A Distributed Calculus for Rôle-Based Access Control

Chiara Braghin
Dip. di Informatica
Univ. Ca' Foscari di Venezia
braghin@dsi.unive.it

Daniele Gorla
Dip. di Sistemi e Informatica
Univ. di Firenze
gorla@dsi.unifi.it

Vladimiro Sassone
Dept. of Informatics
University of Sussex
vs@susx.ac.uk

Abstract

Rôle-based access control (RBAC) is attracting increasing attention because it reduces the complexity and cost of security administration by interposing the notion of *rôle* in the assignment of permissions to users. In this paper, we present a formal framework relying on an extension of the π–calculus to study the behaviour of concurrent systems in a RBAC scenario. We define a type system ensuring that the specified policy is respected during computations, and a bisimulation to equate systems. The theory is then applied to three meaningful examples, namely finding the 'minimal' policy to run a given system, refining a system to be run under a given policy (whenever possible), and minimizing the number of users in a given system without changing the overall behaviour.

1 Introduction

Rôle-based access control (RBAC) [6, 17] has recently emerged as a widely accepted alternative to classical discretionary and mandatory access controls: a standard is currently under development by the National Institute of Standards and Technology (NIST) [7] and several commercial applications directly support some forms of RBAC, e.g., Oracle, Informix and Sybase in the field of commercial database management systems.

RBAC is a flexible and policy-neutral access control technology: it regulates the access of users to information and system resources on the basis of activities the users need to execute in the system. The essence of RBAC lies with the notions of *user*, *rôle* and *permission*: users are authorized to use the permissions assigned to the rôles they belong to. More specifically, RBAC allows for a preliminary assignment of permissions to rôles (thus abstracting from the users that will play the various rôles at run-time). A user may then establish multiple sessions, e.g., by signing on to the system, during which he activates a subset of rôles that he is a member of. This greatly simplifies management tasks since it reduces the cost of administering access control policies, as well as making the administration process less error-prone. Anyway, the complexity of models (e.g., in large systems the number of rôles can exceed hundreds or thousands) demands a structured approach to the analysis and design of such systems.

This paper aims at developing a foundational theory for system behaviours in a RBAC scenario; to the best of our knowledge this is the first attempt in this direction. Our reference model is the so-called RBAC96 model, introduced by Sandhu et al. in the seminal paper [17]. More advanced RBAC models include rôle hierarchies and constraints such as rôle mutual exclusion, separation of duty, delegation of authority and negative permissions. The starting point is the π–calculus [19], which provides very well-established mathematical tools for expressing concurrent and possible distributed systems. Essentially, our idea is to equip the π–calculus with the notion of users (i.e., named processes), with

two new constructs for activation/deactivation of rôles, and with a way to grant permissions to rôles. This is accounted for by associating each process with a name representing a *user* and with a set ρ recording the rôles activated by the user during the current session. Hence, the term $r\{\!|P|\!\}_\rho$ represents a session of the user named r, running a process P with rôles ρ. The definition of the calculus is completed by two constructs that model rôle's activation/deactivation. The semantics of the new primitives is defined by the following reductions:

$$r\{\!|\mathbf{role}\ R.P|\!\}_\rho \longmapsto r\{\!|P|\!\}_{\rho\cup\{R\}} \qquad\qquad r\{\!|\mathbf{yield}\ R.P|\!\}_\rho \longmapsto r\{\!|P|\!\}_{\rho-\{R\}}$$

Intuitively, when a user r activates a rôle R during a session, R must be added to the set of activated rôles ρ, and the remaining of the session P will be executed with the set ρ updated. Vice versa for the deactivation of R.

As an example, the following system

$$client\{\!|\mathbf{role}\ \texttt{auth\_client}.port\_80\langle index.html\rangle.P|\!\}_\rho \parallel server\{\!|port\_80(x).Q|\!\}_{\rho'}$$

models the interaction between a client and an HTTP server. The system contains two users, *client* and *server*, running in parallel. It may evolve as follows. First, user *client* activates the rôle `auth_client` by exercising the **role** action, which in practice would involve to authenticate herself by means of a secure certificate. Then, she sends the request to the HTTP server to the usual port 80, i.e., performs an output action on the channel *port_80*.

