Trust Domains: An Algebraic, Logical, and Utility-theoretic Approach

Gabrielle Anderson and Matthew Collinson and David Pym

University of Aberdeen
Scotland, U.K.
{g.a.anderson, matthew.collinson, d.j.pym}@abdn.ac.uk

Abstract. Complex systems of interacting agents are ubiquitous in the highly interconnected, information-rich ecosystems upon which the world is more-or-less wholly dependent. Within these systems, it is often necessary for an agent, or a group of agents, such as a business, to establish within a given ecosystem a trusted group, or a region of trust. Building on an established mathematical systems modelling framework — based on process algebra, logic, and stochastic methods — we give a characterization of such 'trust domains' that employs logical assertions of the properties required for trust and utility-theoretic constraints on the cost of establishing compliance with those properties. We develop the essential meta-theory and give a range of examples.

1 Introduction

Complex systems of interacting agents are ubiquitous in the highly interconnected, information-rich ecosystems upon which the world is more-or-less wholly dependent. Within these systems, it is often necessary for an agent, or a group of agents, such as a business, to establish within a given ecosystem a trusted group, or a region of trust.

In this paper, we are concerned with characterizing such 'trust domains' within a mathematical systems modelling framework. Developing the modelling framework developed in [6], and initial ideas about trust domains presented in [3], we provide a characterization that employs logic — in order to determine the properties that an agent, or group of agents, must satisfy in order to be trusted — and utility — in order to determine and limit the cost of establishing compliance with these properties.

The key components of the framework are the following: first, a resource-sensitive process algebra, within which the decision-theoretic notion of utility is used to establish the relative cost of different choices; second, a corresponding, resource-sensitive modal logic of processes; and third, a conceptual notion of trust domain, characterized using our algebraic, logical, and utility-theoretic tools, that encapsulates the intuitions described above.

In Section 2, we discuss our motivation in modelling 'trust domains', giving a precise, but informal, introduction to the concept. In Section 3, we provide the necessary conceptual background to our systems modelling approach and, in Section 4, we give a mathematical formulation of a weighted (costed) process algebra that provides a suitable basis for modelling systems and decision-making about choices within them. In

Section 5, we describe the associated modal, substructural logic. Cost modalities play a key role in describing trust domains. In Section 6, we return first, in the light of our mathematical set-up, to the definition of trust domains, and proceed to discuss a range of examples. In Section 7, we discuss some directions for further research.

2 Trust Domains

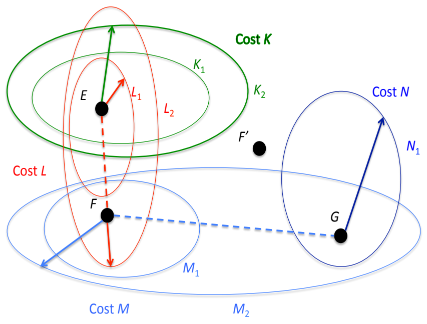

Fig. 1. Iso-utilities and Trust Domains

In systems of interacting agents, an individual or group of agents may establish a part of the system, or a collection of agents within the system, that it trusts. Similarly, a system's designer or manager might establish a collection of parts of the system such that, within any given part, the agents trust one another. We shall refer to such a part of the system, or such a collection of agents, as a 'trust domain'. This term is used in the Trusted Computing Project (`www.trustedcomputing.org.uk`), the Open Trusted Computing consortium (`www.opentc.net`), and the 'Trust Domains' project (`www.hpl.hp.com/research/cloud_security/TrustDomains.pdf`). The literature on models of trust is very large and cannot be surveyed in this short article, but a good survey with a relevant perspective for us is [18].

In this section, we consider informally how an agent might decide which part of its system, or which agents within the system, to trust. We propose a characterization of a trust domain for a given agent within a given system that has two components. First, a logical assertion that expresses the properties that must be possessed by any trusted agent. Second, a cost bound that limits the extent to which the system around the agent can be trusted; that is, the agent will trust only those parts of the system that can be reached or observed within a given expenditure of resource. The intended situation is depicted schematically in Figure 1. This picture is intended to be understood in the context of the classical model of distributed systems (see, for example, [7, 6]) in which processes (here, agents) execute relative to collections of resources, located at specified places within the system. The system is understood as residing within an environment from which events are incident upon it and to which it exports events [7, 6].

Here the agent, E, may be given one of two different choices of cost function, K_E. If $K_E = K$, then F is not within E's trust domain at either the K_1 or K_2 levels. If, however, $K_E = L$, then F is within E's trust domain at the L_2, but not at the L_1 level. Agent F's cost function, M, includes agent G at the M_2 level, but not at the M_1 level ($M_1 < M_2$). F' is in no-one's domain at any of the given levels of cost.

Examples of the propositions that might be associated with a trust domain include access control assertions, such those as described in [2, 1, 6] (and references therein).

We make use of logical *cost modalities*, that place bounds on the cost that is acceptable for a given agent. This formalizes the above notion of levels of trust.

In Section 6, we consider three examples of trust domains, illustrating different aspects of their form and use. Our focus here is on modelling decision-making about trust in situations inspired by corporate environments.

1. *Establishing Boundaries*. In a dock, the harbour master and the captain of an inbound ship must determine at what point responsibility for navigating the ship should pass between them: The captain must trust the harbour master with the care of the ship and the harbour master must trust the captain to navigate the dock safely.
2. *Contract Choices*. In the management of mergers & acquisitions, the deal team must establish valuations of the principals. This requires access to highly confidential information, and the deal team may need to outsource specialized parts of the valuation, so that is a risk that confidential data may be lost. Who should be trusted by the deal team? And with what information?
3. *Information Provenance*. When establishing data-sharing arrangements, the provenance of the evidence used to assess, say, the reliability of the outsourced service provider is of critical importance. How does an agent decide to trust that evidence?

The examples are not intended to cover all of the interesting issues: they are illustrative.

We aim to provide a modelling framework within which this notion of trust domain can be established and shown capable of handling substantial examples, encompassing a variety of situations. We begin with a mathematical treatment of distributed systems, including an account of how agents' choices are modelled.

3 Systems Modelling and Decision-making

The classical model of distributed systems, such as described in [7], provides the conceptual inspiration for our modelling framework, which builds on [6, 3]. First, a model resides within an environment; that is, the part of the system not modelled in explicit structural detailed. The interaction between the model of interest and its environment is captured mathematically using stochastic processes to provide occurrences of events. Second, the structural components of a model can be described as follows:

- Location: locations are the places within (and, indeed, outwith) the system at which resources reside; locations can be logical or physical;
- Resource: resources are the building blocks of the system's services; they can, for example, be consumed, created, and moved between locations by processes;
- Process: processes deliver services within and outwith the system, manipulating the resources that are distributed around the system's logical and physical locations; they interact with the system's environment.

