc{Expr-rep} \mid \textsc{Mngt-rep} \mid \textsc{Safety-rep} \mid \textsc{Memo} \mid token$

Describing the different document types as tokens will be sufficient for us here. In a more advanced stage of the specification process this would justify a more detailed description.

The GOLD architecture allows for both central and local storage of information in *Document Repositories (dr)* within a single VO. A VO may have a number of central storage facilities. Information placed in these storage facilities

may be retrieved using an index or identifier. Each repository, whether local to a company or local to a VO, will be a map from document identifiers to elements of type *doc-type*. We will assume that each agent has a single document repository. It also maintains a mapping from the coalitions to which it belongs to the documents that may be shared with that coalition. There is a single repository at the coalition level for each VO. Thus

$$\begin{aligned} \mathit{Coalition} :: & \ \ldots : \ldots \\ & \mathit{dr} : \mathit{Did} \xrightarrow{m} \mathit{Document} \end{aligned}$$

$$\begin{aligned} \mathit{Agent} :: & \ \ldots : \ldots \\ & \mathit{dr} : \mathit{Did} \xrightarrow{m} \mathit{Document} \\ & \mathit{coalinfo} : \mathit{Cid} \xrightarrow{m} \mathit{Did}\text{-}\mathbf{set} \end{aligned}$$

These repositories will have the ability to pro-actively update users of relevant changes to documents in their repository as well as allow them to interrogate their contents. A collection of simple (omitted) operations will allow agents to update and interrogate these repositories and allow the repositories to update users of changes.

9.3 Information Transfer

An assumption in [CCH$^+$05] is that information (documents) should only be transmitted *within* VOs and not across VO boundaries. All transfers are subject to a unified security infrastructure. Of course, there is nothing to stop two companies in separate VOs communicating using means other than the GOLD architecture.

The operation below transfers a set of documents from one agent to another. The behaviour of the security architecture is again captured as a predicate *authorises*. A transfer is allowed only if it is authorised by an appropriate member of the coalition.

It is important that we identify the particular coalition within which the transfer is taking place. This is because many coalitions may exist at any one time and confidential information relevant to one coalition may easily be passed under the auspices of another.

$\mathit{InfoTransfer}\ (\mathit{from}, \mathit{to}\colon \mathit{Aid}, d\colon \mathit{Document}, \mathit{did}\colon \mathit{Did}, c\colon \mathit{Cid})$

ext wr $\mathit{agents} : \mathit{Aid} \xrightarrow{m} \mathit{Agent}$

pre $\{\mathit{from}, \mathit{to}\} \subset \mathbf{dom}\ \mathit{agents} \land d \in \mathbf{rng}\ \mathit{agents}(\mathit{from}).\mathit{dr} \land$
$\mathit{did} \notin \mathbf{dom}\ \mathit{coals}(c).\mathit{dr} \land$
$c \in \mathbf{dom}\ \mathit{coals} \land \{\mathit{from}, \mathit{to}\} \subset \mathit{coals}(c).\mathit{members} \land$
$\exists a \in \{\mathbf{dom}\ \mathit{agents}\} \cdot \mathit{authorises}(a, \mathit{from}, \mathit{to}, \{d\}, c)$

post $\mathit{agents} = \overleftarrow{\mathit{agents}} \dagger \{\mathit{to} \mapsto \mu(\overleftarrow{\mathit{agents}}(\mathit{to}), \mathit{dr} \mapsto \overleftarrow{\mathit{agents}}(\mathit{to}).\mathit{dr} \cup \{\mathit{did} \mapsto d\})\}$

Note that the predicate *authorises* need not require the intervention of an authorising agent for every information transfer: permission to distribute documents may be granted in advance and stored until referenced by the *authorises* predicate.

If an agent is a member of a coalition, it can add to the coalition level repository and read documents in this repository provided that the coalition-specific access control rules are satisfied.

The operation below adds a single document to a coalition document repository.