The introduction of named users immediately leads to the possibility to implement the system in a distributed way. However, in distributed systems such as the Internet, it is unrealistic to assume global channels. To overcome this problem, we introduce the notion of localized channels *à la* Dπ [11], where each channel is associated to a single user. Syntactically, we achieve this feature by tagging output actions to specify the location of the channel where the exchange should take place. Thus the example above may be rewritten as:

$$client\{\!|\mathbf{role}\ \texttt{auth\_client}.port\_80^{server}\langle index.html\rangle.P|\!\}_\rho \parallel server\{\!|port\_80(x).Q|\!\}_{\rho'}$$

In the rest of the paper we will use this second version of the calculus. Adopting the first version would not change the nature of the calculus, and it would preserve, mutatis mutandis, the validity of our results.

Moreover, we allow user names to be exchanged during communications. This feature adds flexibility and realism to the language, since in distributed systems users have only a partial and evolving knowledge of their execution environment. For example, the client above can be generalized to leave the server identity unspecified and to dynamically retrieve it with an input from channel *choose_a_server*:

$$client\{\!|\mathbf{role}\ \texttt{auth\_client}.choose\_a\_server(x).port\_80^{x}\langle index.html\rangle.P|\!\}_\rho$$

More details on our calculus, together with some illustrative examples, will be given in Section 2.

The mapping among users, rôles and permissions, which controls the access of subjects to objects, is achieved by a pair of relations $(\mathcal{U};\mathcal{P})$, called *RBAC schema*. In $(\mathcal{U};\mathcal{P})$, relation $\mathcal{U}$ is the association users-to-rôles, while $\mathcal{P}$ is the association permissions-to-rôles. As a first contribution of this paper, we define in Section 3 a type system which complements the dynamics of the calculus by providing static guarantees that systems not respecting a given RBAC schema are rejected. In the client/server example above, a client not being authenticated (i.e. not performing a **role** `auth_client` before interacting with

the server) would be rejected, if the RBAC schema enables only authorized users to perform HTTP requests.

Often, the overall structure of a distributed system cannot be known statically. Thus, a typing approach may not be usable in practice. What is needed is a technique to study system components in isolation, compositionally, and under different schemas. Hence, as a second theoretical contribution, in Section 4 we introduce a labelled transition system equipped with some dynamic checks that gives the semantics of programs in a structured way. Moreover, the labelled transition system yields a bisimulation equivalence, adequate with respect to a standardly defined (typed) barbed congruence, that enables the development of some interesting algebraic laws. As an example, we show how RBAC schemas may change the semantic theory of the π–calculus. Consider the following system, adapted from the client/server example above:

$$(\nu\ port\_80^{server}\!:R)(client\{\!|port\_80^{server}\langle index.html\rangle.P|\!\}_{\emptyset} \parallel server\{\!|port\_80(x).Q|\!\}_{\rho'})$$

where $(\nu\ port\_80^{server} : R)$ is the standard restriction operator of a typed π–calculus (it declares $port\_80^{server}$ at type R and limits the visibility of the channel to *client* and *server* only). By resuming the assumption that only authorized users can perform HTTP requests, the above system is blocked, i.e. it is equivalent to the empty system **0**, because the client has not been authenticated. On the contrary, in the pure π–calculus a similar term would have been equivalent to the term resulting from the client/server exchange.

In Section 5 we use types and bisimulations to deal with three meaningful examples: finding the 'minimal' RBAC schema to execute a system, refining a system to be well-typed w.r.t. a given schema (whenever possible), and minimizing the number of users in a given system without changing the overall behaviour. We conclude by comparing our approach with related work in Section 6.

2 The Language

In this section we formally describe the calculus. First, we define its syntax and operational semantics; then, we formalize the RBAC schema to describe the rôles-to-users and permissions-to-rôles assignment.

2.1 Syntax

The calculus is a conservative extension of the π–calculus. We presuppose the following countable and pairwise disjoint sets: $\mathcal{R}$, ranged over by $R, S, \ldots$, of rôle names; $\mathcal{N}_u$, ranged over by $r, s, \ldots$, of user names; $\mathcal{N}_c$, ranged over by $a, b, \ldots$, of channel names; $\mathcal{V}$, ranged over by $x, y, \ldots$, of variables. The syntax of the calculus is given in Table 1, with restricted channels equipped with a rôle as described in Section 2.3.

A system consists of the parallel composition of user sessions that can share restricted channels. A user session $r\{\!|P|\!\}_\rho$ represents a *user* named r executing the code P with the set ρ recording r's active rôles. Observe that different sessions of the same user can run in parallel within a system A: this is the usual notion of sessions in RBAC models.