Mathematically, our treatment of process is based on Milner's synchronous calculus of communicating systems (SCCS) [15], as developed as a basis for systems modelling in [6]; note that asynchronous calculi can be encoded within synchronous calculi [15]. The key idea is that locations, resources, and processes co-evolve, according to a judgement $L, R, E \xrightarrow{a} L', R', E'$, which is read as 'the process E, using resources R at locations L, performs action a and so becomes the process E' that is able to evolve using resources R' at locations L' '. For simplicity of presentation, and with little loss of generality for our present purposes, we suppress locations in the remainder of this presentation, though make informal use of them in our examples, in Section 6. The reader might think of them either as implicitly present, or consider them to be rolled up into the definition of resources (see [6] for relevant technical support).

This judgement is defined using a structural operational semantics, such as in the definition of SCCS [15, 6]. Three mathematical details are important. First, actions are required to form a commutative monoid (a and b combine to form ab). Second, resources are assumed to form a preordered partial commutative *resource monoid*, $\mathbf{R} = (\mathbf{R}, \sqsubseteq, \circ, e)$, in which resource elements $R_1, R_2 \in \mathbf{R}$ can be combined, using the monoid operation to form $R_1 \circ R_2$ (with unit e) or compared, $R_1 \sqsubseteq R_2$, say, using the preorder. The structure of the monoid is subject to some coherence conditions [17, 6]. A key example of a monoid of resources is given by the natural numbers (with 0), with addition as the monoid operation and less-than-or-equals as the order: $(\mathbb{N}, \leq, +, 0)$. Third, the relationship between actions and resources must be specified using a *modification function* that specifies the effect of performing an action a on a resource element R: that is, $\mu : (a, R) \mapsto R'$. Modification functions must satisfy some (mild) coherence conditions relating the monoid structure of actions and the monoidal structure of resources (details may be found in [6]). This treatment of resource just as in bunched logic [17] and in various versions of separation logic [19] and, for brevity, we refrain from further rehearsing its justification here.

With this set-up, the operational semantics, in its basic form, admits rules such as

$$\frac{R, E \xrightarrow{a} R', E' \quad S, F \xrightarrow{b} S', F'}{R \circ S, E \times F \xrightarrow{ab} R' \circ S', E' \times F'} \quad \text{and} \quad \frac{R, E_i \xrightarrow{a} R', E_i'}{R, E_1 + E_2 \xrightarrow{a} R', E_i'} \quad (i = 1, 2)$$

giving, respectively, concurrent product and simple non-deterministic choice.

In determining the extent of trust domains, agents do not, however, make simple non-deterministic choices. Rather, they make choices according to their (situated, or located) preferences. Accordingly, we must set up a version of the approach sketch above that captures a suitable representation of preference-driven choice.

Many process calculi include a form of prioritized sum, for example [20]. In prioritized sums, say $w \cdot a : E + w' \cdot b : F$, with $w > w'$, the option $a : E$ is always preferred in any context in which both a and b are available (which they may not be, because of restriction operations). By contrast, we argue that an agent, even with the same potential choice of actions, should be permitted to associate different costs to the same options, dependent on its situation and the properties of the agents with which it is interacting. To this end we make use of *cost-dependent choice* (or simply *sum*) $\sum_u E_i$, in which an agent has a choice between alternatives E_i, and its preference is codified by the cost function u. Cost is used as in utility theory to encompass uplift for revenue; that is, as in loss. Cost functions map from a resource-process pair to rational numbers.

As we wish to model agents whose preferences differ dependent on their situation, the cost ascribed to the different possible choices at a choice point must not rely on those potential choices alone. For example, if the choice $R, a : E +_u b : F$ (here we use an infix notation) occurs within a wider context $R \circ S, (a : E +_u b : F) \times G$, then the preferences in the given context are determined by the cost calculations $o = u(R \circ S, a : E \times G)$ and $p = u(R \circ S, b : F \times G)$. If the first summand is chosen then we annotate the cost o on the evolution arrow and, if the second is chosen, then we annotate p on the evolution arrow. An occurrence of the same choice $R, a : E +_u b : F$ within a different context, such as $R \circ T, (a : E +_u b : F) \times H$ with $G \neq H$ or $S \neq T$, may have different cost calculations and associated costs.

Along with the concept of a process algebra with costs comes an associated modal logic. Just as in [3, 6], the logic admits the usual classical (or, if preferred, intuitionistic) propositional connectives, as well as thus usual 'separating' or 'resource-sensitive' multiplicatives from bunched logic [17], as in [6], and action modalities, as in Hennessy–Milner logic [10, 9, 16].

Critically in our present setting, it also admits weighted, or cost, modalities. Utility-based decision-making for ordinal preferences can be incorporated into a process-theoretic setting [3]. Standard economic reasons (particularly moves towards uncertain outcomes) require the further development of this to cardinal utility, and we embark upon such a task in this paper. The possibility connectives $\langle \leq n \rangle \phi$ and $\langle > n \rangle \phi$ denote that there exists an evolution whose cost m is, respectively, less than or equal to, or greater than, n, where the resulting state (or continuation) satisfies ϕ. The necessity connectives $[\leq n]$ and $[> n]$ denote, respectively, that in all evolutions whose cost m is less than or equal to, or greater than, n, where the resulting state satisfies ϕ. Thus, a logical property of a trust domain that is guarded by a sequence of choices has an associated cost capturing the associated agent's preferences and determining which choice is made.

4 A Process Algebra with Contextual Costs

Section 3 describes our conceptual formulation of system models and the associated model of decision-making. We now establish the necessary mathematical set-up. Related work, based on Milner's π-calculus, is in [8].

In order to consider the different situations or contexts that a given process trusts, we wish to consider how a process values its options dependent on the context in which it occurs. As such, the sub-evolutions of a composite process may not be independent of each other. This is in contrast with typical process calculi, where the behaviour of a composite process is usually defined in terms of the behaviours of its sub-processes alone.

To see how this works, consider a choice $R, a : E +_u b : F$ that occurs as part of a wider model $R \circ S, (a : E +_u b : F) \times G$. When a choice is made we annotate the cost n of the chosen summand on the evolution (e.g., $R, a : E +_u b : F \Longrightarrow^n R', a : E$). As these costs depend on the context, then the evolution needs to know what this context is. We henceforth annotate the context in which a process is evolved on the underside of the evolution arrow (e.g., $R, a : E +_u b : F \underset{S,[\,]\times G}{\Longrightarrow}^n R', a : E$), where $[\,]$ denotes the hole into which $a : E +_u b : F$ may be substituted to regain the complete system $(a : E +_u b : F) \times G$, and S are the resources allocated to G. In addition, any choices in $[\,] \times G$ will make use of the process that is substituted into the hole $[\,]$. We therefore annotate the process that is substituted, into the process being evolved, on top of the evolution arrow; for example, $S, [\,] \times G \overset{R, a:E+_u b:F}{\Longrightarrow}^m S', [\,] \times G'$.

