Algebra and logic for access control

Matthew Collinson, David Pym

DOI 10.1007/s00165-009-0107-x
BCS © 2009
Formal Aspects of Computing (2010) 22: 83–104

Formal Aspects of Computing

Algebra and logic for access control

Matthew Collinson and David Pym

Hewlett-Packard Laboratories, Long Down Avenue, Stoke Gifford, Bristol BS34 8QZ, UK.
E-mail: matthew.collinson@hp.com; david.pym@hp.com

Abstract. The access control problem in computer security is fundamentally concerned with the ability of system entities to see, make use of, or alter various system resources. We provide a mathematical framework for modelling and reasoning about (distributed) systems with access control. This is based on a calculus of resources and processes together with a Hennessy–Milner-style modal logic, based on the connectives of bunched logic, for which an appropriate correspondence theorem obtains. As a consequence we get a consistent account of both operational behaviour and logical reasoning for systems with access control features. In particular, we are able to introduce a process combinator that describes, as a form of concurrent composition, the action of one agent in the role of another, and provide a logical characterization of this operator via a modality 'says'. We give a range of examples, including analyses of co-signing, roles, and chains of trust, which illustrates the utility of our mathematical framework.

Keywords: Access control, Role, Systems modelling, Process calculus, Resource, Logic

1. Introduction

Access control is one of the fundamental issues in information security.

In computer systems of almost all levels of scale, certain behaviours will be desirable and certain others undesirable. A great many of these behaviours involve the ability of entities (users, programs, processes) in the system to access some other entity (resource, program, process). The denial of undesirable, and the permission of desirable behaviours is thus reduced to the *access control problem*.

Access control is, of course, one of the main issues in computer security, with work in the area extending back over many years (for example by Lampson [Lam71] and by Saltzer and Schroeder [SS75]) and gaining ever greater prominence until the present day. One strand of work in this area concerns the development of logical languages, sometimes called *security languages*, for reasoning about and making access control decisions, as exemplified in work by Abadi et al. [Aba03, ABL93, LAB92] and DeTreville [DeT02].

Important challenges for logical security languages are the representation of *co-signing*, of *roles* and of *chains-of-trust*. The security language in [ABL93] introduces novel connectives for co-sigining and roles and this is sufficient to allow formal inferences to be made, in particular for chains-of-trust. However, the mathematical semantics of such languages, where they exist, are often not transparently related to operational behaviour. Furthermore, there is a hint in [ABL93] that such connectives correspond to concurrent behaviour—as indeed they must.

The lack of a suitable semantics means that, given an existing system in which we care about access control, it is difficult to see how most existing security languages can be used to capture access control behaviour *in a*

Correspondence and offprint requests to: M. Collinson and D. Pym, E-mail: matthew.collinson@hp.com; david.pym@hp.com

provably sound way. Of course, by design, access-control systems can be implemented which closely conform with policies and protocols embodied in security languages.

In this paper, we show that process calculus can be used to give a semantics for a security language, thus giving a meaningful account of (suitably defined) connectives for security languages, including both co-signing and roles. Chain-of-trust arguments arise naturally from the underlying semantics.

Our ultimate concern is to produce models of real systems in which security is critical [BCG08, BGS08, BBC08]. Thus the models we are concerned with will often be *descriptive*, in the sense that they describe the behaviour of existing or imagined systems. We therefore need to be able to model systems with both secure and insecure behaviours. This is in contrast to much formal work in security, where authors are often concerned primarily with *normative* models, in which only desired, secure behaviour is allowed. We retain the ability to place normative constraints upon system behaviour (in models) through the use of the logical security language.

The modelling framework we propose is based on resources, processes, and modal bunched logic developed by Pym, Tofts, and Collinson [PT06, PT07, CPT07, CP09]. The present paper presents an application of that earlier work to give an account of security languages and security-related problems. The basic idea is that resources R and processes, in the sense of (synchronous) process algebra, E co-evolve,

$$R, E \xrightarrow{a} R', E',$$

according to the specification of a partial function, $\mu : (a, R) \mapsto R'$, that determines how an action a evolves E to E' and R to R'. The base case of the operational semantics is given by action prefix:

$$\frac{}{R, a : E \xrightarrow{a} R', E} \quad (\mu(a, R) = R').$$

The theory of this calculus of resources and processes (**SCRP**) has been explored in detail in [PT06, PT07, CPT07, CP09], but a brief review of the process definitions is included.

In the security literature, the entities which act within systems are often referred to as *principals*. A key step for us is the representation of principals as processes. A similar approach has been taken by a number of authors, particularly regarding security protocols, see for example [AG97, SMR06, Sch96].

The **SCRP** calculus comes with a *Hennessy–Milner (modal) logic* [HM85] called **MBI** for the specification and verification of properties of systems (resource-process states). **MBI** is simultaneously a *resource logic* in the sense of bunched logic, **BI** [OP99, Pym99, POY04, Pym02], and its cousin Separation Logic [IO01, OHe07, Rey02].

A key aspect of **SCRP** and **MBI** is the relationship between concurrent composition and multiplicative ('separating') conjunction. This exploits an underlying resource semantics, based on that of BI, in which resources carry monoidal structure:

$$R, E \models \phi_1 * \phi_2 \quad \text{iff} \quad R_1, E_1 \models \phi_1 \quad \text{and} \quad R_2, E_2 \models \phi_2$$

for some R_1, R_2 and E_1, E_2 such that $R = R_1 \circ R_2$ and $R, E \approx R, E_1 \times E_2$.

In our setting, the multiplicative conjunction can be used, for example, to describe a co-signing requirement for resource access. The principals who must co-sign are E_1 and E_2. The resources R_1 and R_2 represent, respectively, E_1 and E_2's separate access rights, together with the shared resource to be accessed. The composite resource $R_1 \circ R_2$ represents the appropriate combination of access rights and shared resource. Each ϕ_i can be used as (a proxy for) some certificate that each E_i holds (in order, say, to sign). A more detailed illustration of this is found in Example 6.3 in Sect. 6.

In order to describe the roles of principals of the form 'E in the role of F', we introduce an additional binary process constructor, $\propto$, into SCRP. Thus, a resource-process state

$$R, E \propto F.$$

represents a principal E in the role F, together with resources R. Note that the role F is itself represented by a process and that it is intended to have fewer abilities than E. Along with this construct comes a logical modality, given by the following forcing definition:

$$R, G \models \{E\}\phi \quad \text{iff} \quad \exists F \text{ s.t. } R, G \approx R, E \propto F \text{ and } F \lessapprox E \text{ and } R, F \models \phi.$$

Here $\lessapprox$ (respectively $\approx$) is the notion of simulation (respectively bisimulation). Thus a logical assertion $\{E\}\phi$, read '*E says* ϕ', is used to describe properties that may hold of some role of a process, rather than the process itself: for example, often the reduced 'user' role of some 'administrator' process has additional safety properties. Example 6.6 in Sect. 6 is of this kind.

The intention of this work follows the tradition of using process calculus as a modelling tool. This paper is intended to serve as a foundation for the modelling of certain existing security situations. In practice, this will take place in the Demos2k tool [Bir79, Dem] and a variant thereof (*LD2k* , [CMP08a, CMP08c]), which is particularly tailored towards event-modelling and performance analysis in distributed systems, and which is closely related to the calculus we have presented here.

There are a number of dedicated calculi for the formal analysis and specification of access control systems. In contrast, the calculus we present here is designed for *modelling* systems that may include access control features; specifically, a notion of role. Perhaps the most successful such calculus of recent years has been RBAC [CFS96], and the analysis of this has been refined over the years; see, for example, [KYM06].

There has been a recent upsurge of interest in access control calculi for situations in which access decisions may change over time; for example, as permissions are granted or revoked [BN07, GRS04]. The semantics of such calculi often features notions of state that capture just the information required for access decisions (for example, current permissions). Model-checking techniques can be used to reason about intended response to requests; for example, as in [GRS04].

There have been some interesting recent developments in access control and security policy design. Specifically, a degree of attention is being given to 'separation of duty' in security policy, in which it is mandated that some task must be undertaken by two (or more) separate entities. This is not new [SS75], but there have recently been attempts to construct logical calculi for such policies. In [LW08], such a calculus is presented that contains a connective $\otimes$ for expressing policies with separation of duty. This connective is given a semantics that makes it essentially identical to the 'separating conjunction' $*$ of Separation Logic [IO01, Rey02]. This in turn is a particular instance of the multiplicative conjunction $*$ of **BI** [OP99, Pym99, POY04, Pym02] and so a close relative of the connective $*$ used in the present work. Indeed, it is noted in Example 6.3 below that separation concerns can easily be represented in the logic **MBIa** that accompanies the process calculus **SCRP**. It seems that the collection of connectives employed in [LW08] is more closely related to **BI** than to Linear Logic.

Section 2 contains a brief review of SCRP. Section 3 describes how this is modified to deal with compound principals that include roles. Section 5 gives the associated modal resource logic and some basic results. Section 6 gives a range of examples:

- First, a basic example in which access is performed by a resource guard on behalf of a general agent;
- Second, a similar example, in which a guard authorizes an agent to access a resource;
- Third, a joint-access request, in which two agents must both request access, and in so doing must combine their permission resources;
- Fourth, exclusive access, a variant of joint access, in which two agents may mutually exclusively access a resource;
- Fifth, authorization by delegation, in which a guard must consult a second authority within which resides the access control list, so establishing a chain of trust;
- Sixth, reduction to role, in which we have an agent together with a role for that agent which has reduced access rights;
- Last, modelling access with assertion-based control, in which access control decisions are based on a logical language rather than just ACLs (cf. Binder [Aba03]), giving another example of a chain of trust, formed by the trust agents have in others' public statements.

Section 7 describes changes that must be made to the set-up when additional logical power is required in the specification language. Finally, Sect. 8 contains a discussion of open problems and further directions that we are pursuing.

2. Resources and processes

Many access control systems live in an environment in which significant events occur simultaneously. Moreover, events of the access system itself may occur concurrently and there may be complex interactions between all parts of the system and the environment. A modelling framework that describes such systems and their environment must be able to capture concurrency in a natural way.

Process calculus, like Milner's CCS [Mil80, Mil89], or the more general synchronous calculus SCCS [Mil83], is an elegant methodology for dealing with such situations. It provides a precise framework for the construction

of models. In particular, it has the important property of *compositionality*: the description of a large system is constructed from those of component subsystems.

SCRP is a form of synchronous process calculus. In contrast to standard process calculi, it has an explicit treatment of resource. **SCRP** was introduced in the papers [PT06, PT07, CPT07, CP09]. The calculus presented in this paper is a closely related variant of those calculi.