Processes **nil**, $P \mid Q$, $!P$, $[u = v]P$, $(\nu a : R)P$, $a(x).P$, $u\langle v\rangle.P$ are the ordinary π-like constructs representing respectively the inactive process, parallel composition of processes, replication (to model recursive process behaviours), value matching, restriction of channel names and standard input/output actions over channels. The novelty of the calculus resides in the prefixes **role** R and **yield** R, and in the locality of channels, as already described in the Introduction. Actions **role** R and **yield** R implement

User names	$r, s, ... \in \mathcal{N}_u$			
Channel names	$a, b, ... \in \mathcal{N}_c$			
Channels	$a^r, b^s, ... \in C = \mathcal{N}_c \times \mathcal{N}_u$			
Values	$m, n, ... \in \mathcal{N}_u \cup C$			
Variables	$x, y, ... \in \mathcal{V}$			
Identifiers	$u, v, ... \in \mathcal{N}_u \cup \mathcal{V} \cup (\mathcal{N}_c \times (\mathcal{N}_u \cup \mathcal{V}))$			
Rôle names	$R, S, ... \in \mathcal{R}$			
Processes	P, Q	::=	**nil**	*nil process*
		|	$P \mid Q$	*composition*
		|	$!P$	*replication*
		|	$(\nu a : R)P$	*restriction*
		|	$[u = v]P$	*matching*
		|	$a(x).P$	*input*
		|	$u\langle v\rangle.P$	*output*
		|	**role** $R.P$	*rôle activation*
		|	**yield** $R.P$	*rôle deactivation*
Systems	A, B	::=	$\mathbf{0}$	*empty system*
		|	$r\{\!\{P\}\!\}_\rho$	*user session*
		|	$A \parallel B$	*composition*
		|	$(\nu a^r : R)A$	*restriction*

Table 1: Syntax of the Calculus

activations/deactivations of rôles in the user session they belong to, and modify the session rôles accordingly. As usual, in the rest of the paper we will omit trailing inactive processes.

Channels are univoquely associated to users. The set of channels C is formed by coupling a channel name with a user name, and it is ranged over by $a^r, b^s, \ldots$. Identifiers, ranged over by $u, v, \ldots$, collect together user names, variables, channels and compound entities made up by a channel name and a variable. The only transmissible *values* are user names and channels and are ranged over by $m, n, \ldots$. Channel names cannot be transmitted because they are meaningless without the specification of the user owning them. Input channels cannot be variables and do not need to be decorated with the name of the user they belong to: this is a syntactic mean to locate channels.[1] On the other hand, output actions must indicate the name of the user containing the invoked channel. For example, $r\{\!\{a^s\langle \ldots \rangle.P\}\!\}_\rho$ models a user r trying to communicate along channel a associated to user s (if any). Notice also that a process like $a(x).b^x\langle v\rangle.P$ can be accepted but, in order to be executed, at run-time x must be assigned a user name r which owns an input channel b^r. These properties will be enforced by the type system of Section 3.

Restrictions $(\nu a : R)P$ and $(\nu a^r : R)A$ and the input prefix $a(x).P$ act as binders for channel name a, channel a^r and variable x, respectively. Furthermore, observe that user names cannot be restricted. This seems reasonable since the creation of a new user is a sensible operation; thus, it has to be performed by the system administrator and it affects the global policy underlying the system.

[1] This requirement could be implemented using global (i.e., not located) channels and having the type system enforce it [18, 8, 2]. In order to emphasize the new aspects of the calculus, we prefer to use syntactic constraints.

System	F (_)	B (_)
0	$\emptyset$	$\emptyset$
$r\{\!\!\{P\}\!\!\}_\rho$	F $_r(P)$	B $_r(P)$
$A \parallel B$	F $(A) \cup$ F (B)	B $(A) \cup$ B (B)
$(\nu a^r\!:\!R)A$	F $(A) - \{a^r\}$	B $(A) \cup \{a^r\}$

Process	F $_r$(_)	B $_r$(_)
nil	$\emptyset$	$\emptyset$
$a(x).P$	$\{a^r\} \cup$ F $_r(P)$	B $_r(P)$
$u\langle v\rangle.P$	$(\{u,v\} \cap C) \cup$ F $_r(P)$	B $_r(P)$
role $R.P$	F $_r(P)$	B $_r(P)$
yield $R.P$	F $_r(P)$	B $_r(P)$
$!P$	F $_r(P)$	B $_r(P)$
$P \parallel Q$	F $_r(P) \cup$ F $_r(Q)$	B $_r(P) \cup$ B $_r(Q)$
$(\nu a\!:\!R)P$	F $_r(P) - \{a^r\}$	B $_r(P) \cup \{a^r\}$
$[u = v]P$	$(\{u,v\} \cap C) \cup$ F $_r(P)$	B $_r(P)$

Table 2: Free and Bound Channels

2.2 Dynamic Semantics

The dynamics of the calculus is given in the form of a reduction relation. As customary, the reduction semantics is based on an auxiliary relation called structural congruence, $\equiv$, which brings the participants of a potential interaction to contiguous positions.