In essence, the judgement for evolution for processes with cost of the form

$$C \xRightarrow[C_1]{C_2}{}^{n} C' \tag{1}$$

denotes how a context C, that exists in a system that can be decomposed as $C_1(C(C_2))$, evolves in terms of its choices. We refer to C as the *(primary) context*, C_1 as the *outer context*, and C_2 as the *substituted*, or *inner context*. Intuitively this denotes the evolution of one part, C, of an entire system, $C_1(C(C_2))$. In order to reason compositionally, we wish to be able to describe the evolution of C independently and structurally. As choices take account of context, this is not possible. The semantics of choice, however, makes use of just the definition of the inner and outer context, disregarding their structure. So we do not need to make use of the structure of C_1 and C_2, as we do with C, but need only record their definitions, for reference at choice points. They are therefore annotated on the evolution arrow, but are not evolved in that relation.

We now describe the theoretical set-up in detail. Assume a set U of symbols, called *formal costs*, with a distinguished element 0_U, called the *neutral cost*. *Processes* are generated by the grammar

$$E ::= \mathbf{1} \mid [\,] \mid a : E \mid \sum_{i \in I}{}_u E_i \mid E \times E. \tag{2}$$

These are really process contexts: the term $[\,]$ is a *hole* into which other processes may be substituted. For this work, it turns out to be convenient to develop contexts as first-class citizens rather than merely meta-theoretic tools.

The *choice* $\sum_{i \in I}{}_u E_i$ is the key construct: it describes situations in which an agent has a choice between alternatives E_i indexed by a $i \in I$, and its preference (in a larger context) is codified by the cost $u \in U$. The infix operator $E +_u F$ may be used for binary sums, and the subscript u may be dropped when $u = 0_U$. The *zero* process $\mathbf{0}$ is defined to be the sum indexed by the empty set and the neutral cost. The zero process, *unit* process $\mathbf{1}$, and *synchronous products* $E \times F$ are well-known in process calculus, as are *prefixes* $a : E$, where $a \in$ Act. Assume, for each formal cost $u \in U$, an associated, real-valued *cost function* $u : Cont \longrightarrow \mathbb{R}$ [13] that fixes an interpretation for each formal symbol $u \in U$. The identically zero function is associated with 0_U. Henceforth, we do not distinguish between formal costs and their costs functions.

A process E is *well-formed* if it contains at most one hole and that hole is not guarded by action prefixes. The process E is *closed* if it has no holes and *open* otherwise. Let $PCont$ be the set of all well-formed processes, $PCCont$ be the set of all closed well formed processes, and $POCont$ be the set of all open well-formed processes. Let $\mathbf{R}$ be a resource monoid and μ be a fixed modification function, as defined in Section 3. Define the products of sets $Cont = \mathbf{R} \times PCont$, $CCont = \mathbf{R} \times PCCont$ and $OCont = \mathbf{R} \times POCont$. The letter C is reserved for contexts. Define $C_\emptyset = e, [\,]$. Brackets will be freely used to disambiguate both processes and contexts. For $C = R, E$, the notational abuses $C \times F = R, (E \times F)$ and $C +_u F = R, (E +_u F)$ will sometimes be used. Substitution in processes, $E(F)$, replaces all occurrences of $[\,]$ in E with F; for example, $(([\,] +_u E) \times G)(F) = (F +_u E) \times G$. Substitution of contexts $C_1(C_2)$, where $C_1 = R, E$ and $C_2 = S, F$, is defined as follows: if E is open,

$$\frac{}{R, \mathbf{1} \xrightarrow[C_1]{C_2}{}^{1} R, \mathbf{1}}\,(\textsc{Tick}) \qquad \frac{}{R, a : E \xrightarrow[C_1]{C_2}{}^{a} \mu(a,R), E}\,(\textsc{Prefix}) \qquad \frac{C_2 \xrightarrow[C_1]{(e,\mathbf{1})}{}^{a} C_2'}{e,[\,] \xrightarrow[C_1]{C_2}{}^{1} e,[\,]}\,(\textsc{Hole})$$

$$(S\times)\frac{R, E \xrightarrow[C_3]{C_2}{}^{a} R', E' \qquad S, F \xrightarrow[C_4]{C_2}{}^{b} S', F'}{R \circ S, E \times F \xrightarrow[C_1]{C_2}{}^{ab} R' \circ S', E' \times F'}\,(\textsc{Prod})$$

Fig. 2. Action Operational Semantics

$$\frac{}{R, \mathbf{1} \xRightarrow[C_1]{C_2}{}^{0} R, \mathbf{1}}\,(\textsc{TickW}) \qquad \frac{}{R, a : E \xRightarrow[C_1]{C_2}{}^{0} R, a : E}\,(\textsc{PrefixW}) \qquad \frac{C_2 \xRightarrow[C_1]{(e,\mathbf{1})}{}^{n} C_2'}{e,[\,] \xRightarrow[C_1]{C_2}{}^{0} e,[\,]}\,(\textsc{HoleW})$$

$$\frac{n = u(C_1(R, E_i(C_2)))}{R, \sum_{I}{}_{u} E_i \xRightarrow[C_1]{C_2}{}^{n} R, E_i}\,(\textsc{SumW}) \qquad (S\times)\frac{R, E \xRightarrow[C_3]{C_2}{}^{o} R, E' \qquad S, F \xRightarrow[C_4]{C_2}{}^{p} S, F'}{R \circ S, E \times F \xRightarrow[C_1]{C_2}{}^{o+p} R \circ S, E' \times F'}\,(\textsc{ProdW})$$

Fig. 3. Operational Semantics of Cost

then $C_1(C_2) = R \circ S, E(F)$, where $E(F)$ is process substitution; if E is closed, then $C_1(C_2) = C_1$.

Developing the formulation sketched above, we separate the operational semantics into two dimensions: the evolution system for performing actions (Figure 2) and the evolution system for determining the cost of possible choices (Figure 3), as in [20], and building on [3]. Overall, the evolution sequences for the calculus are interleavings of the two dimensions. The operational semantics for performing actions is defined in Figure 2. The unit process always ticks, effecting no change. The prefix process evolves via its head action. The hole rule is a technical one used to terminate evolution derivations of open contexts. An important feature of this system is that contextual information about conclusions is propagated up to premisses. In the product case, information about each premiss is propagated up from the conclusion to the other premiss, so that derivations of transitions occur in context. This is effected by the side-condition $(S\times)$ is which states that $C_3 = C_1((S, F(C_2)) \times [\,])$ and $C_4 = C_1((R, E(C_2)) \times [\,])$.