$AddToVO$ ($from$: Aid, d: $Document$, did: Did, c: Cid)

ext wr $coals$: $Cid \xrightarrow{m} Coalition$

pre $from \in \mathbf{dom}\ agents \land from \in coals(c).members \land$
$d \in \mathbf{rng}\ agents(from).dr \land$
$did \notin \mathbf{dom}\ coals(c).dr \land$
$c \in \mathbf{dom}\ coals \land$
$\exists a \in \mathbf{dom}\ agents \cdot a \in coals(c).members \land permits(a, from, \{d\}, c)$

post $coals = \overleftarrow{coals} \dagger \{c \mapsto \mu(\overleftarrow{coals}(c), dr \mapsto \overleftarrow{coals}(c).dr \cup \{did \mapsto d\})\}$

9.4 Authorisation Structure

In [CCH+05], it is assumed that there will be a single instigator of a virtual organisation who will subcontract part of the work to others. Thus the authorisation structure of a GOLD coalition will naturally be a star formation. We model this as

$AuthRel = (Aid \times Aid)$-**set**

where

inv($auth$) $\triangleq$
$\forall as \subseteq \mathbf{dom}\ auth \cdot as \neq \{\} \Rightarrow \exists a \in as \cdot \neg(\exists a' \in as \cdot a\ auth\ a')$

and

card dom $auth = 1$

A subcontractor may choose to further subcontract their work. This would be modelled as a separate VO.

9.5 Provenance

The GOLD architecture will provide a secure *archival* service for future audit, available only to authorised members. This will need to provide a full record of the workings of the VO, including business agreements, contracts, service level agreements, and creation and communication of documents. In the example in Section 6, we explored the case where each coalition member recorded the provenance of the information that they store. The GOLD architecture requires separate archival services for individual VOs. Little is said in detail about these. To model them, we could simply create a document repository at the coalition level.

$Archive$ :: doc : $Did \xrightarrow{m} DocInfo$

$DocInfo$ :: $document$: $doc\text{-}type$
$author$: $Uid \mid Aid$
$predecessor$: $[Did]$
$successor$: $[Did]$
$date\text{-}of\text{-}creation$: $Time$
$transfers$: $(Aid \times Aid \times Aid \times Time)$-**set**

We allow the author to be an individual user or a company. Changes to a document are managed by recording all published versions, together with the predecessor and successor of a document (both optionally null). Under the *transfers* field, we record the transfer *from* the first *Aid to* the second *Aid*, authorised by the third *Aid*, as well as the time at which it was authorised.

The *InfoTransfer* operation definition would need to be expanded to include the automatic updating of the archive facility, but we do not do that here.

9.6 Trust and Perception

Agents may have no relevant information, history or context available when they begin to work together. GOLD attempts to provide a trustworthy medium for running VOs thus making trust an architectural issue [Per05].

The GOLD architecture provides mechanisms (authorisation, authentication, non-repudiation, etc.) which are seen as valuable in promoting trust between companies and trust in the integrity of the interactions between companies. As such, the GOLD architecture does not impose a particular model of trust. Companies will be free to trust (or mis-trust) information, communications and other companies as they choose.

It is possible that the information in different Document Repositories will conflict, and that different companies will therefore have different perceptions of this information. However these would probably be resolved by communication with the other companies (since this is obviously a co-operative environment), rather than by running an inference function.

10 Conclusions, Related and Future Work

We have used a formal model-oriented specification language as the basis for a systematic exploration of the space of dynamic coalitions. The modelling language's emphasis on abstractions of data, state and operations has encouraged a focus on the "meta-information" that characterises the structure and information flows of a coalition. As a result, several dimensions have been identified along which dynamic coalition structures may vary. We have attempted to place one real virtual organisation scheme, that of the GOLD project, in the space spanned by the dimensions that we have identified.

10.1 Future Work

Future work can be envisaged in a number of areas. Two of these areas overlap: development of validation techniques for dynamic coalitions exploiting the formality of the models and development of knowledge/guidance about the design of dynamic coalitions based on the dimensions mapped out in the modelling work done so far. We also anticipate modelling the Domain Based Security approach and examining how this combines with dynamically forming coalitions. The question of responsibility for coalition membership has already been explored in Section 2. We anticipate broadening this exploration to consider the area of responsibility for actions within coalitions.