In this paper, we often use partial functions, writing $exp \downarrow$ and $exp \uparrow$ to mean that an expression exp is, respectively, defined or undefined. We also make use of Kleene equality between expressions: the left-hand side of an equality, $lexp \simeq rexp$, is defined if and only if the right-hand side is defined, and when defined they are equal.

Mild constraints are placed upon the type of resource treated. A *resource monoid*, modelling the composition and comparison of resource elements [OP99, Pym02, POY04], is a structure

$$\mathbf{R} = (\mathbf{R}, \circ, \mathbf{e}, \sqsubseteq) .$$

We do not use a separate notation to distinguish the carrier set $\mathbf{R}$ from the structure. We reserve the letters R, S, T, U, V for resources. The structure has a preorder $\sqsubseteq$, a partial, binary composition $\circ$, and has a distinguished element $\mathbf{e}$. The operation $\circ$ satisfies monoid associativity and commutativity axioms up to Kleene equality. The unit of $\circ$ is $\mathbf{e}$. Composition with this unit is always defined. Therefore, the structure satisfies the unit axiom for a commutative monoid up to actual equality. Resource monoids are further required to satisfy the *bifunctoriality condition*:

if $R \sqsubseteq R'$ and $S \sqsubseteq S'$ and $R' \circ S' \downarrow$ then $R \circ S \downarrow$ and $R \circ S \sqsubseteq R' \circ S'$

for all R, R', S, S' in $\mathbf{R}$. For the purposes of this paper, the preorder $\sqsubseteq$ is always taken to be the equality relation.

We assume a commutative monoid, Act, of *actions*. Just as in standard process algebra, these actions correspond to the events of a system. We reserve the letters a, b, c, ... for actions. Composition is written by juxtaposition and the unit action is written 1. For the purposes of this paper we assume that the action monoid is generated freely from *atoms*, for which we reserve the letter α.

Assume a (partial) function, called a *modification*, $\mu : \mathsf{Act} \times \mathbf{R} \longrightarrow \mathbf{R}$, satisfying two *coherence* conditions:

1. $\mu(1, R) = R$ for all $R \in \mathbf{R}$;
2. if $\mu(a, R)$, $\mu(b, S)$ and $R \circ S$ are all defined then the Kleene equality $\mu(ab, R \circ S) \simeq \mu(a, R) \circ \mu(b, S)$ holds.

We define a total operation called *hiding*, that takes any resource R and any action a and produces an action $\nu R.a$. Any action a may be written uniquely (up to re-ordering) as a product $a = \prod\{\alpha_i \mid i \in I\}$ for some finite set I. Then we may take

$$\nu R.a = \prod\{\alpha_i \mid i \in I \ \&\ \mu(\alpha_i, R) \uparrow\}. \tag{1}$$

Recall that the product of an empty set of actions gives the identity action.

There are five basic forms of process in **SCRP**: prefix, sum, product, hiding, constant. The letters A, B, C, D, E, F, G, H are reserved for processes. A *state* consists of a resource and a process. *Operational behaviour* is given by transitions (binary relations) labelled by actions on the set of states. We detail the forms of process and their operational behaviour within states below. The definition constitutes a structural operational semantics [Plo04].

A *prefix* process is of the form $a : E$ where E is any process and a is any action. The operational rule for this is

$$\frac{}{R,\, a : E \xrightarrow{a} \mu(a, R),\, E} \quad (\mu(a, R) \downarrow)$$

where R is any resource. When $\mu(a, R)$ is defined we say that a is *enabled at* R.

A *sum* is of the form $\sum_{i \in I} E_i$, where I is an arbitrary index set and each E_i is a process. We often use the infix notation $E + F$ when the cardinality of the index set is 2. The rule

$$\frac{R,\, E_i \xrightarrow{a} R',\, E_i'}{R,\, \sum_{i \in I} E_i \xrightarrow{a} R',\, E_i'}$$

gives the operational behaviour for sums. The *zero* process, $\mathbf{0}$, is the special case of a sum that arises when the index set I is empty. A state with $\mathbf{0}$ as its process component makes no state transitions.

A (*synchronous*) *product* is of the form $E \times F$, where E and F are processes. The rule

$$\frac{R, E \xrightarrow{a} R', E' \qquad S, F \xrightarrow{b} S', F'}{R \circ S, E \times F \xrightarrow{ab} R' \circ S', E' \times F'} \quad (R \circ S \downarrow)$$

describes the evolution of states formed from product processes. The idea is that the two component processes should bring together their resources in order to agree a simultaneous step forward.

A *hiding* (or just *hide*) is of the form $\nu R.E$ where E is a process and R is a resource. The operational rule is

$$\frac{R \circ S, E \xrightarrow{a} R' \circ S', E'}{R, \nu S.E \xrightarrow{\nu S.a} R', \nu S'.E'} \quad (\mu(\nu S.a, R) = R').$$

The idea of hiding is that the process $\nu S.E$ carries private, local resource S that is hidden from external processes, and that it may use this resource to evolve into a new process $\nu S'.E'$. Furthermore, the action $\nu S.a$ through which this happens does not exhibit the atomic actions enabled by S.

A *constant* is defined through the use of a recursive definition. They are a way of introducing recursive process terms into the calculus. An alternative is to use fixed points, as described in [PT06, PT07, CPT07, CP09]. A recursive definition takes the form

$$\begin{array}{l} X_1 := E_1 \\ \quad \vdots \qquad \vdots \\ X_n := E_n \end{array}$$

for some integer n, where each X_i is a *process variable* and each E_i may contain any of the $X_1, \ldots X_n$ but no other process variables. We usually write a tuple of processes as E and the ith component as E_i. Each of these systems of simultaneous equations uniquely specifies a canonical (least) solution, that is, a sequence of processes C satisfying the equations. Each C_i is a *constant*. We often write $\mathsf{X} := \mathsf{E}$ for the tuple of specifications, and $\mathsf{C} := \mathsf{E}$ to associate particular constant names to the canonical solution. Suppose that each C_i is C_i. The operational rule for such a constant C_i is

$$\frac{R, E_i[\mathsf{C}/\mathsf{E}] \xrightarrow{a} R', G'}{R, C_i \xrightarrow{a} R', G'},$$

where E are the defining expressions for the sequence C. For example, the special process $\mathbf{1}$, that can only tick, is defined by the equation $X := 1 : X$. We usually abbreviate definition by constants by writing the names of the constants rather than variables, so for example $\mathbf{1} := 1 : \mathbf{1}$.

3. An extended calculus for principals

In this section, we present a process calculus **ACCRP** which is tailored to describing systems formed from principals. Thus the kinds of principals which arise in access control problems in computer security are to be described as process terms in **SCRP**. Our approach has some common ground with work that uses process calculus for the formal analysis of security protocols [AG97, RSG01, Sch96], but this will not be our main concern.

The tailoring of the calculus reflects the fact that there are certain compound principals which occur time-and-again in the security literature. In particular, there are *conjunctions* of principals, which may perform some access just if both principals do so together—we use the synchronous product (in a particular way) for such processes, as in Example 6.3 below. There are also principals which are formed by adopting *roles* with fewer capabilities. We introduce a dedicated new connective $\propto$ for roles. Further discussions of compound principals may be found in the papers by Abadi et al. [Aba03, ABL93, LAB92] which have strongly influenced the present work.

To form the calculus **ACCRP** the grammar of **SCRP** is extended with the role constructor, so that

$$E ::= \cdots \mid E \propto E.$$

The operational rule for the constructor introducing a principal 'E in the role of F', where F is a principal with reduced capacity (at a reduced resource), is

$$\frac{R, F \xrightarrow{a} R', F' \quad R \circ S, E \xrightarrow{a} R' \circ S', E'}{R \circ S, E \propto F \xrightarrow{a} R' \circ S', E' \propto F'} \quad (F \lessapprox E) \tag{2}$$

where $\lessapprox$ is defined below. As is usual the operational semantics describes all possible traces of states that systems may travel. Examples of traces of events in access control decisions are given in Sect. 6, the simplest being those of Examples 6.1 and 6.2. An example of the use of the role constructor is given in Example 6.6.

We define notions of equivalence and inequivalence for states and processes. We define

$$R, E \lessapprox S, F \tag{3}$$

if $R = S$ and, whenever $R, E \xrightarrow{a} R', E'$, there is some F' with $R, F \xrightarrow{a} R', F'$ and $R', E' \lessapprox R', F'$. In such circumstances, we say that S, F *simulates* R, E.

We write $R, E \approx R, F$ iff $R, E \lessapprox R, F$ and $R, F \lessapprox R, E$ both hold, and say that R, E is *(locally) bisimilar* to R, F. We write $E \lessapprox F$ or $E \approx F$, when, respectively, $R, E \lessapprox R, F$ or $R, E \approx R, F$ for all resources R.

We note that $\lessapprox$ is defined by mutual recursion with processes since the role constructor uses $\lessapprox$ as a side-condition. This is somewhat unusual for a process calculus. In order to make this work, we do not allow process variables to occur inside role constructors (note also that they are not required conceptually). That is, we restrict the form of the recursive equations used to define constants.

We introduce a syntactic complexity measure, $h(E)$, of the height of the tower of $\propto$ connectives used to define each process E:

$$\begin{aligned} h(X) &= 0 \\ h(a : E) &= h(E) \\ h(E \times F) &= \max\{h(E), h(F)\} \\ h(E \propto F) &= \max\{h(E), h(F)\} + 1 \\ h(\sum_{i \in I} E_i) &= \max(\{h(E_i) \mid i \in I\} \cup \{0\}) \\ h(C_i) &= \max\{h(E_1), \ldots, h(E_n)\} \end{aligned}$$

where, in the final clause above, $E_1, \ldots, E_n$ are the components of E defining the constants $C_1, \ldots, C_n$ in $\mathsf{C} := \mathsf{E}$. Notice that the processes in the side-conditions of the role rule have lower complexity than the role that is the source of the transition.

Transitions do not increase process height: the proof is the evident induction on derivations.

Lemma 3.1 *If* $R, E \xrightarrow{a} R', E'$, *then* $h(E') \leq h(E)$.

Therefore the definition of simulation can be re-stated in a stratified way. Thus, to show $R, E \lessapprox R, F$ we need only compare transitions into processes in the same or lower strata. Hence the mutual recursion between transitions and simulation is well-defined.

4. Algebraic and dynamical properties

A number of simple properties of **ACCRP**-systems hold. As these systems are defined through the use of structural operational semantics, a critical proof technique is the use of induction on the structure of derivations of state-transitions. The first property to observe is that the evolution of resource is completely determined by the choice of action.

Lemma 4.1 *If* $R, E \xrightarrow{a} R', E'$ *then* $R' = \mu(a, R)$.