The operational semantics for determining the cost of possible choices, as defined in Figure 3, is used to determine the cost of a given set of choices of a process. A neutral cost is given to tick, prefix, and hole processes, as they contain no choices. The sum process $\sum_{I}{}_{u} E_i$ represents a preference-based choice by the agent: it evolves to one of its summands, annotating the value of that summand in the wider context on the evolution arrow, according to its cost function u. A special case of the sum is for the zero process $\mathbf{0}$, which never evolves. The product evolves two processes synchronously in parallel, according to the decomposition of the associated resources, and annotates the sum of the sub-processes' costs on the evolution arrow. This approach to combining costs, and the value given to tick or prefix processes, is one possible design decision, and will be considered more fully in future work. We make use of the abbreviation $C \xRightarrow{n} C'$ and $C \xrightarrow{a} C'$ to denote $C \xRightarrow[e,[\,]]{e,\mathbf{1}}{}^{n}\ C'$ and $C \xrightarrow[e,[\,]]{e,\mathbf{1}}{}^{a}\ C'$, respectively.

To demonstrate how contextual decisions can be utilized in modelling, we give a simple example (inspired by [5]). Consider a banker who has a presentation (for a client, that includes confidential business data) on a USB drive. The banker may chose to access the drive or not, depending on the situation. The banker is modelled as a process

$$Banker = present : Banker' +_{u_B} idle_B : Banker', \tag{3}$$

where u_B represents its costs. The banker may be willing to access the presentation when visiting a client, on the assumption that the client's network is firewalled, so making the document safe from attack. In order to do so, however, the banker must be given access to a computer by the client. The client is modelled as

$$Client = logIn : Client' + idle_C : Client', \tag{4}$$

which, for simplicity, makes a non-deterministic choice between logging the guest in and idling. The interaction between the banker and the client is a form of joint access control (i.e., both agents must grant access), in which the banker cannot show the presentation without having been logged in, and the client cannot see the presentation unless the banker accesses it. If the banker's cost function is u_B, then we have

$$u_B(C_C(R, idle_B : Banker')) = 0.3 \qquad u_B(C_C(R, present : Banker')) = 0.1. \tag{5}$$

and the banker would prefer to present the work. Here, where C_C is the client context, the banker can access the presentation with a low cost

$$R, Banker \xRightarrow[C_C]{e,\mathbf{1}}{}^{0.1}\ R, present : Banker' \xrightarrow{present} R, Banker'. \tag{6}$$

In a different situation — here, a different context — the banker may have different costs associated with the possible choices. Consider a home computer, compromised by an attacker who wants to steal the presentation, but cannot do so unless the banker accesses it from the USB stick. The attacker is modelled as

$$Attacker = steal : Attacker' + idle_A : Attacker' \tag{7}$$

In this situation, the banker prefers to idle than to work on the presentation.

$$u_B(C_A(R, idle_B : Banker')) = 0.2 \quad u_B(C_A(R, present : Banker')) = 0.6. \tag{8}$$

and the banker has a much higher cost, due to the increased risk of the data being stolen, when performing the $present$ action

$$R, Banker \overset{e,\mathbf{1}}{\underset{C_A}{\Longrightarrow}}{}^{0.6} R, present : Banker' \xrightarrow{present} R, Banker'. \tag{9}$$

Were we reasoning about the decisions that the banker would make, we could easily argue that the document would not be accessed from home, as the banker's cost for doing so is so high. Indeed, we could straightforwardly implement this by introducing a restriction operator to the language to filter choices above/below a given cost bound.

A fundamental aspect of process calculus is the ability to reason equationally about behavioural equivalence of processes [15]. We now adapt these notions to suit the calculus above, which incorporates ideas from [6].

The *bisimilarity (or bisimulation) relation* $\sim \subseteq PCont \times PCont$ is the largest binary relation such that, if $E \sim F$, then for all $a \in$ Act, for all $R, R', S, T \in \mathbf{R}$, and for all $G, H, I, J \in PCont$ with $G \sim I$ and $H \sim J$, then

1. for all $E' \in PCont$, if $R, E \overset{T,H}{\underset{S,G}{\longrightarrow}}{}^{a} R', E'$, then there is F' such that $R, F \overset{T,J}{\underset{S,I}{\longrightarrow}}{}^{a} R', F'$ and $E' \sim F'$, and if $R, E \overset{T,H}{\underset{S,G}{\Longrightarrow}}{}^{n} R, E$, then there is F' such that $R, F \overset{T,J}{\underset{S,I}{\Longrightarrow}}{}^{n} R, F'$ and $E' \sim F'$, and
2. for all $F' \in PCont$, if $R, F \overset{T,J}{\underset{S,I}{\longrightarrow}}{}^{a} R', F'$, then there is E' such that $R, E \overset{T,H}{\underset{S,G}{\longrightarrow}}{}^{a} R', E'$ and $E' \sim F'$, and if $R, F \overset{T,J}{\underset{S,I}{\Longrightarrow}}{}^{n} R, F'$, then there is E' such that $R, E \overset{T,H}{\underset{S,G}{\Longrightarrow}}{}^{n} R, E'$ and $E' \sim F'$.

The union of any set of relations that satisfy these two conditions also satisfies these conditions, so the largest such relation is well-defined. Define $\sim \subseteq Cont \times Cont$ by: if $E \sim F$ then $R, E \sim R, F$ for all $R \in \mathbf{R}$ and $E, F \in Cont$.

Definition 1. *A cost function, u,* respects bisimilarity *if, for all $C_1, C_2 \in Cont$, $C_1 \sim C_2$ implies $u(C_1) = u(C_2)$.*

That is, behaviourally equivalent (bisimilar) states are required to be indistinguishable by u. Note that the cost reductions $\Rightarrow^n$ used in the definition of bisimulation do not necessarily use cost functions to determine the cost n, as the base case reduction rules for tick, prefix, and hole processes all output a constant zero cost. The set U of utilities respects bisimilarity if every $u \in U$ respects bisimilarity. Any real-valued function defined on the quotient $Cont/\sim$ defines a cost that respects bisimilarity. Henceforth cost functions are assumed to respect bisimilarity. We can show that if bisimilar contexts are substituted into each other, then the result is bisimilar:

Proposition 1. *If $E \sim G$ and $F \sim H$ then $E(F) \sim G(H)$.*

All proofs are omitted in this short paper.

With this result, we can obtain a key property for reasoning compositionally.

Theorem 1 (Bisimulation Congruence). *The relation $\sim$ is a congruence. It is reflexive, symmetric and transitive, and for all a, E, F, G with $E \sim F$, and all families $(E_i)_{i\in I}$, $(F_{i\in I})_I$ with $E_i \sim F_i$ for all $i \in I$, $a : E \sim a : F$, $E \times G \sim F \times G$, and $\sum_{i\in I}{}_u E_i \sim \sum_{i\in I}{}_u F_i$.*

In order to reason equationally about processes, it is also useful to establish various algebraic properties concerning parallel composition and choice. We derive these below, for our calculus. We use the binary version of sum here in order to aid comprehension, but finite choices between sets of processes work straightforwardly.