Since in our calculus only channels and channels names can be bound by restriction, we also need to modify the standard notion of free and bound names of the π–calculus to *free* and *bound channels*. The formal definition of functions F (A) and B (A) is given in Table 2; it exploits the auxiliary functions F $_r(P)$ and B $_r(P)$ for processes of user r. Notice that, when computing F $_r(u\langle v\rangle.P)$ and F $_r([u = v]P)$, we need $\{u,v\} \cap C$ because u or v can syntactically be variables or user names. Alpha-conversion is then standardly defined and it allows the renaming of bound channels, bound channel names and bound variables. Throughout the paper, we only consider closed terms (i.e. terms without unbound variables) and always assume that bound items are pair-ways distinct and different from the free ones; by using alpha-conversion, this requirement can be always satisfied.

Definition 2.1 (Structural Congruence). The structural congruence relation, $\equiv$, is the least congruence on systems equating alpha-convertible systems, stating that $\parallel$ is commutative and associative and has **0** as identity, and satisfying the following laws

$$r\{\!\!\{P|Q\}\!\!\}_\rho \equiv r\{\!\!\{P\}\!\!\}_\rho \parallel r\{\!\!\{Q\}\!\!\}_\rho \qquad r\{\!\!\{(\nu a\!:\!R)P\}\!\!\}_\rho \equiv (\nu a^r\!:\!R)r\{\!\!\{P\}\!\!\}_\rho$$

$$(\nu a^r\!:\!R)(\nu b^s\!:\!S)A \equiv (\nu b^s\!:\!S)(\nu a^r\!:\!R)A \quad (\nu a^r\!:\!R)(A \parallel B) \equiv (\nu a^r\!:\!R)A \parallel B \quad \text{if } a^r \notin \text{F } (B)$$

$$r\{\!\!\{!P\}\!\!\}_\rho \equiv r\{\!\!\{P \,|\, !P\}\!\!\}_\rho \qquad r\{\!\!\{[u = u]P\}\!\!\}_\rho \equiv r\{\!\!\{P\}\!\!\}_\rho$$

Definition 2.2 (Reduction Relation). The reduction relation, $\longmapsto$, is the least relation on systems satisfying the following laws

$$r\{\!\!\{a(x).P\}\!\!\}_\rho \parallel s\{\!\!\{a^r\langle n\rangle.Q\}\!\!\}_{\rho'} \longmapsto r\{\!\!\{P[n/x]\}\!\!\}_\rho \parallel s\{\!\!\{Q\}\!\!\}_{\rho'}$$

$$r\{\!\!\{\textbf{role}\, R.P\}\!\!\}_\rho \longmapsto r\{\!\!\{P\}\!\!\}_{\rho\cup\{R\}} \qquad r\{\!\!\{\textbf{yield}\, R.P\}\!\!\}_\rho \longmapsto r\{\!\!\{P\}\!\!\}_{\rho-\{R\}}$$

$$\frac{A \longmapsto A'}{A \parallel B \longmapsto A' \parallel B} \qquad \frac{A \longmapsto A'}{(\nu a^r\!:\!R)A \longmapsto (\nu a^r\!:\!R)A'} \qquad \frac{A \equiv A' \quad A' \longmapsto B' \quad B' \equiv B}{A \longmapsto B}$$

All structural rules are standard, but the first two. The first states that a session of user r with rôles ρ hosting two processes running in parallel can be split in two parallel sessions of r with rôles ρ. The

second one states that a restriction of a channel name inside a user can be turned into a restriction over the corresponding channel at the system level. Similarly, the reduction relation is an extension of [16] with the rules for actions **role** R and **yield** R. The first action adds R to the rôles ρ activated in the current session, while the second one removes R from ρ. Notice that, by exploiting the first structural rule and the rules for **role**/**yield**, the user $r\{\!|\mathbf{role}\, R.P \mid \mathbf{yield}\, S.Q|\!\}_\rho$ evolves into $r\{\!|P|\!\}_{\rho\cup\{R\}} \parallel r\{\!|Q|\!\}_{\rho-\{S\}}$, i.e. actions **role**/**yield** only affect the process thread executing them.