Proof Induction on the structure of derivations. The base case is where E is a prefix process; then, for any R, R', E' as in the statement of the lemma, we have $R' = \mu(a, R)$. The induction hypothesis is that all shorter derivations satisfy the statement of the theorem. Take, for example, the role definition in rule (2) above. The induction hypothesis gives $R' \circ S' = \mu(a, R \circ S)$ using the right-hand premise, but this is precisely the required property for the conclusion. We omit the other cases, as they are equally straightforward, but note that the product

case relies upon the second coherence condition on modifications, and that the hiding case uses the side-condition on the hiding rule. □

The local bisimulation relation is a congruence.

Proposition 4.1 *The relation $\approx$ on processes is a congruence for the process constructors. It is an equivalence relation, and, in particular, for all E, F, G:*

1. *if $E \approx F$ then $E \propto G \approx F \propto G$, and*
2. *if $E \approx F$ then $G \propto E \approx G \propto F$.*

Proof The proof of all parts of this follow from the definition of $\approx$ and by applying the standard methods for proving that bisimulation is a congruence, see [Mil83] for example. For example, consider the final point in this lemma. Suppose that $E \approx F$ and that we have a transition of $G \propto E$. This must come from some derivation ending with a role rule

$$(E \lessapprox G)\frac{R, E \xrightarrow{a} R', E' \quad R \circ S, G \xrightarrow{a} R' \circ S', G'}{R \circ S, G \propto E \xrightarrow{a} R' \circ S', G' \propto E'},$$

but then we may replace this final rule with

$$(F \lessapprox G)\frac{R, F \xrightarrow{a} R', F' \quad R \circ S, G \xrightarrow{a} R' \circ S', G'}{R \circ S, G \propto F \xrightarrow{a} R' \circ S', G' \propto F'}$$

with $R', E' \approx R', F'$, since $F \approx E \lessapprox G$. Thus the set of pairs $\{\langle (R, G \propto E), (R, G \propto F)\rangle \mid R, E \approx R, F\}$ is closed under transitions of the left-hand component. Similarly, it is closed under transitions of the right-hand component. □

Interestingly, the second congruence statement above does not hold for (unidirectional) simulation because of the asymmetry of the role rule. Processes satisfy a number of other equalities and inequalities, including the following:

Proposition 4.2 *The constructor $\propto$ has the following properties with respect to bisimulation:*

1. $E \propto F \lessapprox E$;
2. $E \propto E \approx E$;
3. *If* $G \lessapprox E$ *then* $G \lessapprox E \propto G$;
4. *If* $E \lessapprox F$ *then* $E \propto G \lessapprox F \propto G$;
5. $(E_1 \propto F_1) \times (E_2 \propto F_2) \lessapprox (E_1 \times E_2) \propto (F_1 \times F_2)$;
6. $(E_1 \propto F_1) \times (E_2 \propto F_2) \approx (E_1 \times E_2) \propto (F_1 \times F_2)$ *provided* $F_1 \times F_2 \lessapprox E_1 \times E_2$ *implies* $F_1 \lessapprox E_1$ *and* $F_2 \lessapprox E_2$;
7. $E \propto (F \propto G) \lessapprox (E \propto F) \propto G$ *provided* $F \lessapprox E$ *and* $G \lessapprox E \propto F$;
8. $(E \propto F) \propto G \lessapprox E \propto (F \propto G)$ *provided* $G \lessapprox F$.

Proof Again, the proofs of these points are direct uses of the definition of simulation, and we omit most of them. For example, consider the fifth point. Suppose that we have some transition $T, (E_1 \propto F_1) \times (E_2 \propto F_2) \xrightarrow{a_1 a_2} T', (E_1' \propto F_1') \times (E_2' \propto F_2')$. This must come from some derivation ending with a product rule, with premisses each ending with a role rule. Thus the resources are of the form $T = (R_1 \circ S_1) \circ (R_2 \circ S_2)$ and $T' = (R_1' \circ S_1') \circ (R_2' \circ S_2')$ and we have some $S_i, F_i \xrightarrow{a_i} S_i', F_i'$ and $R_i \circ S_i, E_i \xrightarrow{a_i} R_i' \circ S_i', E_i'$ and $S_i, F_i \xrightarrow{a_i} S_i', F_i'$ for $i = 1, 2$. We then have the products $T, E_1 \times E_2 \xrightarrow{a_1 a_2} T', E_1' \times E_2'$ and $S_1 \circ S_2, F_1 \times F_2 \xrightarrow{a_1 a_2} S_1' \circ S_2', F_1' \times F_2'$. Hence we have $T, (E_1 \times E_2) \propto (F_1 \times F_2) \xrightarrow{a_1 a_2} T', (E_1' \times E_2') \propto (F_1' \times F_2')$, as required. □

The above results tells us that we have a system that formally reconstructs the following natural properties of roles: any agent acting in one of its roles is no more powerful (has fewer or equal capabilities) than the original agent; an agent E in the role E is as powerful as the agent E; if E is no more powerful than F then every role of E is no more powerful than the corresponding role of F; a synchronous product of agents acting in roles is no more powerful than the product of the agents in the role of the product of their roles (and under certain circumstances this extends to equality); an agent E in the role of 'F in the role G' is equally powerful as the agent 'E in the role F' in the role G, provided that no roles of greater power are used. The presence of such properties is extremely important for the logic of the next section.

5. Logic

In Hennessy–Milner logic [HM85, Sti01] a forcing relation is used to relate CCS processes to assertions of their properties, with the judgement $E \models \phi$ being read as 'process E has property ϕ'. The language of propositions typically contains the classical propositional connectives, $\wedge$, $\vee$, and $\neg$, together with the classical action modalities $\langle a \rangle$ and $[a]$. In our setting, the corresponding judgement, $R, E \models \phi$, says that property ϕ holds of process E in the presence of resources R; that is, of the system model R, E.

In our synchronous setting, we are able to provide an analysis of various structural aspects of processes. In particular, we obtain essentially the following logical characterization of the synchronous product: $R, E \models \phi_1 * \phi_2$ iff $R_1, E_1 \models \phi_1$ and $R_2, E_2 \models \phi_2$ for some R_1, R_2 and E_1, E_2 such that $R = R_1 \circ R_2$ and $R, E \approx R, E_1 \times E_2$. This characterization stands in contrast to the situation for CCS [Sti01], in which $E \mid F \models \phi$ iff $E \models \phi/F$ where the definition of ϕ/F involves 'distributing the process though the formula'. We also obtain a characterization of hiding in terms of the multiplicative existential quantifier (see below), which exploits also the presence of resources in the forcing judgement. Both of these structural characterizations are exploited in the access-control examples presented in Sect. 6.

In general, the logic MBI, introduced in [PT06, PT07, CPT07], admits a range of connectives and quantifiers, including multiplicative modalities. For the present paper, however, we introduce a logic **MBIa** for reasoning about properties of **ACCRP** systems from which we omit, for technical reasons, the multiplicative implication, $-\!\!*$, and the multiplicative modalities. This logic is able to express some interesting aspects of access control properties.

Assume a countable set, ActVar of *action variables*, ranged over by x, and a constant symbol a for each action a of **ACCRP**. Let $\mathsf{A} = \mathsf{ActVar} \cup \mathsf{Act}$ and let a range over this set. We assume a given set of *relation* symbols on actions, each with a given arity. Atomic formulae φ consist of all instances of relations, that is, if p is a relation symbol of arity n and $\mathsf{a}_1, \ldots, \mathsf{a}_n \in \mathsf{A}$, then $p(\mathsf{a}_1, \ldots, \mathsf{a}_n)$ is an atomic formula.

The formulae of the language **MBIa** are defined by the grammar

$$\phi := \bot \mid \top \mid \varphi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid I \mid \phi * \phi \mid \langle \mathsf{a} \rangle \phi \mid [\mathsf{a}]\phi \mid \{E\}\phi \mid \exists\, x.\phi \mid \exists_{\nu}\, x.\phi \mid \forall\, x.\phi \mid \forall_{\nu}\, x.\phi \ ,$$

for $\mathsf{a} \in \mathsf{A}$ and processes E. The connectives $\top$, $\neg$, $\vee$, $\wedge$ and $\to$ are the connectives true, negation, disjunction, conjunction and implication of classical logic. The connectives $\langle \mathsf{a} \rangle \phi$ and $[\mathsf{a}]\phi$ are classical modal connectives, intended to express properties that hold after, respectively, some or any, instance of a from a state. The connectives I and $*$ are known as the *unit* and *multiplicative conjunction*, often pronounced *star*. The connective $\exists$ is classical existential quantification. The connective $\exists_{\nu}$ is the *multiplicative existential* quantifier. The modality $\{E\}\phi$ is read E *says* ϕ, and is intended to express the fact that a process may indirectly witness a fact through the use of a role of E. The *sentences* are just the formulae without free variables. For any formula ϕ, let $\phi[a_1/x_n, \ldots, a_n/x_n]$ be the formula formed by replacing each occurrence of each variable x_i by the term a_i. More generally, one may want to allow function symbols on actions, compound action terms, equalities between such terms and further logical operators.

A valuation $\mathcal{V}$ for the language above is fixed by choosing an $(n+1)$-ary relation $\mathcal{V}(p)$ (between n actions and one state) for each relation symbol p of arity n. Each set $\mathcal{V}(p)$ must be closed under the relation $\approx$. An *assignment*, η, is a function from ActVar to Act. For any η, let $\eta[a/x]$ be the assignment that is identical to η, except that $\eta(x) = a$. A valuation is extended to an interpretation of formulae by means of a forcing relation $\vDash$, as in Fig. 1.

All of the clauses, except for $\{E\}$, in Fig. 1 have been previously studied in the context of **SCRP**. In particular, notice how $*$ specifies that a state is (up to bisimilarity) a synchronous product with suitably sub-divided resource, and how $\exists_{\nu}$ specifies a hiding. In Example 6.3 below $*$ is used to characterize joint-access requests. In Example 6.5 $\exists_{\nu}$ is used to characterize an agent which may grant access to others, but does so by first consulting a private resource. In Example 6.7 $\exists_{\nu}$ is used to characterze an agent which uses private beliefs to grant access in a system where there are public statements by agents about which accesses should be granted. The 'says' modality $\{E\}\phi$ specifies a state that is a role (up to bisimilarity) and that the role it takes satisfies ϕ. Thus in Fig. 1 the role F of E witnesses ϕ for G (even when E itself does not). Example 6.6 demonstrates the use of the role constructor and 'says'.

The Hennessy–Milner-style result given below holds. This shows that algebraically equivalent processes satisfy the same logical specifications.