Proposition 2 (Algebraic Properties). *(1)* $E +_u F \sim F +_u E$*; (2)* $E \times \mathbf{0} \sim \mathbf{0}$*; (3)* $E \times \mathbf{1} \sim E$*; (4)* $E \times F \sim F \times E$*; and (5)* $E \times (F \times G) \sim (E \times F) \times G$.

5 A Cost-sensitive Modal Logic

We now introduce a cost-sensitive modal logic of system properties. The semantics is given using a satisfaction relation

$$C \models_{C'} \phi, \tag{10}$$

where C is a closed context, C' is an open context, and ϕ is a formula of a (Hennessy–Milner-style) modal logic of processes: this may be read 'the *primary context* C satisfies ϕ in the *surrounding context* C'' (cf. (1)). The context C may satisfy different logical propositions, perhaps even negations of each other, when placed in different surrounding contexts; an example of this is below. Context-sensitive logics have been studied previously [14, 4]. The structural nature of processes and resources provides a semantic framework in which such logics seem particularly natural.

The propositions of the logic are defined by the grammar

$$\begin{aligned}\phi ::= p \mid \bot \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \langle a \rangle \phi \mid [a]\phi \mid I \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi \mid \\ \langle \leq n \rangle \phi \mid [\leq n]\phi \mid \langle > n \rangle \phi \mid [> n]\phi,\end{aligned} \tag{11}$$

where p ranges over atomic propositions, a over actions, and n over rational numbers. The symbols for propositions for *truth*, *falsehood*, *negation* and *(additive) conjunction*, *disjunction*, and *implication* are standard. The (*additive*) *modal connectives* are $\langle a \rangle$ and $[a]$. The connectives I, $*$, and $\mathrel{-\!\!*}$ are the *multiplicative unit*, *conjunction*, and *implication*, respectively. The (*cost*) *modal connectives* are $\langle \leq n \rangle$, $[\leq n]$, $\langle > n \rangle$, $[> n]$, and denote possible and necessary modal bounds on costed evolutions.

The interpretation of cost modalities is straightforward. The possibility connectives $\langle \leq n \rangle \phi$ and $\langle > n \rangle \phi$ denote that there exists an evolution whose cost m is less than or equal to, or greater than, n, respectively, where the resulting state satisfies ϕ. The necessity connectives $[\leq n]$ and $[> n]$ denote that in all evolutions whose cost m is less than or equal to, or greater than, n, respectively, where the resulting state satisfies ϕ. The satisfaction relation for cost modalities is specified in Figure 4, with the satisfaction relation for additive formulae specified in Figure 5, and that for multiplicative formulae specified in Figure 6.

We describe the interpretation with an example: recall the model of the banker's context dependent preferences and choices (cf. (3-9)). In the client context, the banker has a low-cost choice of 0.1 (cf. 6), but in the attacker context all its possible evolutions are of higher cost of 0.2 and 0.6, respectively (cf. 9). Hence we can show that the banker

$C_1 \models_{C_2} \langle \leq n \rangle \phi$	iff	there are C'_1, C'_2, m, o such that $C_1 \overset{e,\mathbf{1}}{\underset{C_2}{\Longrightarrow}}{}^{m} C'_1$ and $C_2 \overset{C_1}{\underset{C_\emptyset}{\Longrightarrow}}{}^{o} C'_2$, and $m \leq n$ and $C'_1 \models_{C'_2} \phi$
$C_1 \models_{C_2} [\leq n] \phi$	iff	for all C'_1, C'_2, m, o such that if $C_1 \overset{e,\mathbf{1}}{\underset{C_2}{\Longrightarrow}}{}^{m} C'_1$ and $C_2 \overset{C_1}{\underset{C_\emptyset}{\Longrightarrow}}{}^{o} C'_2$ and $m \leq n$, then $C'_1 \models_{C'_2} \phi$
$C_1 \models_{C_2} \langle > n \rangle \phi$	iff	there are C'_1, C'_2, m, o such that $C_1 \overset{e,\mathbf{1}}{\underset{C_2}{\Longrightarrow}}{}^{m} C'_1$ and $C_2 \overset{C_1}{\underset{C_\emptyset}{\Longrightarrow}}{}^{o} C'_2$, and $m > n$ and $C'_1 \models_{C'_2} \phi$
$C_1 \models_{C_2} [> n] \phi$	iff	for all C'_1, C'_2, m, o such that if $C_1 \overset{e,\mathbf{1}}{\underset{C_2}{\Longrightarrow}}{}^{m} C'_1$ and $C_2 \overset{C_1}{\underset{C_\emptyset}{\Longrightarrow}}{}^{o} C'_2$ and $m > n$, then $C'_1 \models_{C'_2} \phi$

Fig. 4. Interpretation of Propositional Cost Modalities

process has different logical properties in different contexts

$$R, Banker \models_{C_C} \langle \leq 0.1 \rangle \top \qquad R, Banker \models_{C_A} \neg(\langle \leq 0.1 \rangle \top), \tag{12}$$

where $\top$ is a formula that is true for all processes in all contexts.

The standard interpretation of Hennessy–Milner logics uses the relation specified by the operational semantics as a Kripke structure to support the modal connectives. In our work, the operational semantics is more complex: a context occurs, and evolves alongside an outer context. Therefore, when we consider whether $C_1 \models_{C_2} \langle \leq n \rangle \phi$ holds, we have to consider whether there are evolutions of the form $C_1 \overset{e,\mathbf{1}}{\underset{C_2}{\Longrightarrow}}{}^{m} C'_1$ *and* $C_2 \overset{C_1}{\underset{C_\emptyset}{\Longrightarrow}}{}^{o} C'_2$ such that $C'_1 \models_{C'_2} \phi$ and $m \leq n$. The occurrence of the tick process and the empty context ensure that no extraneous contextual information is introduced into the evolutions of interest. Other modal operators are interpreted similarly.

A *valuation*, $\mathcal{V}$, is a function that maps each atomic proposition to a $\sim$-closed set of closed contexts. In the interpretation of atoms, the surrounding context is wrapped around the primary context, and the valuation of the atom consulted to see if it contains this compound context. This is what makes our logic context-sensitive. $\top$, $\bot$, $\neg$, $\wedge$, $\vee$, and $\rightarrow$ are all interpreted (essentially) classically. The interpretation of the multiplicative connectives here is similar to that for the logic MBI in [6].