2.3 RBAC Schema

To conclude the presentation of the RBAC96 model, we need to define the *RBAC schema*, i.e., the rôles-to-users and permissions-to-rôles associations, where permissions enable the actions a user can perform within a system.

Managing rôles and their interrelationships is a difficult and sensitive task that is often centralized and delegated to a small team of security administrators. In our framework, the RBAC schema consists of a pair of finite relations $(\mathcal{U}\,;\mathcal{P})$, where $\mathcal{U}$ assigns rôles to users, while $\mathcal{P}$ assigns permissions to rôles. More formally,

$$\mathcal{U} \subseteq_{\text{fin}} (\mathcal{N}_u \cup C) \times \mathcal{R} \qquad\qquad \mathcal{P} \subseteq_{\text{fin}} \mathcal{R} \times \mathcal{A}$$

where $\mathcal{A} \triangleq \{R^{\uparrow}, R^{?}, R^{!}\}_{R\in\mathcal{R}}$ represents the set of performable actions. Intuitively, permission $R^{\uparrow}$ determines the possibility to activate rôle R (via the action **role**), while permissions $R^{?}$ and $R^{!}$ determine the possibility of performing input and output actions over a channel of rôle R, respectively. Notice that permissions over input/output actions are not defined in terms of channels, but of channel rôles. In this way, we are flexible enough to model both the permission to communicate over a single channel (i.e., when the relation $\mathcal{U}$ maps only one channel to a rôle), and the permission to communicate over the member of a group of channels (i.e., when relation $\mathcal{U}$ maps more than one channel to the same rôle). Such a case may be useful in situations where more channels can handle the same kind of requests (see Example 1 for a possible situation). Observe that, if $\mathcal{U}$ assigns rôle R to a channel, then the fact that R is assigned certain permissions by $\mathcal{P}$ is irrelevant. Moreover, since channels can be considered as methods provided by users, it seems reasonable that each channel is assigned only one rôle. A RBAC schema satisfying this last requirement is called *well-defined*; in the rest of the paper we shall only consider well-defined RBAC schemas.

In the definition of set $\mathcal{A}$ no permission represents actions **yield**. Indeed, we assume that a rôle can be deactivated if (and only if) it has been activated before.

To conclude the presentation of our language, we now give a couple of examples using the features introduced so far. Here and in the rest of the paper, we use a number of notational conventions. We write $\mathcal{U}(\_)$ to denote the set of all the rôles R such that $(\_, R) \in \mathcal{U}$; similarly, we call the *domain* of $\mathcal{U}$ its left projection. Moreover, we let $\mathcal{P}(\rho)$ to mean $\bigcup_{R\in\rho} \mathcal{P}(R)$.

Example 1. Let us now formalize in our framework a scenario where a bank client is waiting to be served by one of the branch cashiers available. There are two users, r and s, representing respectively the client and the bank branch, while cashiers are modelled as channels belonging to user s, named $c_1, \ldots, c_n$. The rôles available are `client` and `branch_cashier`. Relation $\mathcal{U}$ assigns rôle `client` to user r and `branch_cashier` to channels c_i, while $\mathcal{P}$ assigns to `client` the permission to communicate with any of the cashiers, i.e., $(\texttt{client}, \texttt{branch\_cashier}^{!}) \in \mathcal{P}$. In this way, r can indistinctly activate any of the cashier methods. The overall system can be described as follows (where Π is a shorthand

for parallel composition):

$$r\{\!|\textbf{role}\ \texttt{client}.\mathit{enqueue}^s\langle r\rangle.\mathit{dequeue}(z).z\langle \mathit{req}_1\rangle.\cdots.z\langle \mathit{req}_k\rangle.z\langle \mathit{stop}\rangle.\textbf{yield}\ \texttt{client}|\!\}_\rho \quad \|$$
$$s\{\!|(\nu\, \mathit{free})(!\mathit{enqueue}(x).\mathit{free}(y).\mathit{dequeue}^x\langle y\rangle \quad | \quad \Pi_{i=1}^n \mathit{free}^s\langle c_i^s\rangle \quad |$$
$$\Pi_{i=1}^n\ !c_i(x).(\ [x = \mathit{withdrw\_req}] <\text{handle withdraw request}> \ |$$
$$[x = \mathit{dep\_req}] <\text{handle deposit request}> \ | \ \ldots \ | \ [x = \mathit{stop}]\mathit{free}^s\langle c_i^s\rangle)\ )|\!\}_{\rho'}$$

Once the client enters the bank (i.e., she activates rôle `client`), she queues up and waits to be served. When one of the cashiers becomes available (information maintained locally by the bank via the reserved channel *free*), the client is notified and can make requests along the received channel *z*. Cashiers repeatedly receive requests; we at least assume methods to handle money withdraw and deposit (for simplicity, we do not consider the order in which clients arrive; a system of queues can however be added routinely).