$R, G, \eta \models \varphi(x_1, \ldots x_n)$ iff $(\eta(x_1), \ldots, \eta(x_n), (R, G)) \in \mathcal{V}(\varphi)$

$R, E, \eta \models \bot$ never

$R, E, \eta \models \top$ always

$R, E, \eta \models I$ iff $R = \mathbf{e}$ and $R, E \approx R, \mathbf{1}$

$R, E, \eta \models \neg\phi$ iff $R, E, \eta \models \phi$ does not hold

$R, E, \eta \models \phi \wedge \psi$ iff $R, E, \eta \models \phi$ and $R, E, \eta \models \psi$

$R, E, \eta \models \phi \vee \psi$ iff $R, E, \eta \models \phi$ or $R, E, \eta \models \psi$

$R, E, \eta \models \phi \rightarrow \psi$ iff $R, E, \eta \models \phi$ implies $R, E, \eta \models \psi$

$R, E, \eta \models \phi_1 * \phi_2$ iff $\exists R_1, R_2, E_1, E_2.\ R = R_1 \circ R_2\ ,\ R, E \approx R, E_1 \times E_2\ ,\ R_1, E_1, \eta \models \phi_1\ ,\ R_2, E_2, \eta \models \phi_2$

$R, E, \eta \models [\mathsf{a}]\phi$ iff $\forall R', E'.\ R, E \xrightarrow{\mathsf{a}} R', E'$ implies $R', E', \eta \models \phi$

$R, E, \eta \models \langle \mathsf{a} \rangle \phi$ iff $\exists R', E'.\ R, E \xrightarrow{\mathsf{a}} R', E'$ and $R', E', \eta \models \phi$

$R, G, \eta \models \{E\}\phi$ iff $\exists F.\ R, G \approx R, E \propto F$ and $F \lessapprox E$ and $R, F, \eta \models \phi$

$R, E, \eta \models \exists x.\phi$ iff $\exists a.\ R, E, \eta \models \phi[a/x]$

$R, E, \eta \models \forall x.\phi$ iff $\forall a.\ R, E, \eta \models \phi[a/x]$

$R, E, \eta \models \exists_\nu x.\phi$ iff $\exists S, F, a.\ R \circ S \downarrow$ and $\mu(a, S) \downarrow$ and $R, E \approx R, \nu S.F$ and $R \circ S, F, \eta \models \phi[a/x]$

$R, E, \eta \models \forall_\nu x.\phi$ iff $\forall S, F, a.\ R \circ S \downarrow$ and $\mu(a, S) \downarrow$ and $R, E \approx R, \nu S.F$ implies $R \circ S, F, \eta \models \phi[a/x]$

Fig. 1. Interpretation of Logical Formulae

Theorem 5.1 *If* $R, E \approx R, F$ *and* $R, E, \eta \models \phi$ *then* $R, F, \eta \models \phi$.

Proof The proof is by induction on the structure of ϕ. The base of the induction is assumed because the interpretation of atomic formulae is assumed to be closed under $\approx$. Most of the other steps are contained in the proof of the analogous result in [PT06, PT07]. Now consider the step for $\{E\}\phi$. Suppose that $R, E \approx R, F$ and $R, E, \eta \models \{G\}\phi$. Then there is some H with $R, E \approx R, G \propto H$ and $R, H, \eta \models \phi$. Since $\approx$ is transitive, we also have $R, F \approx R, G \propto H$, and so $R, F, \eta \models \{G\}\phi$. □

The converse to the above theorem holds, so that logically equivalent states are algebraically equivalent. The proof for the analogous result in [PT06, PT07, CPT07] suffices.

Theorem 5.2 $R, E \approx R, F$ *whenever* $R, E, \eta \models \phi$ *iff* $R, F, \eta \models \phi$ *for all* ϕ.

A number of important reasoning principles are justified in this context. The first of these reveals part of the intended meaning, that if some process has a property then any process that uses it as a role also has a version of that property, but guarded by a use of the 'says' modality.

Proposition 5.1

1. *If* $R, G, \eta \models \phi$ *and* $G \lessapprox F$ *then* $R, F \propto G, \eta \models \{F\}\phi$.
2. *If* $R, G, \eta \models \{E\}\phi$ *and* $E \approx F$ *then* $R, G, \eta \models \{F\}\phi$ *holds.*
3. $R, E, \eta \models \phi$ *iff* $R, E \propto E, \eta \models \phi$.
4. *If* $R, E, \eta \models \phi$ *then* $R, E, \eta \models \{E\}\phi$.
5. *If* $R, G, \eta \models (\{E_1\}\phi_1) * (\{E_2\}\phi_2)$ *then* $R, G, \eta \models \{E_1 \times E_2\}(\phi_1 * \phi_2)$.

Proof

1. If $R, G, \eta \models \phi$ and $G \lessapprox F$, then by definition of the interpretation $R, F \propto G, \eta \models \{F\}\phi$, since $R, F \propto G \approx R, F \propto G$.
2. If $R, G, \eta \models \{E\}\phi$ then there is some H such that $R, G \approx R, E \propto H$ and $R, H, \eta \models \phi$. If $E \approx F$ then $E \propto H \approx F \propto H$ by Proposition 4.1. Therefore $R, G, \eta \models \{F\}\phi$.
3. By Proposition 4.2 we have $E \approx E \propto E$ and so the third point holds by Theorem 5.1.
4. The fourth point follows from the third and the interpretation of 'says'.
5. Suppose $R, G, \eta \models (\{E_1\}\phi_1) * (\{E_2\}\phi_2)$. Then there are R_1, R_2, G_1, G_2 such that $R = R_1 \circ R_2$, $R, G \approx R, G_1 \times G_2$, and $R_j, G_j, \eta \models \{E_j\}\phi_j$ for $j = 1$ and $j = 2$. Then there are $F_1 \lessapprox E_1$ and $F_2 \lessapprox E_2$ such

that $R_j, G_j \approx R_j, E_j \propto F_j$ and $R_j, F_j, \eta \vDash \phi_j$ holds. By Proposition 4.2 we have $(E_1 \propto F_1) \times (E_2 \propto F_2) \approx (E_1 \times E_2) \propto (F_1 \times F_2)$. By transitivity of $\approx$ we have $R, G \approx R, (E_1 \times E_2) \propto (F_1 \times F_2)$. We also have $F_1 \times F_2 \lessapprox E_1 \times E_2$ as a scholium of Proposition 4.2. Furthermore $R, F_1 \times F_2, \eta \vDash \phi_1 * \phi_2$. Therefore $R, G, \eta \vDash \{E_1 \times E_2\}(\phi_1 * \phi_2)$. □

Many other semantic reasoning principles hold; see [CP09] for a discussion and a deductive system, in the absence of roles and 'says'.

6. Examples

The examples of this section are intended to illustrate some of the most common access control situations, how they may be *modelled* in resource-based process algebra, and which logical specifications they satisfy. They are *not* intended to give complete formal renderings of existing protocols, although we believe that this could certainly be done for many protocols, as **SCRP** has at least the same expressive power as CSP [Hoa85, Sch96, RSG01]. Similarly, we do *not* claim that our logical language is the only logical language that can express each of the properties below. Rather our goal is to describe, specify and reason about models of systems in which security concerns are critical. Here we demonstrate that the foregoing calculus is a practical semantic foundation for such work, with sufficient richness to capture structural properties of composite agents, for example co-signing and roles, but also with well-specified operational behaviour.

In each of the following examples we specify a modification on atomic actions only. The fact that this extends uniquely to a modification (on all actions) is a consequence of a mild generalization of a previous result (Proposition 17 in [CP09]). The generalization allows one to work with resource monoids that are not required to be total, and such that cancellation exists only as a partial function ($\forall R, S, T.\ R \circ S = R \circ T \Longrightarrow S = T$).

Let 2 be the resource monoid $\{0, 1\}$ with composition +, such that $0 + 0 = 0$, $0 + 1 = 1$, and $1 + 1 \uparrow$. This kind of resource is often used to represent a semaphore. We use these, and the positive integers, as channels to communicate and moderate interaction between processes: the approach in modelling languages like Demos2k is essentially the same.

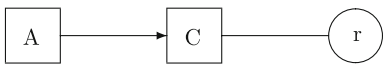

Fig. 2. A guarded resource

Example 6.1 (Access by proxy).

Consider the situation in Fig. 2, with a principal A attempting to access a resource r via a guard C. Assume that the way this works is that A makes a request to access r and then C either implements this or does not depending on whether it believes A has the right to perform this act.

This can be modelled through the use of a synchronous product

$$A \times C$$

where A and C are defined by the equations,

$$A = 1 : A + a : A \qquad C = 1 : C + c : C$$

and where a is the access request and c performs that access on behalf of A.

In order for this to make sense, we must define an appropriate resource monoid and modification function. These should ensure that certain sequences of actions may occur and certain others may not. In this example, and those that follow, this sequencing will be controlled through the use of semaphores which form part of the resource.

For example, consider the situation in which r is intended to hold an integer and a makes a request to increment r. We could take resources to be triples of the form

$$\langle m, n, L \rangle,$$

where m is an integer, $n \in 2$ and L is a set of actions of C that are allowed access to r. The integer m represents the contents of r. The integer n represents a resource component (like a semaphore or buffer) used to communicate the access request from A to C. The set L represents an access control list. Indeed, we adopt the informal convention of calling such a set an ACL.

A meaningful choice of resource monoid composition takes pointwise addition on the first two components and non-overlapping disjoint union for the third component. That is,

$$\langle m, n, L\rangle \circ \langle m', n', L'\rangle = \begin{cases} \langle m + m', n + n', L \cup L'\rangle & \text{if } L \cap L' = \emptyset \\ \uparrow & \text{otherwise} \end{cases}$$

for all suitable m, m', n, n', L, L'. The unit of this monoid is $\langle 0, 0, \emptyset\rangle$.

Let the set of atomic actions contain just the actions a and c. We choose the modification so that

$$\mu(a, \langle m, n, L\rangle) = \langle m, 1, L\rangle$$

$$\mu(c, \langle m, n, L\rangle) = \begin{cases} \langle m + 1, 0, L\rangle & \text{if } n = 1,\ c \in L \\ \uparrow & \text{otherwise.} \end{cases}$$

We find a sequence of access events of the form

$$\begin{aligned} \langle 0, 0, [c]\rangle, A \times C &\xrightarrow{a} \langle 0, 1, [c]\rangle, A \times C \\ &\xrightarrow{c} \langle 1, 0, [c]\rangle, A \times C \end{aligned}$$

for example. On the other hand, the increment of r only takes place after a request has been issued and the corresponding action of C found in the list L. Thus we see that C increments r on behalf of A.

Note that the choices of resource, resource composition and modification function are tied to significant aspects of operational behaviour, and that here we are just taking a simple example.