Recall again the example of the banker who decides which actions to take in different contexts (3-9). In a situation that consists of a client (context C_C), the banker can access the presentation with low cost, but in a situation that consists of an attacker (context C_A) the banker accessing the presentation has a high cost; that is,

$$R_B, Banker \models_{C_C} \langle \leq 0.3 \rangle \langle present \rangle \top \text{ and } R_B, Banker \models_{C_A} \neg(\langle \leq 0.3 \rangle \langle present \rangle \top). \tag{13}$$

$C \models_{C'} p$	iff	$C'(C) \in \mathcal{V}(p)$
$C \models_{C'} \bot$		never
$C \models_{C'} \top$		always
$C \models_{C'} \neg\phi$	iff	$C \not\models_{C'} \phi$
$C \models_{C'} \phi \wedge \psi$	iff	$C \models_{C'} \phi$ and $C \models_{C'} \psi$
$C \models_{C'} \phi \vee \psi$	iff	$C \models_{C'} \phi$ or $C \models_{C'} \psi$
$C \models_{C'} \phi \rightarrow \psi$	iff	$C \models_{C'} \phi$ implies $C \models_{C'} \psi$
$C_1 \models_{C_2} \langle a \rangle \phi$	iff	there are C_1', C_2', b such that if $C_1 \xrightarrow[C_2]{e,\mathbf{1}}{}^{a} C_1'$and $C_2 \xrightarrow[C_\emptyset]{C_1}{}^{b} C_2'$, then $C_1' \models_{C_2'} \phi$
$C_1 \models_{C_2} [a]\phi$	iff	for all C_1', C_2', b such that if $C_1 \xrightarrow[C_2]{e,\mathbf{1}}{}^{a} C_1'$ and $C_2 \xrightarrow[C_\emptyset]{C_1}{}^{b} C_2'$, then $C_1' \models_{C_2'} \phi$

Fig. 5. Interpretation of Additive Propositional Formulae

$R, E \models_{C'} I$	iff	$R = e$ and $E \sim \mathbf{1}$
$R, E \models_{C'} \phi * \psi$	iff	there are S, T, F, G such that $R = S \circ T$, $E \sim F \times G$, and $S, F \models_{C'(T,[\,]\times G)} \phi$ and $T, G \models_{C'(S, F\times[\,])} \psi$
$R, E \models_{C'} \phi \mathrel{-\!\!*} \psi$	iff	for all S, F such that $R \circ S$ is defined and $S, F \models_{C'} \phi$, $R \circ S, E \times F \models_{C'} \psi$

Fig. 6. Interpretation of Multiplicative Propositional Formulae

Hence, in different contexts the process satisfies different propositions that, moreover, would be inconsistent over the same context.

If we make use of real value quantification we can recover optimality properties about the least or most costly choices, as in [3]. For example, we could state that the most costly option has logical property ϕ as $\exists x.[> x]\bot \wedge \langle \leq x \rangle \phi \wedge \neg[> x]\neg\phi$, using standard techniques to define equality with inequalities and negation. This could be used to reason about a scheduler's possible options.

Behaviourally equivalent processes are also logically equivalent (they satisfy the same logical properties). This is half of the Hennessy–Milner property [10, 9].

Theorem 2. *If $C_1 \models_{C_2} \phi$, and $C_1 \sim C_3$, and $C_2 \sim C_4$, then $C_3 \models_{C_4} \phi$.*

Hence, bisimilar processes can be used interchangeably within a larger system, without changing the logical properties of the larger system.

It is unclear whether a useful converse can be obtained, for the given bisimulation relation. With restrictions on the available fragments of the logic, and a different (*local*) equivalence relation, however, it is possible to obtain a converse [3]. The local equivalence, however, fails to be a congruence, and as such its usefulness is limited. It is a strictly local reasoning tool.

The logic might also be enriched to handle expected cost [13]. Quantitative path-based logical properties of Markov Chains are studied in [11]: they support reasoning about complex notions such as average utility with a given time discount, but do not provide compositionality results over model structures. A more extensive study of such extensions is future work.

In game-theoretic approaches to security, the notion of a level of security is important. That is, if a defender chooses to perform some defensive action: then all possible attacks have a high cost for the attacker. With preference modalities we can make statements relevant to security levels. To see this, consider the proposition

$$\phi \mathrel{-\!\!*} [< n][d](\neg\langle a\rangle\top), \tag{14}$$

This proposition states that any attacker that is characterized by ϕ, when a defensive action is effected, there is no possible choice to attack that incurs a cost less than bound n (we hold the costs of the defender constant). The multiplicative implication operator permits us to reason about composition within arbitrary processes, and hence of the efficacy of defensive measures relative to an arbitrary (partially) described attacker. The interaction between multiplicative implication and cost operators is surprisingly powerful, and can be used to describe the intuitive description of trust domains [3].

6 Trust Domains Revisited

In Section 2, Given different preference functions and different bounds, a given agent may decide to trust different agents. We now formalize that definition, and provide a selection of examples that demonstrate how trust domains can model natural problems.

In [3], trust domains are defined relatively informally, in terms of an agent E that is considering what to trust, a logical property ψ that denotes some *goal property* for the agent, and a cost bound n. A trust domain then consists of the set of contexts into which the agent can be substituted, where the entire system can evolve to some other system that satisfies the goal property, and the cost of the evaluation, is within the cost bound n.

Here, using a cost modality, we define a trust domain as

$$TD((R,E),\phi,\psi,n) = \{S,F \mid S,F \vDash_{C_\emptyset} \phi \text{ and } R\circ S, E\times F \vDash_{C_\emptyset} \langle\leq n\rangle\psi\}, \tag{15}$$

where ϕ limits the agents being considered with, for example, some locality condition.

Recall that multiplicative implication is valid when, for any context that fulfils the left hand side of the implication, if it is composed with the current agent, then the joint system fulfils the right hand side of the implication. In essence, a trust domain is the collection of such contexts, for the logical property $\phi \mathrel{-\!\!*} \langle\leq n\rangle\,\psi$ interpreted with respect to the agent R, E that is doing the trusting.

We now turn to the three examples from Section 2 in order to illustrate these ideas.

Establishing Boundaries. We consider how a harbour master and a ship's captain establish an appropriate point at which to transfer control of the ship between them. In attempting to establish a boundary between the parts of a system controlled by co-evolving agents, use must be made of the agents' preferences. We must consider both agents with 'fixed' preferences, which do depend on the context structure, such as heavy seas, but not on the cost functions of other agents in the system, and agents with 'variable' preferences, depending on the cost functions of other agents in the system.

Upon approach the harbour, the process $Capt$ can either move forward under the ship's own propulsion, wait out at sea, or transfer control to a tug (operated by the harbour master). The forward action takes a token representing the ship one location closer to the port. The wait action idles. The transfer action transfers control to a tug. The process $Capt'$ then defers to the tug (idles).

$$Capt = \text{forw} : Capt +_u 1.Capt +_u \text{transf\_contr} : \mathbf{1}. \tag{16}$$

The cost function u encodes the captain's preferences with respect to the environment. The captain wants to hand over to the harbour master as soon as possible (e.g., for insurance reasons) and will transfer control as soon as the harbour master is willing

$$\text{for all } R, C.u(C(R, \text{transf\_contr} : \mathbf{1})) = 0.1. \tag{17}$$