Example 2 (Prerequisite rôle). In some circumstances, one may want to require a rôle to be activated only by a user already playing a certain rôle. This is a particular model of constrained RBAC called *prerequisite rôle* (see, e.g., [17]). In the banking scenario of Example 1, imagine that r is member of `client`, `user` and `authenticated_user` rôles, and that the bank policy requires a preliminary authentication phase to identify its clients. This can be implemented by having $(\texttt{authenticated\_user}, \texttt{client}^{\uparrow}) \in \mathcal{P}$; hence `authenticated_user` must be present in ρ to enable the evolution of r given above.

Example 2 shows that some form of 'default' rôle may be needed to kickstart users' activities. Hence, ρ in $r\{\!|P|\!\}_\rho$ is used both to record the rôles activated in the session and to assign some default rôles to r at the outset.

3 Static Semantics

The type system described below provides static guarantees that the set of actions performed by any user during the computation respects the RBAC schema, given an initial set ρ of activated rôles. The syntax of types can be defined by the following productions (with $\widetilde{\_}$ denoting a tuple of entities of kind $\_$):

$$\begin{array}{llll} \textit{Message Types} & T & ::= & \rho[\widetilde{a : C}] \mid C \\ \textit{Channel Types} & C & ::= & R(T) \end{array}$$

Type $\rho[a_1 : R_1(T_1), \ldots, a_n : R_n(T_n)]$ can be assigned to a user r belonging to rôles in ρ and owning channels $\widetilde{a^r}$ of type $\widetilde{R(T)}$. Type $R(T)$ can be assigned to channels exchanging values of type T and belonging to rôle R.

A *typing environment* Γ is a finite partial mapping from $\mathcal{N}_u \cup \mathcal{V}$ into types; thus we write $\Gamma(\_) = T$ to refer to the type T of the user name or variable $\_$. A typing environment can be extended as follows:

$$\Gamma, x : T \triangleq \Gamma \uplus \{x : T\}$$
$$\Gamma, a^r : C \triangleq \Gamma' \ \text{ where } \ \Gamma'(s) = \begin{cases} \Gamma(s) & \text{if } s \neq r \\ \rho[a : C, \widetilde{b : C'}] & \text{if } s = r \text{ and } \Gamma(r) = \rho[\widetilde{b : C'}] \text{ and } a \notin \widetilde{b} \end{cases}$$

In the rest of the paper, we denote with $\uplus$ the union of functions/relations with disjoint domains. A typing environment Γ can be used to type a system under a schema $(\mathcal{U}; \mathcal{P})$ only if the rôle information in Γ respects the associations in $\mathcal{U}$. This intuition is formalized by the following definition.

Definition 3.1. Given a RBAC schema $(\mathcal{U};\mathcal{P})$ and a typing environment Γ, we say that Γ *respects* $\mathcal{U}$ if, for all $r \in dom(\Gamma)$ with $\Gamma(r) = \rho[a_1 : R_1(T_1), \ldots, a_n : R_n(T_n)]$, it holds that $\mathcal{U}(r) = \rho$ and $\mathcal{U}(a_i^r) = \{R_i\}$, for all $i = 1, \ldots, n$.

The primary judgments of the type system are of the form $\Gamma \vdash^{\mathcal{P}} A$, that should be read as "the system A is well-formed with respect to environment Γ and relation $\mathcal{P}$". This fact, together with the requirement that Γ respects $\mathcal{U}$, implies that A respects the RBAC schema $(\mathcal{U};\mathcal{P})$. To infer the main judgment, we rely on two auxiliary judgments, one for identifiers and one for processes. Judgment $\Gamma \vdash u : T$ states that the identifier u has type T in Γ; judgment $\Gamma;\rho \vdash_r^{\mathcal{P}} P$ states that P respects Γ and $\mathcal{P}$ when it is run in a session of r with rôles ρ activated.