Validation of Dynamic Coalition Models: The validation of a model of a dynamic coalition involves checking internal consistency and assessing emergent properties of the model. The consistency checks are described in VDM as *proof obligations* – logical conjectures that must be discharged in order for a model to be regarded as internally consistent. They include basic properties such as ensuring that preconditions are total as well as more sophisticated constraints such as ensuring that operations respect invariants on the state. Under certain conditions, proof obligations can be discharged automatically. However, in general, they may require human-guided proof. Analysis of emergent behaviour is often done by proposing *validation conjectures*. These are, again, formal statements of desired properties of a model and may require human-guided proof. A validation conjecture, might, for example, state that no more than two agents in a coalition know a particular high-value fact at a particular time. Experience suggests that formulation of relevant validation conjectures requires some thought.

Discharging proof obligations and assessing validation conjectures can be done at various levels of confidence. An executable formal model can, for example, be tested. At the other extreme, a human-guided but machine-assisted proof can be constructed. For modes of dynamic coalitions, one may envisage several useful validation activities:

- An executable model linked to a suitable interface through an application programmer interface (this is a well-established technique for basic validation of models in VDM [AS99, FL98, FLM+05]). The interface permits ad hoc exploration of the model by a user. With features such as invariant and precondition checking, this permits a systematic analysis of normative as well as some failure behaviours.

- Building on an executable model, scenarios can be defined and executed. A scenario is a script involving invocations of the model's operations (adding or removing members, transferring specific information etc.).

- Scripts represent paths through state transition models. Is it possible to generate a state-transition machine (Kripke structure) for an abstraction of a given DC model? If so then it may be possible to search systematically for states having specific properties.

Designing Dynamic Coalitions: Are dynamic coalitions designed or do they emerge? The challenge in building systems to support dynamic coalitions lies in providing just sufficient structure to permit validation of emergent properties without over-constraining heterogeneity and flexibility. The dimensions identified in the modelling work described here may be useful in guiding those building systems that support dynamic coalitions. Possible directions for further work include:

- Exploring/validating the dimensions by applying them to a wider range of known DC structures. Can we identify new structures not considered before? Can we describe the result of two coalitions merging together?

- Developing proof obligations for DCs, or (weaker but more practical), a checklist for DC developers.

- Investigating the description of structures that are superimposed on the DC itself, such as access control, and investigate policy languages for describing access control on the basis of metadata.

Domain Based Security: Domain Based Security [Hug02, HW05, War03] is an approach to information security that focuses on the way information is shared. Developing the domain based security model for an organisation involves two parts, an Infosec business model and an Infosec architecture model. The business model is built first by identifying *domains* within the organisation, where a domain represents a group of people and the information they use. Information is assumed to flow freely within a domain, and domain based security is concerned with the information flow between domains. The Infosec business model also describes the physical places where people work, and a connection between a place and a domain exists when people in that physical space are able to work in the domain. The *types* of communication channels between domains (email, web, etc.) are also explicitly represented.

Infrastructure constraints are modelled using the Infosec Infrastructure model. This involves drawing infrastructure *islands*, each of which may be geographically distributed, and are only connected using clearly identified points of connection. The challenging and illuminating task will be to integrate this model with our existing models of dynamic coalitions, in order to examine and predict how dynamic coalitions might function in a Domain Based Security environment. We anticipate that this strand of work will benefit greatly from further progress on designing and validating dynamic coalitions.

Responsibility: Section 2 explores the question: *where does responsibility lie for coalition membership?* We want to broaden this question to include all coalition actions. We want as well to have some way of "keeping records" of coalition behaviour, so that these questions can be asked retrospectively. It may be sufficient to extend meta-information to include an "authorised-by" component, or we may need to keep records external to the operation of the VO.

10.2 Related Work

The VO literature contains some attempts at taxonomies of the subject, many from the view of management science. One example is [Let01], in which Virtual Organisations are categorised according to the informational flow between each other and to customers. Thus, for example, a *virtual face* organisation (one which presents a single front to a customer) is distinguished from a *co-alliance* (one where each organisation may deal directly with the customer.) Considering the information flows within the VO, a *star alliance* consists of subordinates each talking only to a lead partner, and a *value-alliance* consists of members organised in a ring, each adding value to the product. This taxonomy is from the point of view of management science, and therefore excludes consideration of the full range of dynamic coalitions that we have tried to capture in this paper.