Many variations on this first simple example are easily expressed:

1. In the above example, an access c is not necessarily granted immediately after any request a, or indeed before any other request a. This is a simple choice, and it is easy to modify this to give more sophisticated interactions.
2. In the above example, accesses come in pairs $\langle a, c\rangle$ consisting of the request a and the requested action c. If two different agents wish to perform the same access upon r, but they have different permissions, we must have two distinct actions (a_1 and a_2, say, with corresponding c_1 and c_2) representing the two different access requests. For some situations, an alternative would be to use an ACL containing the access requests a_i, for then the same c may result from both a_i.
3. The resource and modification can be changed so that the action c may occur arbitrarily often after a single instance of a. An additional agent (and action) which stops access may then be added, if desired.
4. The modification can be changed so that permission to access can only be exercised once: the μ can be chosen to remove c from L following an action of c.
5. If L is a multiset, then μ can be chosen so that units of permission are consumed. Again, c removes an instance of c from L. It may be natural then to have an additional agent that creates permissions (by adding to the ACL).
6. A blacklisting approach is easily modelled by taking L to be a 'blocked-list': this is achieved by changing the modification so that μ is defined at c when $c \notin L$, instead of $c \in L$.

Example 6.2 (Direct access).

This example modifies the previous one so that A itself performs an action upon r after requesting and receiving permission from C.

We consider a system with process component $A \times C$ again, but with

$$\begin{aligned} A &= 1 : A + a : A' \\ A' &= 1 : A' + i : A \\ C &= 1 : C + c : C \end{aligned}$$

where this time a is the access request sent to C, c is the response sent back to A and i is the incrementation action on r.

Resources are taken to be of the form $\langle m, n, p, L\rangle$ where $m \in \mathbb{N}$, $n, p \in 2$ and L is a list of actions. The component p is used to represent the signal from C to A. Composition of resources is defined pointwise using the resource monoids defined above.

We choose the modification with:

$$\mu(a, \langle m, n, p, L\rangle) = \langle m, 1, p, L\rangle$$

$$\mu(c, \langle m, n, p, L\rangle) = \begin{cases} \langle m, 0, 1, L\rangle & \text{if } n = 1 \text{ and } c \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i, \langle m, n, p, L\rangle) = \begin{cases} \langle m + 1, n, 0, L\rangle & p = 1 \\ \uparrow & \text{otherwise.} \end{cases}$$

Then we find, for example, that with $c \in L$, the system $\langle 0, 0, 0, L\rangle, A \times C$ makes transition sequences $\ldots \xrightarrow{a} \ldots \xrightarrow{c} \ldots \xrightarrow{i} \ldots$, where each access i must be preceded by some response c and that must be preceded by some request a.

It is straightforward to extend the above to situations in which there are multiple agents $A_1, \ldots, A_n$ attempting to access multiple resources via multiple guards. We take a synchronous product of all the agents A_i and all the guards. More interesting situations arise when there are assumptions about concurrent accesses.

Example 6.3 (Joint-access requests).

In this example, there are two principals (processes) A_1 and A_2 that can only access some basic resource r via some guard process C after they have both made requests.

We take the process part of the system to be $A_1 \times A_2 \times C$ with

$$A_1 = 1 : A_1 + a_1 : A_1' \qquad A_1' = 1 : A_1' + i : A_1$$
$$A_2 = 1 : A_2 + a_2 : A_2 \qquad C = 1 : C + c : C$$

and resources of the form $\langle m, n_1, n_2, p, L\rangle$, where m is an integer, $n_1, n_2, p \in 2$ and L is an ACL represented by a set of actions. Take composition of resource to be pointwise addition. Take each access request a_j to increment n_j (undefined if $n_j = 1$); c to increment p and set both $n_j = 0$ just when $n_1 = n_2 = 1$ (otherwise undefined) and $c \in L$; let the access i resulting from a_1, a_2 increment m and set $p = 0$, when $p = 1$ (otherwise undefined). More precisely,

$$\mu(a_1, \langle m, n_1, n_2, p, L\rangle) = \langle m, n_1 + 1, n_2, p, L\rangle$$

$$\mu(a_2, \langle m, n_1, n_2, p, L\rangle) = \langle m, n_1, n_2 + 1, p, L\rangle$$

$$\mu(c, \langle m, n_1, n_2, p, L\rangle) = \begin{cases} \langle m, 0, 0, 1, L\rangle & \text{if } n_1 = n_2 = 1 \text{ and } c \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i, \langle m, n_1, n_2, p, L\rangle) = \begin{cases} \langle m + 1, n_1, n_2, 0, L\rangle & \text{if } p = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

for all $m \in \mathbb{N}$ and $n_1, n_2, p \in 2$.

Then we have transition sequences of the following forms:

$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{a_2} \ldots \xrightarrow{c} \ldots \xrightarrow{i} \ldots$$

$$\ldots \xrightarrow{a_2} \ldots \xrightarrow{a_1} \ldots \xrightarrow{c} \ldots \xrightarrow{i} \ldots$$

$$\ldots \xrightarrow{a_1 a_2} \ldots \xrightarrow{c} \ldots \xrightarrow{i} \ldots$$

amongst the possible system behaviours.

Let ϕ_j be the property that the resource-component at n_j is 1. From the point of view of **MBIa**, we have that the judgement relation

$$\langle m, 1, 1, p, L\rangle, A_1 \times A_2, \eta \vDash \phi_1 * \phi_2$$

holds for any m, p, η, because $\langle m, 1, 0, p, L\rangle, A_1, \eta \vDash \phi_1$ and $\langle 0, 0, 1, 0, \emptyset\rangle, A_2, \eta \vDash \phi_2$ both hold and

$$\langle m, 1, 0, p, L\rangle \circ \langle 0, 0, 1, 0, \emptyset\rangle = \langle m, 1, 1, p, L\rangle.$$

Thus, this judgement expresses the fact that both access requests have been made. From this it may be inferred that the response c may grant permission for the joint requests a_j to perform the access action i. To summarize: the $*$ connective describes co-signing situations in a particularly natural way.

There are important variants of the joint-access pattern.

1. This kind of example can be further refined so that the use of $*$ also requires that two signatories must hold *disjoint* permissions. This is closely related to the notion of *separation of duty* [LW08] and to Separation Logic [IO01, Rey02].
2. In the example above access can be granted after the two requests are made in any order, or simultaneously. This can be modified so that access is only granted if the two agents make their request simultaneously. Indeed, it is a particular version of a concurrent *handshaking* situation, as described in [PT06, PT07, CPT07].
3. The two authorizing agents must both give authorization in some chosen sequence. This can be captured in **SCRP** by a specific use of *resource-transfer* as exposed in [PT06, PT07, CPT07].
4. These examples extend to situations requiring agreement between multiple parties.

Example 6.4 (Exclusive access).
We suppose that we are in a situation in which we have two agents A_1 and A_2 that both wish to access r via C, but that only one of the agents A_i must be able to access r at any time. This is a classic concurrent mutual-exclusion situation and is modelled in **SCRP** through the use of a resource that can only be used by one process at a time.

We take resources to be of the form

$$\langle m, n_1, n_2, p_1, p_2, q, L\rangle,$$

where m is the integer-valued content of r, the $n_i, p_i, q \in 2$ and L is an ACL. We take the composition of resources to be defined pointwise with the operations indicated above.

We choose atomic actions a_1, a_2, c_1, c_2, i and a modification as follows:

$$\mu(a_1, \langle m, n_1, n_2, p_1, p_2, q, L\rangle) = \langle m, 1, n_2, p_1, p_2, q, L\rangle$$

$$\mu(a_2, \langle m, n_1, n_2, p_1, p_2, q, L\rangle) = \langle m, n_1, 1, p_1, p_2, q, L\rangle$$

$$\mu(c_1, \langle m, n_1, n_2, p_1, p_2, q, L\rangle) = \begin{cases} \langle m, 0, n_2, 1, p_2, q, L\rangle & \text{if } n_1 = 1 \text{ and } c_1 \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(c_2, \langle m, n_1, n_2, p_1, p_2, q, L\rangle) = \begin{cases} \langle m, n_1, 0, p_1, 1, q, L\rangle & \text{if } n_2 = 1 \text{ and } c_2 \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i, \langle m, n_1, n_2, p_1, p_2, q, L\rangle) = \begin{cases} \langle m, n_1, n_2, p_1, p_2, q, L\rangle & \text{if } q = 1, \text{ and } p_1 = 1 \text{ or } p_2 = 1 \\ \uparrow & \text{otherwise.} \end{cases}$$

We define the processes

$$\begin{aligned} A_1 &= 1 : A_1 + a_1 : A_1' \\ A_1' &= 1 : A_1' + i : A_1 \\ A_2 &= 1 : A_2 + a_2 : A_2' \\ A_2' &= 1 : A_2' + i : A_2 \\ C &= 1 : C + c_1 : C + c_2 : C \end{aligned}$$

where a_1, a_2 are the respective requests by A_1 and A_2 to perform i.

We take $L = \{c_1, c_2\}$ and consider any system

$$\langle m, n_1, n_2, p_1, p_2, q, L\rangle, A_1 \times A_2 \times C\ .$$

There are transition sequences of the three forms

$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{c_1} \ldots \xrightarrow{i} \ldots$$
$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{c_1} \ldots \xrightarrow{i} \ldots$$
$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{a_2} \ldots \xrightarrow{c_1} \ldots \xrightarrow{c_2} \ldots \xrightarrow{i} \ldots \xrightarrow{i}$$

amongst others, but not of the form

$$\ldots \xrightarrow{ii} \ldots$$

because $q \in 2$. That is, requests a_1 and a_2 can be made, possibly simultaneously, the responses c_1 and c_2 can come back, possibly simultaneously, and incrementations i can be made, but *not* simultaneously.

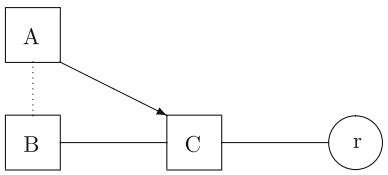

Fig. 3. Guarded resource with delegation

Example 6.5 (Authorization by delegation).

Consider a situation in which the guard C must now consult some other principal B who owns an ACL, L, that says that the requested access should be granted. This is a simple version of the situation described in Fig. 3.

Such a situation can be modelled using resources of the form $\langle m, n, p, q, k, L\rangle$ where m is an integer, n, p, q, k are copies of the semaphore 2, and L is a list of actions. Composition is defined pointwise from the operations on components considered in the previous examples. Let $L_0 = \{a\}$ and consider the resource $R_{L_0} = \langle 0, 0, 0, 0, 0, L_0\rangle$.