The typing rules are collected in Table 3. Most of them are self-explanatory; we comment below the most significant ones, i.e. those related to the actions in our calculus. The underlying idea beyond these rules is that an action can be executed only if the current session has activated a rôle enabling the action. Rule (T-I) states that, for typing $a(x).P$ in a session of r where rôles ρ are activated, we need to establish that a^r has type $R(T)$ in Γ, that inputs over a channel of group R can be performed when playing rôles ρ and that P is typeable once assumed that x has type T. Rule (T-O) is similar: it checks that an output over a channel of group R is allowed when rôles in ρ are activated. Moreover, it also requires that the transmitted value v can be assigned type T in Γ. Rule (T-R^) states that for typing process **role** $R.P$ in a session of r where rôles ρ are activated, we need to check that r can assume rôle R, that ρ enables the activation and that P is typeable for r having activated $\rho \cup \{R\}$. Rule (T-Y) states that process **yield** $R.P$ is legal for r only when R has been previously activated and if P is typeable for r when R is off.

Definition 3.2 (Well-typedness). Given a RBAC schema $(\mathcal{U};\mathcal{P})$ and a system A, we say that A is *well-typed for* $(\mathcal{U};\mathcal{P})$ if there exists a typing environment Γ respecting $\mathcal{U}$ such that $\Gamma \vdash^{\mathcal{P}} A$.

We now prove the soundness of the type system in the standard way, i.e., by proving subject reduction and type safety, which ensure that only systems abiding by the RBAC schema are allowed (i.e., users performing actions permitted by their duly activated rôles).

Theorem 3.1 (Subject Reduction). *If* $\Gamma \vdash^{\mathcal{P}} A$ *and* $A \longmapsto A'$, *then* $\Gamma \vdash^{\mathcal{P}} A'$.

Proof. A rather standard proof by induction on the length of the reduction from A to A'. In the inductive case, the reasoning is by a cases analysis of the reduction in question. □

Theorem 3.2 (Type Safety). *Let A be a well-typed system for $(\mathcal{U};\mathcal{P})$. Then*

1. whenever $A \equiv (\nu\, \widetilde{a^r{:}R})(A' \parallel r\{\!|P|\!\}_\rho)$, it holds that $\rho \subseteq \mathcal{U}(r)$
2. whenever $A \equiv (\nu\, \widetilde{a^r{:}R})(A' \parallel r\{\!|\mathbf{role}\, R.P|\!\}_\rho)$, it holds that $R \in \mathcal{U}(r)$ and $R^{\uparrow} \in \mathcal{P}(\rho)$
3. whenever $A \equiv (\nu\, \widetilde{a^r{:}R})(A' \parallel r\{\!|\mathbf{yield}\, R.P|\!\}_\rho)$, it holds that $R \in \rho$
4. whenever $A \equiv (\nu\, \widetilde{a^r{:}R})(A' \parallel r\{\!|b(x).P|\!\}_\rho)$, it holds that either $b^r{:}S \in \widetilde{a^r{:}R}$ and $S^? \in \mathcal{P}(\rho)$, or $b^r \notin \widetilde{a^r}$ and $S^? \in \mathcal{P}(\rho)$, where $\{S\} = \mathcal{U}(b^r)$
5. whenever $A \equiv (\nu\, \widetilde{a^r{:}R})(A' \parallel r\{\!|b^s\langle n\rangle.P|\!\}_\rho)$, it holds that either $b^s{:}S \in \widetilde{a^r{:}R}$ and $S^! \in \mathcal{P}(\rho)$, or $b^s \notin \widetilde{a^r}$ and $S^! \in \mathcal{P}(\rho)$, where $\{S\} = \mathcal{U}(b^s)$

Proof. By definition, there exists an environment Γ respecting $\mathcal{U}$ such that $\Gamma \vdash^{\mathcal{P}} A$. The proof proceeds by a simple induction on the length of the inference for this judgment. □

Typing Identifiers:

(T-I $_1$)
$$\frac{\Gamma(\_) = \rho[\widetilde{a : C}] \qquad \_ \in \mathcal{N}_u \cup \mathcal{V}}{\Gamma \vdash \_ : \rho[\widetilde{a : C}]}$$

(T-I $_2$)
$$\frac{\Gamma(\_) = \rho[a : C, \widetilde{b : C'}] \qquad \_ \in \mathcal{N}_u \cup \mathcal{V}}{\Gamma \vdash a^{\_} : C}$$

Typing Processes:

(T-N)
$$\frac{}{\Gamma; \rho \vdash^{\mathcal{P}}_r \mathbf{nil}}$$

(T-I)
$$\frac{\Gamma \vdash a^r : R(T) \quad R^? \in \mathcal{P}(\rho) \quad \Gamma, x : T; \rho \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r a(x).P}$$