Acknowledgments

We are grateful to Tom McCutcheon and Ramsay Taylor of the UK Defence Science and Technology Laboratory for their encouragement to examine information flow in dynamic coalition structures. We acknowledge much helpful input from colleagues in the Interdisciplinary Collaboration on Dependability of Computer-based Systems (DIRC), especially Peter Ryan, Michael Harrison and Fred Schneider. This research was supported by DSTL and the EPSRC project GOLD.

References

[Abr96] J.-R. Abrial. *The B-Book: Assigning programs to meanings*. Cambridge University Press, 1996.

[And96] D.J. Andrews, editor. *Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language*. International Organization for Standardization, December 1996. International Standard ISO/IEC 13817-1.

[AS99] Sten Agerholm and Wendy Schafer. Analyzing SAFER using UML and VDM++. In John Fitzgerald and Peter Gorm Larsen, editors, *VDM in Practice*, pages 139–141, September 1999.

[BHBF05] A Burns, I.J. Hayes, G. Baxter, and C.J. Fidge. Modelling temporal behaviour in complex socio-technical systems. Technical Report 390, University of York, 2005.

[CCH+05] Adrian Conlin, Nick Cook, Hugo Hilden, Panos Periorellis, and Rob Smith. GOLD Architecture Document. Technical Report CS-TR-923, University of Newcastle, 2005.

[CHTCB96] Tushar Deepak Chandra, Vassos Hadzilacos, Sam Toueg, and Bernadette Charron-Bost. On the impossibility of group membership. Technical Report 2782, INRIA Rocquencourt, January 1996.

[Cri91] F. Cristian. Reaching agreement on processor group membership in synchronous distributed systems. *Distributed Systems*, 4(4):175–187, 1991.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*. MIT press, 1995.

[FL98] John Fitzgerald and Peter Gorm Larsen. *Modelling systems: practical tools and techniques in software development*. Cambridge University Press, 1998.

[FLM+05] John Fitzgerald, Peter Gorm Larsen, Paul Mukherjee, Nico Plat, and Marcel Verhoef. *Validated Designs for Object-oriented Systems*. Springer Verlag, London, 2005. ISBN 1-85233-881-4.

[GA04] C. Gacek and B. Arief. The many meanings of open source. *IEEE Software*, 21(1):34–40, January/February 2004.

[GOL] The GOLD Project. `http://www.goldproject.ac.uk/`.

[Hay93] Ian Hayes, editor. *Specification Case Studies*. Prentice Hall International, second edition, 1993.

[Hug02] K. J. Hughes. Domain Based Security: enabling security at the level of applications and business processes. White paper, QinetiQ, 2002.

[HW05] Kay Hughes and Simon Wiseman. Analysis of information security risks: Policy for protection through to implementation. In *4th European Conference on Information Warfare and Security*, July 2005.

[Jon90] C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall International, second edition, 1990. ISBN 0-13-880733-7.

[Let01] N. Lethbridge. An I-based Taxonomy of Virtual Organisations and the Implications for Effective Management. *Informing Science*, 4(1):17–24, 2001.

[Mod] The VDM Dynamic Coalition Models. `http://www.dirc.org.uk/resources/dc.html`.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.

[New90] A. Newell. *Unified Theories of Cognition*. Harvard, 1990.

[Per05] Panos Periorellis. Trust Position - GOLD. Technical Report CS-TR-908, University of Newcastle, 2005.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[SZGM05] N. Sanchez, D. Zubiaga, J. González, and A Molina. Virtual Breeding Environment: A First Approach to Understanding Working and Sharing Principles. In *Proceedings of InterOp-ESA'05*, 2005.

[War03] Karl Warrener. Facilitating Risk Balance - An Architectural Approach. In *15th Annual Canadian Infomration Technology Security Symposium*, May 2003.