We use agents

$$\begin{aligned} A &= 1 : A + a : A_0 \\ A_0 &= 1 : A_0 + i : A \\ B &= 1 : B + bb' : B \\ C &= 1 : C + c : C + d : C \end{aligned}$$

in a product

$$A \times C \times \nu R_{L_0}.B$$

featuring a hiding. The modification is defined by:

$$\mu(a, \langle m, n, p, q, k, L\rangle) = \langle m, 1, p, q, k, L\rangle$$

$$\mu(b, \langle m, n, p, q, k, L\rangle) = \begin{cases} \langle m, n, 0, 1, k, L\rangle & \text{if } p = 1 \text{ and } a \notin L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(b', \langle m, n, p, q, k, L\rangle) = \begin{cases} \langle m, n, p, q, k, L\rangle & \text{if } a \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(c, \langle m, n, p, q, k, L\rangle) = \begin{cases} \langle m, n, 1, q, k, L\rangle & \text{if } n = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(d, \langle m, n, p, q, k, L\rangle) = \begin{cases} \langle m, n, p, 0, 1, L\rangle & \text{if } q = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i, \langle m, n, p, q, k, L\rangle) = \begin{cases} \langle m + 1, n, p, q, 0, L\rangle & \text{if } k = 1 \\ \uparrow & \text{otherwise.} \end{cases}$$

Thus: a is an access request made by A to B, using the channel n; c represents C asking B, using the channel p, if the request should be granted; b' represents B consulting its private ACL, L_0; b is the signal from B to C, using the channel q, that the access should be granted; d is the signal from C to A, using the channel k, that the access has been granted; i is the actual incrementation action that takes place, given that all the above have happened.

With the following resources,

$$R_0 = \langle 0,0,0,0,0,\emptyset\rangle \qquad R_1 = \langle 0,1,0,0,0,\emptyset\rangle$$
$$R_2 = \langle 0,0,1,0,0,\emptyset\rangle \qquad R_3 = \langle 0,0,0,1,0,\emptyset\rangle$$
$$R_4 = \langle 0,0,0,0,1,\emptyset\rangle \qquad R_5 = \langle 1,0,0,0,0,\emptyset\rangle$$

we have, for example, the system evolutions

$$\begin{aligned} & R_0, A \times C \times \nu R_{L_0}.B \\ \xrightarrow{a}\ & R_1, A_0 \times C \times \nu R_{L_0}.B \\ \xrightarrow{c}\ & R_2, A_0 \times C \times \nu R_{L_0}.B \\ \xrightarrow{b}\ & R_3, A_0 \times C \times \nu R_{L_0}.B \\ \xrightarrow{d}\ & R_4, A_0 \times C \times \nu R_{L_0}.B \\ \xrightarrow{i}\ & R_5, A \times C \times \nu R_{L_0}.B \end{aligned}$$

making use of

$$\frac{R_2 \circ R_{L_0}, B \xrightarrow{bb'} R_3 \circ R_{L_0}, B}{R_2, \nu R_{L_0}.B \xrightarrow{b} R_3, \nu R_{L_0}.B}$$

to give the b-transition.

Let ϕ be the assertion 'the q component of the resource is 1'. The relation

$$R_2, \nu R_{L_0}.B, \eta \vDash \langle b\rangle\phi$$

specifies that the system $R_2, \nu R_{L_0}.B$ can signal to C that the access should be granted. In more detail, this happens because

$$R_2, \nu R_{L_0}.B, \eta \vDash \exists_\nu x.\langle bx\rangle\phi$$

which holds because the private ACL L_0 can be consulted by B using the hidden action b'.

It is a straightforward matter to extend the preceding example to longer chains of trust: each delegation is specified in the logic by a suitable $\exists_\nu$ satisfaction statement. Statements may then be chained together using the definition of the satisfaction relation to show that appropriate access decisions can be taken.

Example 6.6 (Reduction to a role).

Consider a situation where some process C guards two basic resources r_1 and r_2, a process A and a role B. The process A can make an access requests a_j to increment the value m_j stored at r_j for both $j = 1$ and $j = 2$. However, in the role B it can only make the access request a_1.

Take resources of the form $\langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle$ where the $m_j \in \mathbb{N}$, the $p_j, q_j \in 2$ and L is a set of actions (representing an ACL containing permitted responses of C). Let

$$R = \langle 0,0,0,0,0,0,\{c_1,c_2\}\rangle \quad S = \langle 0,0,0,0,0,0,\{c_1\}\rangle \quad \mathbf{e} = \langle 0,0,0,0,0,0,\emptyset\rangle$$

be resources.

We consider the process terms:

$$\begin{aligned} A &= \nu R.A' + d : (A \propto B) \\ B &= \nu S.A' \\ A' &= 1 : A' + a_1 : A'' + a_2 : A'' \\ A'' &= 1 : A' + i_1 : A + i_2 : A \\ C &= 1 : C + c_1 : C + c_2 : C. \end{aligned}$$

We take μ as follows:

$$\mu(a_1, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \langle m_1, m_2, 1, p_2, q_1, q_2, L\rangle$$

$$\mu(a_2, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \langle m_1, m_2, p_1, 1, q_1, q_2, L\rangle$$

$$\mu(c_1, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \begin{cases} \langle m_1, m_2, p_1, p_2, 1, q_2, L\rangle & \text{if } p_1 = 1 \text{ and } c_1 \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(c_2, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \begin{cases} \langle m_1, m_2, p_1, p_2, q_1, 1, L\rangle & \text{if } p_2 = 1 \text{ and } c_2 \in L \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(d, R) = R$$

$$\mu(i_1, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \begin{cases} \langle m_1 + 1, m_2, p_1, p_2, 0, q_2, L\rangle & \text{if } q_1 = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i_2, \langle m_1, m_2, p_1, p_2, q_1, q_2, L\rangle) = \begin{cases} \langle m_1, m_2 + 1, p_1, p_2, q_1, 0, L\rangle & \text{if } q_2 = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

The system $\mathbf{e}, A$ has transition sequences of the form

$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{c_1} \ldots \xrightarrow{i_1} \ldots$$
$$\ldots \xrightarrow{a_2} \ldots \xrightarrow{c_2} \ldots \xrightarrow{i_2} \ldots$$
$$\ldots \xrightarrow{d} \ldots \xrightarrow{a_1} \ldots \xrightarrow{c_1} \ldots \xrightarrow{i_1} \ldots$$

amongst others, but not of the form

$$\ldots \xrightarrow{d} \ldots \xrightarrow{a_2} \ldots \xrightarrow{c_2} \ldots \xrightarrow{i_2} \ldots$$

because the permissions associated with B do not include those for the $\langle a_2, c_2\rangle$ request-response pair. Similarly, the system $\mathbf{e}, A \propto B$ has

$$\ldots \xrightarrow{a_1} \ldots \xrightarrow{c_1} \ldots \xrightarrow{i_1} \ldots$$

as a possible behaviour, but not

$$\ldots \xrightarrow{a_2} \ldots \xrightarrow{c_2} \ldots \xrightarrow{i_2} \ldots .$$

We may express logically the fact that $A \propto B$ cannot perform all the accesses that A can, given resource $\mathbf{e}$. Given $\phi := \langle a_2\rangle\langle c_2\rangle\langle i_2\rangle\top$ we find that

$$\mathbf{e}, A, \eta \vDash \phi \quad \mathbf{e}, B, \eta \vDash \neg\phi \quad \mathbf{e}, A \propto B, \eta \vDash \{A\}\neg\phi$$

hold.

Note that we can often use simple resource assertions instead of complex modal assertions: for example, it is often enough to specify that sufficient resource is present to enable an action to fire instead of specifying that the action can fire using a modality.

The following example is of a significantly different type. In the foregoing examples, the systems we modelled were all based on variants of agents with ACLs. The logical language we had was then used to make assertions about systems. In contrast, in the following example, a security-language is used to govern access decisions. Thus there are logical formulae *in* the models, as well as *about* the models.

Example 6.7 (Modelling access with assertion-based control).

Consider a system with four agents A, C_1, C_2, C_3. The first of these wishes to perform some operation. However, it only performs this action when it receives permission; it receives permission from the second agent; the second agent, in turn, receives the information that permission should be granted from the third agent, which it trusts. The third agent receives the information from the fourth agent, which it trusts. This is a simple chain-of-trust. The second agent does not explicitly trust the fourth, however it does so implicitly.

We assume the existence of a simple language of access assertions

$$p_0 ::= \mathsf{mayAcc}(A, i, r_1)$$
$$p ::= p_0 \mid \mathsf{states}(C_k, p_0) \mid \neg p_0$$

with $1 \leq k \leq 3$. That is, for this example, we are only concerned with the ability of A (defined below) to access some resource r_1 with the operation i. This is easily generalized by extending p_0 with many atoms. This language should not be confused with the Hennessy–Milner logic described earlier. Let $\mathcal{L}$ be the set of propositions of this language.

We assume the existence of basic actions with approximate intended meanings:

c_j^p	C_j states p
b_j^p	update belief of C_j with p, in the light of trusted statement that p
a	request to do some operation (access) on r_1
d	unlock the requested operation
i	do the operation requested

for $1 \leq j \leq 3$ and p, as above. In addition, for slightly technical reasons, there are also actions d', ${c_2'}^p$, b^p present. These are further explained by the modification.

We define a multiset of formulae of $\mathcal{L}$ to be *consistent* when it does not contain both p and $\neg p$, for any p. Reserve the letters capital Γ, Δ for such multisets. The set of multisets of consistent formulae is a resource monoid with the operation

$$\Gamma \circ \Delta = \begin{cases} \Gamma \cup \Delta & \text{if the multiset union } \Gamma \cup \Delta \text{ is consistent} \\ \uparrow & \text{otherwise} \end{cases}$$

for any Γ and Δ. We write the unit as $\emptyset$. When this composite is defined we say that Γ and Δ are consistent.

We take the protected resource to be an integer m and the operation that accesses it to be an incrementation i. The resources for the system we wish to describe are then of the form $\langle m, n, k, \Gamma\rangle$, where $m, n \in \mathbb{N}$, $k \in 2$ and Γ is a consistent set of formulae. The composition operation acts pointwise on the components of these quadruples, with the component operations indicated above. The unit is $\mathbf{e} = \langle 0, 0, 0, \emptyset\rangle$. As usual, the letters R, S stand for such resources. We define the abbreviation $R \circ \Delta = \langle m, n, k, \Gamma \circ \Delta\rangle$ for any resource $R = \langle m, n, k, \Gamma\rangle$ and consistent set of formulae Δ. Similarly, we write $p \in R$ as an abbreviation for $p \in \Gamma$, where Γ is the multiset component of R.