In each location, from the high seas onwards, the captain will have a preference as to whether to continue moving forwards or to wait at that location for a tug. This preference will depend on the context; in heavier seas, the captain will wait for a tug further out. Consider a sequence of locations: $Ocean \to L_1 \to L_2 \to Harbour$. Let s_{loc} refer to the presence of the ship at location loc. We define that in calm seas that the captain is willing to take the ship as far as L_2 and then will wait; that is, for all R, E,

$$\begin{array}{ll} u((R, s_{L_1}), \text{forw} : Capt \times E) = 0.3 & u((R, s_{L_2}), \text{forw} : Capt \times E) = 0.7 \\ u((R, s_{L_1}), 1 : Capt \times E) \quad = 0.7 & u((R, s_{L_2}), 1 : Capt \times E) \quad = 0.3, \end{array} \tag{18}$$

where R doesn't include $rough$. However, in rough seas the captain prefers to wait at both L_1 and L_2; that is, for all E, R and all $loc \in \{L_1, L_2\}$

$$\begin{array}{l} u((R, s_{loc}, rough), \text{forw} : Capt \times E) = 0.7 \\ u((R, s_{loc}, rough), 1 : Capt \times E) \quad = 0.3. \end{array} \tag{19}$$

The harbour master may then choose to wait or to have the tug take control of the ship:

$$Master = 1 : Master +_v \text{acpt\_contr} : Master' \quad Master' = \text{tow} : Master' \tag{20}$$

The harbour master wants to take control of incoming ships as late as possible so as to have the highest throughput (less time spent on each ship means the same number of tugs can get more ships through), but no later than location L_2. If the captain refuses to come further in, the harbour master will compromise and send a tug further out. This calls for a more complex cost function, one that depends on the decisions of the captain

$$v(R, E) = 0.3 \text{ if } u(R, E) < u(R, F) \qquad v(R, E) = 0.7 \text{ if } u(R, E) \geq u(R, F) \tag{21}$$

$$v(R, \text{acpt\_contr} : Master \times Capt) = 0.5, \tag{22}$$

for all R, where $E = \text{wait} : Master \times Capt$ and $F = \text{forw} : Capt \times Master$. The harbour master chooses to wait control if the captain is willing to continue forwards, and accepts control over waiting if the captain is not. Evidently there is game theoretic analysis to be done here, which will be considered in future work.

The trust domain can be evidenced as follows below (using location informally). The goal property is for the tug to attach to the ship (and thereafter to pull it into the harbour). This can be done at either position L_1 or position L_2. In calm seas, the captain's cost judgement will permit the making of the connection at either position L_1 or position L_2, while the harbour master will only permit it at L_2. In stormy seas, however, the captain's cost judgement permits the connection only at L_1, and the harbour master will apply different preferences in order to make the connection at L_1. Note the combination of logical properties (tug connection) and cost properties (intersection of cost boundaries of two different actors).

Contract Choices. We consider a mergers and acquisitions (M&A) deal team. One goal of such a team is to provide valuations of companies that are under consideration for mergers or acquisitions. The task requires access to very confidential details of the company being valued. Often the deal team will contract out specialized aspects of the valuation to external specialists. These contractors may have varying level of security infrastructure. One of the key risks in company valuation is data loss, and the risk is exacerbated when information is shared outside the deal group to external contractors.

Consider a scenario in which the M&A deal team has two potential contractors. The first contractor is a smaller firm that is more specialized, but as it is smaller has a less secure IT infrastructure. The M&A deal team could get a better service (e.g., more efficient, with more accurate valuation, etc.) from using this firm, but incurs more risk of data loss. The second contractor is a large firm that is more generalized, and has a more secure IT infrastructure.

We model each contractor as a process that can provide a valuation (either specialized or general) or idle. The specialized contractor, in addition, has the possibility of leaking information. We make use of the resource $data_{gen}$ to denote data when it is sent to the general contractor, and $data_{spec}$ to denote data when it is sent to the specialized contractor. Both the valuation actions and the leak action require the appropriate data resource, and are not enabled when the resource is absent. These actions ensure mutual exclusion. The general and specific contractors, respectively, are modelled as

$$\begin{aligned} Contr_{gen} &= \text{gen\_val} : Contr'_{gen} + 1 : Contr_{gen} \\ Contr_{spec} &= \text{spec\_val} : Contr'_{spec} + \text{leak} : Contr_{spec} + 1 : Contr_{spec}. \end{aligned} \tag{23}$$

In this example, we are concerned with the choices that the deal team can make and are more interested in *what* the respective contractors can do, rather than *why* they do.

We model the deal team as a process that chooses between the two firms (or idles), and then proceeds to receive a valuation from the chosen contractor.

$$Deal = \text{enbl\_spec} : Deal' +_u \text{enbl\_gen} : Deal' +_u 1 : Deal. \tag{24}$$

The enbl_spec action produces the $data_{spec}$ resource, which enables the specialized contractor to produce a valuation, and the enbl_gen action produces the $data_{gen}$ resource, which enables the generalized contractor to produce a valuation. The whole system is then defined as

$$e, Deal \times Contr_{gen} \times Contr_{spec}. \tag{25}$$

We define the deal team's cost function u as:

$$\begin{aligned} u(e, Contr_{gen} \times Contr_{spec} \times \text{enbl\_spec} : Deal') &= 0.7 \\ u(e, Contr_{gen} \times Contr_{spec} \times \text{enbl\_gen} : Deal') &= 0.3 \\ u(e, Contr_{gen} \times Contr_{spec} \times 1 : Deal) &= 0. \end{aligned} \tag{26}$$

We can consider a trust domain based on the logical property $\langle \text{gen\_val} \rangle \top \vee \langle \text{spec\_val} \rangle \top$, which denotes that an evaluation (specialized or general) is provided. If we were to define a cost bound of 0.5, for example, then the general contractor would be chosen but not the specialized one, as they both provide the required service, but there is too much risk associated with the specialized contractor.

Information Provenance. One issue of particular importance, when making decisions about data sharing arrangements, is the provenance of the evidence used by the contracting company. In particular, the evidence can be categorized as verifiable, or can be taken 'on trust' (perhaps due to a history of positive interaction). The level of evidence required by a decision-maker can be mitigated using technical and social mechanisms, such as use of virtual machines (as in Contract Choices) and external certification (e.g., ISO27000 security certification). We can then consider what risks and costs can occur, given a choice of how much verifiable evidence will be required, and how much can be taken on trust (in the presence of some mitigating mechanisms).

Consider two different contractors, which a bank can choose to employ. Contractor A is safer, and can leaks can only occur in one way; we model this by saying that the $leak_A$ action is only enabled by resource r_1. Contractor D is a little less safe, and its leaks can occur in one of two ways; we model this by saying that the $leak_D$ action is enabled by either of the leak pathways, modelled as resources p_1 and p_2. We then make use of different pieces evidence, which provide different guarantees. We consider two pieces of 'trust' evidence, the first (t_1) that rules out the first type of leaks (modelled by the fact that $p_1 \circ t_1 \uparrow$), and the second (t_2) that rules out the second type of leaks (modelled by the fact that $p_2 \circ t_2 \uparrow$). We also consider a piece of 'verifiable' evidence v, that rules out both types of leaks (modelled by the fact that $p_1 \circ v \uparrow$ and $p_2 \circ v \uparrow$).