[Wel84] Ann Welsh. *A Database Programming Language: Definition, Implementation and Correctness Proofs*. PhD thesis, University of Manchester, 1984.

A Operations for Model Σ_m

NewCoal (c: *Cid*, a: *Aid*)
ext wr *coals* : *Cid* $\xrightarrow{m}$ *Aid*-**set**
pre $c \notin$ **dom** *coals*
post $coals = \overleftarrow{coals} \cup \{c \mapsto \{a\}\}$

Join (a: *Aid*, c: *Cid*)
ext wr *coals* : *Cid* $\xrightarrow{m}$ *Aid*-**set**
rd *agents* : *Aid* $\xrightarrow{m}$ *Agent*
pre $a \in$ **dom** *agents* $\wedge\ c \in$ **dom** *coals*$(\wedge a \notin coals(c))$
post $coals = \overleftarrow{coals} \dagger \{c \mapsto \overleftarrow{coals}(c) \cup \{a\}\}$

Remove (a: *Aid*, c: *Cid*)
ext wr *coals* : *Cid* $\xrightarrow{m}$ *Aid*-**set**
pre $c \in$ **dom** *coals* $\wedge\ a \in coals(c)$
post if $coals(c) = \{a\}$
then $coals = \{c\} \lhd \overleftarrow{coals}$
else $\overleftarrow{coals} \dagger \{c \mapsto \overleftarrow{coals}(c) \setminus \{a\}\}$

It may be possible to dissolve a coalition:

DissolveCoal (c: *Cid*)
ext wr *coals* : *Cid* $\xrightarrow{m}$ *Aid*-**set**
pre $c \in$ **dom** *coals*
post $coals = \{c\} \lhd \overleftarrow{coals}$

B Model Σ_{auth}

Coalition :: *members* : *Aid*-**set**
threshold : $\mathbb{R}$
inv (-, *threshold*) $\triangleq 0 \leq threshold \wedge threshold \leq 1$

Σ_{auth} :: *coals* : *Cid* $\xrightarrow{m}$ *Coalition*
agents : *Aid* $\xrightarrow{m}$ *Agent*
inv $(coals, agents) \triangleq \bigcup\{c.members \mid c \in$ **rng** $coals\} \subseteq$ **dom** *agents*

NewCoal (c: *Cid*, a: *Aid*, t: $\mathbb{R}$)
ext wr *coals* : *Cid* $\xrightarrow{m}$ *Coalition*
pre $c \notin$ **dom** *coals* $\wedge\ 0 \leq threshold \wedge threshold \leq 1$
post $coals = \overleftarrow{coals} \cup \{c \mapsto mk\text{-}Coalition(a, t)\}$

Join (a: Aid, c: Cid, $supp$: Aid-**set**)

ext wr $coals \;:\; Cid \xrightarrow{m} Coalition$
rd $agents \;:\; Aid \xrightarrow{m} Agent$

pre $a \in \mathbf{dom}\, agents \wedge c \in \mathbf{dom}\, coals \wedge a \notin coals(c) \wedge$
$mandated(coals(c), supp)$

post $coals = \overleftarrow{coals} \dagger \{c \mapsto \mu(\overleftarrow{coals}(c), members \mapsto \overleftarrow{coals}(c).members \cup \{a\})\}$

Remove (a: Aid, c: Cid, $supp$: Aid-**set**)

ext wr $coals \;:\; Cid \xrightarrow{m} Coalition$

pre $c \in \mathbf{dom}\, coals \wedge a \in coals(c) \wedge$
$mandated(coals(c), supp)$

post if $coals(c) = \{a\}$
then $coals = \{c\} \lhd \overleftarrow{coals}$
else $\overleftarrow{coals} \dagger \{c \mapsto \mu(\overleftarrow{coals}(c), members \mapsto \overleftarrow{coals}(c).members - \{a\})\}$

$mandated : Coalition \times Aid\text{-}\mathbf{set} \to \mathbb{B}$

$mandated(mk\text{-}Coalition(membs, thr), supp) \;\; \triangleq$
$supp \subseteq membs \wedge \mathbf{card}\, supp / \mathbf{card}\, membs \geq thr$