The modification is specified by:

$$\mu(a, \langle m, n, k, \Gamma\rangle) = \langle m, 1, k, \Gamma\rangle$$

$$\mu(b_1^p, R) = \begin{cases} R & \text{if } \mathsf{states}(C_2, p) \in R \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(b_2^p, R) = \begin{cases} R & \text{if } \mathsf{states}(C_3, p) \in R \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(b^p, R) = R \circ [p]$$

$$\mu(c_2^p, R) = \begin{cases} R \circ [\mathsf{states}(C_2, p)] & \text{if } n = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu({c_2'}^p, R) = \begin{cases} R & \text{if } p \in R \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(c_3^p, R) = R \circ [\mathsf{states}(C_3, p)]$$

$$\mu(d, \langle m, n, k, \Gamma\rangle) = \begin{cases} \langle m, 0, 1, \Gamma\rangle & \text{if } n = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(d', R) = \begin{cases} R & \text{if } p \in R \quad \text{and} \quad n = 0 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(i, \langle m, n, k, \Gamma\rangle) = \begin{cases} \langle m + 1, 0, 0, \Gamma\rangle & \text{if } k = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

on atomic actions.

We define the resources (multisets)

$$S_1 = \mathbf{e} = S_2 \quad S_1' = \mathbf{e} \circ [p_0] = S_2'.$$

We consider the process $A_1 \times C_1 \times C_2 \times C_3$, where

$$\begin{aligned}
A &= 1 : A + a : A + i : A \\
C_1 &= \nu S_1.D_1 \\
D_1 &= 1 : D_1 + dd' : D_1 + b_1^{p_0}\, b^{p_0} : D_1 \\
C_2 &= \nu S_2.D_2 \\
D_2 &= 1 : D_2 + b_2^{p_0}\, b^{p_0} : D_2 + c_2^{p_0}\, {c_2'}^{p_0} : D_2 \\
C_3 &= 1 : C_3 + c_3^{p_0} : C_3
\end{aligned}$$

Thus: A requests to perform the operation i using the access a; the guard C_1 must decide whether to allow this access or not, and to do this it consults C_2, which in turn consults C_3. The fact that C_1 trusts C_2, which trusts C_3 is built into the model.

There is a transition sequence:

$$\begin{aligned}
&\mathbf{e},\ A \times \nu S_1.D_1 \times \nu S_2.D_2 \times C_3 \\
\xrightarrow{a}\ &\langle 0, 1, 0, \emptyset\rangle,\ A \times \nu S_1.D_1 \times \nu S_2.D_2 \times C_3 \\
\xrightarrow{c_3^{p_0}}\ &\langle 0, 1, 0, [\mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1.D_1 \times \nu S_2.D_2 \times C_3 \\
\xrightarrow{b_2^{p_0}}\ &\langle 0, 1, 0, [\mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{c_2^{p_0}}\ &\langle 0, 1, 0, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{b_1^{p_0}}\ &\langle 0, 1, 0, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{d}\ &\langle 0, 0, 1, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{i}\ &\langle 1, 0, 0, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{a}\ &\langle 1, 1, 0, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{d}\ &\langle 1, 0, 1, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
\xrightarrow{i}\ &\langle 2, 0, 0, [\mathsf{states}(C_2, p_0), \mathsf{states}(C_3, p_0)]\rangle,\ A \times \nu S_1'.D_1 \times \nu S_2'.D_2 \times C_3 \\
&\dots
\end{aligned}$$

for example. Thus belief cascades down from C_3 to C_1 and this allows A to perform the operation requested. In the present model, belief cannot be revoked, and so the chain-of-trust does not need to be consulted for the second incrementation. The action ${c_2'}^{p_0}$ is used to check if C_2 believes p_0, whilst the action $c_2^{p_0}$ extrudes information back into the global resource by making the public statement $\mathsf{states}(C_2, p_0)$. The action d' is used to check that C_1 believes p_0, whilst d extrudes information by unlocking the incrementation. The operational semantics of hiding means that the actions ${b_i}^{p_0}$ check the current collection of public statements, whilst our usages of the action b^{p_0} update the internal beliefs of C_1' and C_2' according to those checks.

The intended trust relation

$$C_2 \text{ states } p_0 \quad \text{implies} \quad C_1 \text{ believes } p_0$$

is partially captured by the Hennessy–Milner logic as follows. Let $R = \langle m, n, k, \Gamma\rangle$ and suppose that Γ and $[p]$ are consistent. The relation $R,\ C_1, \eta \vDash \mathsf{states}(C_2, p_0)$ means that $\mathsf{states}(C_2, p_0) \in \Gamma$. Then $R,\ D_1 \overset{b_1 p_0}{\to} b p_0\ R,\ D_1$ and so $R,\ C_1 \overset{b_1 p_0}{\to} R,\ \nu S_1' \cdot D_1$. Let p_0 be valued so that $T, G, \eta \vDash p_0$ iff p_0 lies in the multiset of formulae in T, for any T, G, η. The relation $R,\ \nu S_1' \cdot D_1, \eta \vDash \exists_\nu x \cdot p_0$ captures the fact that the agent $\nu S_1' \cdot D_1$ believes p_0. Hence the trust relation is described here by the formula

$$R,\ C_1, \eta \vDash \mathsf{states}(C_2, p_0) \to \exists x.\langle x\rangle(\exists_\nu y \cdot p_0)$$

which says that if C_1 has the public statement from C_2 that p_0 then C_1 can change to become an agent which believes p_0 (it has this as private information). This argument scales up to the whole system because the four components of the synchronous product defining it may always tick.

We note that the use of formulae-as-resources and hiding gives a simple account of agents with private belief, and conjecture that more sophisticated versions of this approach could be of use in many other situations.

These examples can be further extended so that longer chains-of-trust and more complex patterns are used for determining accesses.

7. Global simulation and multiplicative implication

The logical calculus **MBIa** of Sect. 5 above combines Hennessy–Milner-style modal connectives with a separating conjunction in the style of **BI** and Separation Logic. The multiplicative implication, $-\!\!*$, of **BI** was omitted as it complicates the treatment somewhat. Here, we rectify that omission and also include versions of the multiplicative modalities $\langle a\rangle_\nu$ and $[a]_\nu$ previously considered for the original version of **MBI** [PT06, PT07, CPT07].

In order to give such a treatment we must make some alterations to the notion of bisimulation, and so to the process calculus. We define a modified process calculus **ACCRPb** together with a new simulation relation $\sim$. The new process calculus is formed by replacing the rule for roles with the rule

$$\frac{R, F \overset{a}{\to} R', F' \quad R \circ S, E \overset{a}{\to} R' \circ S', E'}{R \circ S, E \propto F \overset{a}{\to} R' \circ S', E' \propto F'}\ (F \lesssim E) \tag{4}$$

where $\lesssim$ is the largest relation on processes such that, if $E_1 \sim E_2$ and $R, E_1 \overset{a}{\to} R', F_1$ for any R, R', F_1, then there is some F_2 such that $R, E_2 \overset{a}{\to} R', F_2$ and $E_2 \lesssim F_2$. We call the relation $\sim = \lesssim \cap \gtrsim$ the *global bisimulation relation*. As with $\approx$, the definition of this relation can be stratified to show that the mutually recursive definition above makes sense. The relation $\sim$ is extended to states by taking $R, E \sim S, F$ just if $R = S$ and $E \sim F$.

The relation $\sim$ is a congruence and all of the results of Proposition 4.2 are retained, but with $\lesssim$ (and $\sim$) replacing $\lessapprox$ (respectively $\approx$) throughout.

The logical language **MBIa** is extended as follows

$$\phi ::= \ldots \mid \phi \mathrel{-\!\!*} \phi \mid \langle a\rangle_\nu \phi \mid [a]_\nu \phi$$

to give a new language **MBIb**.

The notion of valuation for the logic is changed so that $\mathcal{V}(p)$ is closed under the relation $\sim$ on states. The interpretation of Fig. 1 is modified so that, wherever some relation $R, E \approx R, F$ is expressed, it is replaced by the relation $E \sim F$ on processes. The use of $\lessapprox$ for $\{E\}\phi$ is replaced by a use of $\lesssim$. In addition we extend the interpretation with the clauses in Fig. 4.

$$R, E, \eta \vDash \phi \mathrel{-\!\!*} \psi \ \text{ iff } \ \forall S, F.\ \ S, F, \eta \vDash \phi \ \text{ implies } \ R \circ S, E \times F, \eta \vDash \psi$$

$$R, E, \eta \vDash \langle \mathsf{a}\rangle_\nu \phi \ \text{ iff } \ \exists S, E'.\ R \circ S, E \overset{\mathsf{a}}{\to} \mu(a, R \circ S), E' \ \text{ and } \ \mu(a, R \circ S), E', \eta \vDash \phi$$

$$R, E, \eta \vDash [\mathsf{a}]_\nu \phi \ \text{ iff } \ \forall S, E'.\ R \circ S, E \overset{\mathsf{a}}{\to} \mu(a, R \circ S), E' \ \text{ implies } \ \mu(a, R \circ S), E', \eta \vDash \phi$$

Fig. 4. Extended Interpretation of Logical Formulae

We then have that Theorem 5.1 holds with $\sim$ replacing $\approx$ throughout and with all formulae, ϕ, drawn from **MBIb**. On the other-hand counterexamples exist to Theorem 5.2 when $\sim$ is used in place of $\approx$ (and where μ is a non-trivial modification function). We conjecture that, in general, under reasonable conditions, Theorem 5.2 using $\sim$ does not hold. The results of Proposition 5.1 hold with $\lesssim$ (resp. $\sim$) replacing $\lessapprox$ (resp. $\approx$) throughout.

The relation $\approx$ is not used when dealing with **MBIb** since then the essential result of Theorem 5.1 does not hold for many non-trivial modification functions. That is, there can be E and F such that $R, E \approx R, F$ for all R but some ϕ such that $R, E, \eta \vDash \phi$ and $R, F, \eta \vDash \neg\phi$, see [CPT07] for details. Indeed $\sim$ should be used whenever we wish to use a logical language that either features the multiplicative modalities or that features both the multiplicative implication and the additive modalities. To summarize:

1. For the $(\top, I, \wedge, \vee, \neg, \to, \langle - \rangle, [-], \exists, \forall, \exists_\nu, \forall_\nu)$-fragment of the logical language the use of the local simulation, $\lesssim$, throughout will be suitable.
2. For any fragment featuring $\langle - \rangle_\nu$ or $[-]_\nu$, or $-\!\!*$ together with either of $\langle - \rangle$, $[-]$ the global simulation, $\lesssim$, should be used throughout.

The new logical connectives can be used as follows:

($-\!\!*$). Imagine a situation in which there is some component E that is intended to plug into certain types of system, and imagine that E comes with resources R. Suppose that we wish to guarantee that whenever R, E is plugged into some suitable system S, F, that the resulting compound system makes no accesses to some resource r. This can be expressed by the proposition $R, E \vDash \phi -\!\!* (\neg\psi)$, where ψ is an appropriate 'access r' proposition, as in the previous examples, and ϕ represents the 'suitability' condition.

($[a]_\nu$). Let ψ be as in the previous example. Suppose that we have some system R, E satisfying $[a]_\nu(\neg\psi)$. The resources R often include the permissions of E (as with the ACLs of the earlier examples) and that resource composition takes the union of permissions. Then the logical formula above guarantees that there is no way for E to access the resource r, no matter how its permissions are extended.