We model the bank as a process that chooses between two contractors

$$Bank = B_A +_u B_D \quad B_A = \text{chooseA} : \mathbf{1} \quad B_D = \text{chooseD} : \mathbf{1}. \tag{27}$$

the 'chooseA' (resp. 'chooseD') action creates a token c_A (resp. c_D) that enables contractor A (resp. D) to proceed. We model the contractors as processes that can accept a contract token and then proceed to either provide a leak or possibly leak some data

$$A = \text{acpt}_\text{A}.(\text{report} : A' +_0 \text{leak}_\text{A} : A') \quad D = \text{acpt}_\text{D}.(\text{report} : D' +_0 \text{leak}_\text{D} : D'). \tag{28}$$

The report actions are the same in both cases, but 'leak' actions are different for each process. In contractor A the leak can only occur by the first pathway, which is modelled by the 'leak_A' action being enabled only by the resource p_1. In contractor D, however, the leak can occur by either of the pathways, which is modelled by the '$\mathrm{leak_D}$' action being enabled both by the resource p_1 and by the resource p_2.

We then consider different ways the the bank can value, and make use of, the evidence to which it has access. Consider one possible valuation

$$\begin{array}{c} u(v, B_A \times A) = u(v, B_D \times D) = 0.1 \\ u(t_1, B_A \times A) = u(t_1, B_D \times D) = 0.4 \end{array} \quad u(t_2, B_A \times A) = u(t_2, B_D \times D) = 0.6. \tag{29}$$

Here, the scenario where verifiable evidence can be obtained, to the effect that no leaks will occur (modelled as the fact that the resource v is present), has the least risk, with a cost of 0.1. The scenario where trusted evidence can be obtained to the effect that the first type of leak will not occur (modelled as the fact that the resource t_1 is present) is the next most risky, with a cost of 0.4. The scenario in which trusted evidence can be obtained that the second type of leak will not occur (modelled as the fact that the resource t_2 is present) is most risky, with a cost of 0.6.

Given this valuation we can show that errors can occur, even below the rather generous cost bound of 0.5. As the banker expects the cost of company D at 0.4, in the presence of trust guarantee t_1, it is possible to show that at least one of the two leak actions occurs within the cost bounds

$$e, B_A +_u B_D \not\models_{C_\emptyset} (p \wedge \phi) \multimap ([\text{chooseA}]\,[< 0.5][\text{leak}_\text{A}]\bot \wedge [\text{chooseD}]\,[< 0.5][\text{leak}_\text{D}]\bot), \tag{30}$$

where p is a proposition that denotes the presence of resource t_1 and ϕ limits the processes under consideration, to be put in parallel with, to either contractor A or contractor D, modulo bisimulation.

Consider a slightly more nuanced valuation of the provenance of pieces of evidence. Perhaps, through experience, the bank comes to realize that the first trusted property t_1 is not quite so effective an indicator with respect to the second contractor. We can then define a different cost function, v, which assigns more risk to t_1 for the contractor D

$$\begin{array}{cc} v(v, B_D \times A) = u'(v, B_D \times D) = 0.1 & v(t_1, B_D \times A) = 0.4 \\ v(t_1, B_D \times D) = 0.5 & v(t_2, B_D \times A) = u'(t_2, B_D \times D) = 0.6. \end{array} \tag{31}$$

With this new costing, no leaks can occur within the given cost bounds:

$$e, B_A +_v B_D \models_{C_\emptyset} (p \wedge \phi) \multimap ([\text{chooseA}]\,[< 0.5][\text{leak}_\text{A}]\bot \wedge [\text{chooseD}]\,[< 0.5][\text{leak}_\text{D}]\bot). \tag{32}$$

7 Further Work

Further work includes extending the calculus to include probabilistic evolution and expected cost [13], thereby enriching the notion of trust domain with the ability to accommodate stochastic environments. Although we have given some quite rich examples, it remains for us to explore a meta-theoretically more systematic collection of compositional and structural properties. This work begins with an exploration of the transitivity of trust domains — suppose that agent B is within agent A's trust domain, and that agent C is within B's trust domain: how does A decide whether to trust C (e.g., in supply chains, or in outsourcing)? — and would also include a more careful distinction between intensional (as in the harbour) and extensional (as in M&A) formulations of trust domains. This work would also consider how to do substitution within trust domains in a way that preserved the selected set of contexts. One approach would be to make use of Theorem 2, which would permit us to substitute bisimilar agents and preserve the goal properties that define the trust domain.

It would also be interesting to consider whether the (pre)sheaf-theoretic semantics considered by Winskel [12] can be adapted to our calculus.

References

1. M. Abadi. Logic in access control. In *Proc. LICS 2003*, 228–233. IEEE , 2003.
2. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Prog. Langs. Sys.*, 15(4):706–734, 1993.
3. G. Anderson, M. Collinson, and D. Pym. Utility-based Decision-making in Distributed Systems Modelling. In *Proc. 14th TARK*, Burkhard C. Schipper (editor), Chennai, 2013. Computing Research Repository (CoRR): http://arxiv.org/corr/home. ISBN: 978-0-615-74716-3.
4. J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. CUP, 1997.
5. A. Beautement, R. Coles, J. Griffin, C. Ioannidis, B. Monahan, D. P. C. Author), A. Sasse, and M. Wonham. Modelling the Human and Technological Costs and Benefits of USB Memory Stick Security. In M. E. Johnson, editor, *Managing Information Risk and the Economics of Security*, 141–163. Springer, 2008.
6. M. Collinson, B. Monahan, and D. Pym. *A Discipline of Mathematical Systems Modelling*. College Publications, 2012.
7. G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley; 3rd edition, 2000.
8. M. Hennessy. A calculus for costed computations. *Logical Methods in Computer Science* 7(1), paper 9, 2011. DOI: 10.2168/LMCS-7(1:7)2011
9. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.
10. M. Hennessy and G. Plotkin. On observing nondeterminism and concurrency. In *Proceedings of the 7th ICALP*. LNCS 85: 299–308, 1980.
11. W. Jamroga. A temporal logic for Markov chains. In *Proc. AAMAS 2008*, 607–704. ACM Digital Library, 2008.
12. A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.
13. R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. Wiley, 1976.
14. J. McCarthy. Formalizing context. In *IJCAI*, pages 555–562, 1993.
15. R. Milner. Calculi for synchrony and asynchrony. *Theoret. Comp. Sci.*, 25(3):267–310, 1983.
16. R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.
17. P. O'Hearn and D. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
18. S. Ramchurn, D. Huynh, and N. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.
19. J. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc. 17th LICS*, 55–74, IEEE, 2002.
20. C. Tofts. Processes with probability, priority and time. *Formal Aspects of Computing*, 6(5):536–564, 1994.