($\langle a \rangle_\nu$). We may often wish to specify a system that cannot make a particular access, for example with $R, E \vDash \neg\langle c \rangle \top$ for some access c, because it lacks permission, but such that it, if granted permission it can make the access, e.g $R, E \vDash \langle c \rangle_\nu \top$.

8. Directions

Mild changes to the rules of **SCRP** can result in calculi with significantly different properties. For example, it has been shown [CP09] how the algebraic and logical theories can be considerably strengthened from the original version presented in [PT06, PT07, CPT07]. The simple changes made were small changes to the coherence conditions on modification and side-conditions on the operational rules. Those changes result in the *admissibility* of the following rule:

$$\frac{R, E \xrightarrow{a} R', E'}{R \circ S, E \xrightarrow{a} R' \circ S, E'} \quad (R \circ S \downarrow)$$

for all R, R', S, S', E, E'. That is, this rule emerges as a property of systems. We note that it is similar to the *frame rule* of Separation Logic, [IO01, Rey02]. When this rule is not admissible, it may be useful to include it explicitly. In particular, this rule leads to algebraic relations like $E \times 1 \approx E$ and for the role constructor, associativity, $E \propto (F \propto G) \approx (E \propto F) \propto G$, for all E, F and G. On the logical side, we get logical axioms like $\phi * I \leftrightarrow \phi$. It also leads to $(\{F \propto G\}\phi) \leftrightarrow (\{F\}(\{G\}\phi))$ between the $\{-\}$ modality and the role constructor. Despite these useful consequences, we have not chosen to include this frame rule (implicitly or explicitly), since it may not be applicable to all the situations we wish to model. For example, if resources contain access blacklists, and composition joins those lists by non-overlapping union, as above, then the frame rule above would be undesirable. In such situations, a more subtle treatment using the order of the resource monoid should be used, as with the intuitionistic version of **MBI** in [CP09].

A good deal of work remains to be done on the model-checking problems for **SCRP**-like calculi. These problems are significantly harder than standard model-checking problems, since they involve searches for resource decompositions, searches across processes and appropriate (bi)simulation checks. However, the possession of such model-checking tools would give more powerful reasoning methods for semantically-justified logical access control, in a manner complementary to that of [ABL93, DeT02].

Together with B. Monahan we have produced a modelling environment, *LD2k* , for large-scale distributed systems. This work is ongoing, but summaries of early versions may be found in [CMP08a, CMP08c]. The tool is an extension of the existing Demos2k tool [Dem] which has firm foundations in the process algebra SCCS. It is intended that *LD2k* will have a process calculus semantics in a variant of **SCRP**. Access control is a property of

interest in many of the models we wish to consider and the work presented herein is a foundational study related to such models. However, in the models we wish to consider resources will typically be physically distributed and it will be pragmatic to have *location* play a role as fundamental as that of resource. Location should be a first-class citizen in the underlying process calculus, in the same way that resource is a first-class citizen in **SCRP**. The paper [PT07] contains some basic ideas about the embodiment of the location concept in a process algebra with transitions of the form $L, R, E \xrightarrow{a} L', R', E'$, where L, L' are locations, R, R' are resources and E, E' are processes. These have been further developed in [CMP08b]. Practical security modelling in Demos2k and a new version of *LD2k* is ongoing work with colleagues, particularly at HP Labs [BCG08, BGS08, BBC08].

Acknowledgments

We are grateful to Brian Monahan and Jonathan Hayman for their help and suggestions. Guy McCusker suggested a version of the plugging example for $-\!\!*$ given in Sect. 7 in a related discussion. We thank also Chris Tofts for helpful contributions related to this work.

References

[Aba03] Abadi M (2003) Logic in access control. In: Proceedings of LICS'03, pp 228–233
[ABL93] Abadi M, Burrows M, Lampson B, Plotkin G (1993) A calculus for access control in distributed systems. ACM Trans Progrogram Lang Syst 4(15):706–734
[AG97] Abadi M, Gordon A (1997) A calculus for cryptographic protocols: the spi calculus. In: Proceedings conference Computer and Communications Security. ACM Press, London, pp 36–47
[BBC08] Baldwin A, Beres Y, Casassa Mont M, Griffin J, Shiu S (2008) Identity analytics: using modeling and simulation to improve data security decision making. Technical Report HPL-2008-188, HP Labs, 2008. http://www.hpl.hp.com/techreports/2008/HPL-2008-188.html
[BCG08] Beautement A, Coles R, Griffin J, Ioannidis C, Monahan B, Pym D, Sasse MA, Wonham M (2008) Modelling the human and technological costs and benefits of USB memory stick security. In: Johnson ME, (ed) Managing information risk and the economics of security. Springer, Heidelberg
[BGS08] Beres Y, Griffin J, Shiu S, Heitman M, Markle D, Ventura P (2008) Analysing the performance of security solutions to reduce vulnerability exposure window. In: Proceedings of 2008 annual computer security applications conference (ACSAC). IEEE
[Bir79] Birtwistle G (1979) Demos—discrete event modelling on Simula. Macmillan, New York
[BN07] Becker MY, Nanz S (2007) A logic for state-modifying authorization policies. In: 12th European symposium on research in computer security (ESORICS), Lecture Notes in Computer Science, vol 4734
[CFS96] Coyne EJ, Feinstein HL, Sandhu R, Youman CE (1996) Role-based access control models. IEEE Comput 29(2):38–47
[CMP08a] Collinson M, Monahan B, Pym D (2008a) Located Demos2k—towards a tool for modelling processes and distributed resources. Technical Report HPL-2008-76, HP Labs, 2008. http://library.hp.com/techpubs/2008/HPL-2008-76.html
[CMP08b] Collinson M, Monahan B, Pym D (2008b) A logical and computational theory of located resource. Technical Report HPL-2008-74R1, HP Labs, 2008 (Submitted). http://library.hp.com/techpubs/2008/HPL-2008-74R1.html
[CMP08c] Collinson M, Monahan B, Pym D (2008c) An update to located Demos2k. Technical Report HPL-2008-205, HP Labs, 2008. http://library.hp.com/techpubs/2008/HPL-2008-205.html
[CP09] Collinson M, Pym D (2009) Algebra and logic for resource-based systems modelling. Technical Report HPL-2009-21, HP Labs, 2009 (Submitted). http://library.hp.com/techpubs/2009/HPL-2009-10.html.
[CPT07] Collinson M, Pym D, Tofts C (2007) Errata for formal aspects of computing (2006) 18:495–517 and their consequences. Formal Aspects Comput 19(4):551–554
[Dem] Demos2k. http://www.demos2k.org
[DeT02] DeTreville J (2002) Binder, a logic-based security language. In: Proceedings of 2002 IEEE symposium on security and privacy, pp 105–113
[GRS04] Guelev DP, Ryan MD, Schobbens P-Y (2004) Model-checking access control policies. In: Seventh information security conference (ISC'04), Lecture Notes in Computer Science, vol 3225. Springer, Heidelberg
[HM85] Hennessy M, Milner R (1985) Algebraic laws for nondeterminism and concurrency. J ACM 32(1):137–161
[Hoa85] Hoare CAR (1985) Communicating sequential processes. Prentice-Hall, Englewood Cliffs
[IO01] Ishtiaq S, O'Hearn PW (2001) BI as an assertion language for mutable data structures. In: Proceedings of POPL 2001. ACM, London, pp 14–26
[KYM06] Kamoda H, Yamaoka M, Matsuda S, Broda K, Sloman M (2006) Access control policy analysis using free variable tableaux. Information Processing Society of Japan (IPSJ) Digital Courier, vol 2
[LAB92] Lampson B, Abadi M, Burrows M, Wobber E (1992) Authentication in distributed systems: theory and practice. ACM Trans Comput Syst 4(10):265–310
[Lam71] Lampson BW (1971) Protection. In: Proceedings of fifth Princeton symposium information sciences and systems, pp 437–443
[LW08] Li N, Wang Q (2008) Beyond separation of duty: An algebra for specifying high-level security policies. J ACM 55(3)
[Mil80] Milner R (1980) A calculus of communicating systems, Lecture Notes in Computer Science, vol 92. Springer, Heidelberg
[Mil83] Milner R (1983) Calculi for synchrony and asynchrony. Theor Comput Sci 25:267–310
[Mil89] Milner R (1989) Communication and concurrency. Prentice-Hall, Englewood Cliffs

[OHe07] O'Hearn PW (2007) Resources, concurrency and local reasoning. Theor Comput Sci 375(1–3):271–307
[OP99] O'Hearn P, Pym D (1999) The logic of bunched implications. Bull Symb Logic 5(2):215–244
[Plo04] Plotkin GD (2004) Structural operational semantics. J Logic Algebraic Program 60:17–139 (Original manuscript 1981)
[POY04] Pym D, O'Hearn P, Yang H (2004) Possible worlds and resources: the semantics of **BI**. Theor Comput Sci 315(1):257–305
[PT06] Pym D, Tofts C (2006) A calculus and logic of resources and processes. Formal Aspects Comput 18(4):495–517. Errata in [CPT07]
[PT07] Pym D, Tofts C (2007) Systems modelling via resources and processes: philosphy, calculus, semantics, and logic. In: Cardelli L, Fiore M, Winskel G (eds) Computation, meaning and logic: articles dedicated to Gordon Plotkin, Electronic Notes in Theoretical Computer Science, vol 107. Elsevier, Amsterdam, pp 545–587. Errata in [CPT07]
[Pym99] Pym D (1999) On bunched predicate logic. In: Proceedings of LICS'99, pp 183–192. IEEE, New York
[Pym02] Pym DJ (2002) The semantics and proof theory of the logic of bunched implications, Applied Logic Series, vol 26. Kluwer, Dordrecht. Errata at: http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf
[Rey02] Reynolds JC (2002) Separation logic: a logic for shared mutable data structures. In: Proceedings of LICS'02. IEEE, New York, pp 55–74
[RSG01] Ryan P, Schneider S, Goldsmith M, Lowe G, Roscoe B (2001) The modelling and analysis of security protocols. Addison-Wesley, Reading
[Sch96] Schneider S (1996) Security properties and CSP. In: IEEE symposium on security and privacy, pp 174–187
[SMR06] Scedrov A, Mitchell JC, Ramanathan A, Teague V (2006) A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. Theor Comput Scie 353:118–164
[SS75] Saltzer JH, Shroeder MD (1975) The protection of information in computer systems. Proc IEEE 63(9):1278–1308
[Sti01] Stirling C (2001) Modal and temporal properties of processes. Springer, Heidelberg

Received 9 July 2008
Accepted in revised form 9 February 2009 by J.V. Tucker
Published online 17 March 2009