(T-P)
$$\frac{\Gamma; \rho \vdash^{\mathcal{P}}_r P \quad \Gamma; \rho \vdash^{\mathcal{P}}_r Q}{\Gamma; \rho \vdash^{\mathcal{P}}_r P \mid Q}$$

(T-O)
$$\frac{\Gamma \vdash u : R(T) \quad \Gamma \vdash v : T \quad R^! \in \mathcal{P}(\rho) \quad \Gamma; \rho \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r u\langle v \rangle.P}$$

(T-B)
$$\frac{\Gamma; \rho \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r !P}$$

(T-R)
$$\frac{\Gamma \vdash r : \rho'[\widetilde{a : C}] \quad R \in \rho' \quad R^\uparrow \in \mathcal{P}(\rho) \quad \Gamma; \rho \cup \{R\} \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r \mathbf{role}\, R.P}$$

(T-R)
$$\frac{\Gamma, a^r : R(T); \rho \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r (\nu a : R)P}$$

(T-Y)
$$\frac{R \in \rho \qquad \Gamma; \rho - \{R\} \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r \mathbf{yield}\, R.P}$$

(T-M)
$$\frac{\Gamma; \rho \vdash^{\mathcal{P}}_r P}{\Gamma; \rho \vdash^{\mathcal{P}}_r [u = v]P}$$

Typing Systems:

(T-E)
$$\frac{}{\Gamma \vdash^{\mathcal{P}} \mathbf{0}}$$

(T-S)
$$\frac{\Gamma \vdash r : \rho'[\widetilde{a : C}] \quad \rho \subseteq \rho' \quad \Gamma; \rho \vdash^{\mathcal{P}}_r P}{\Gamma \vdash^{\mathcal{P}} r\{\!|P|\!\}_\rho}$$

(T-S P)
$$\frac{\Gamma \vdash^{\mathcal{P}} A \qquad \Gamma \vdash^{\mathcal{P}} B}{\Gamma \vdash^{\mathcal{P}} A \parallel B}$$

(T-S R)
$$\frac{\Gamma, a^r : R(T) \vdash^{\mathcal{P}} A}{\Gamma \vdash^{\mathcal{P}} (\nu a^r : R)A}$$

Table 3: Typing Rules

Example 3. Let us consider the banking scenario again. As described before, the system is modelled by having two users *r* and *s*, i.e. the client and the bank respectively. In the real world, it is unrealistic to allow any bank client to ask for any kind of bank operation: e.g., when a client requires a credit card she is always asked for some credentials. To model this finer scenario, we let each available operation to be modelled as a specific method, activable through a specific channel (e.g., channel *wdrw* handles withdraw requests, *opn* handles open account requests, *cc* handles credit card requests, etc.). The communication along different channels requires different rôles and, thus, it is a way to control the credentials of the client. In this setting, the cashier c_i of Example 1 is implemented by the following

process (the remaining behaviour of the bank is implemented like in Example 1):

$$c_i(x).(\; [x = \mathit{withdrw\_req}]\, \mathit{wdrw}(y).\cdots \quad | \quad [x = \mathit{open\_req}]\, \mathit{opn}(y).\cdots \quad | \\ [x = \mathit{creditcard\_req}]\, \mathit{cc}(y).\cdots \quad | \quad \ldots \quad | \quad [x = \mathit{stop}]\, \mathit{free}^s\langle c_i^s \rangle \;)$$

Let relation $\mathcal{U}$ assign channel *wdrw* (resp., *opn* and *cc*) the group `wdrw` (resp., `opn` and `cc`), and $\mathcal{P}$ be such that $\{(\texttt{rich\_client}, \texttt{cc}^!)\,, (\texttt{client}, \texttt{wdrw}^!)\,, (\texttt{user}, \texttt{opn}^!)\,, (\texttt{user}, \texttt{client}^\uparrow)\,, (\texttt{rich}, \texttt{rich\_client}^\uparrow)\} \subseteq \mathcal{P}$. Thus, under this schema, the client

$$r\{\!|\textbf{role}\ \texttt{client}.\mathit{enqueue}^s\langle r\rangle.\mathit{dequeue}(z).z\langle \mathit{creditcard\_req}\rangle.\mathit{cc}^s\langle \mathit{signature}\rangle.z\langle \mathit{stop}\rangle|\!\}_{\{\texttt{user}\}}$$

is not well-typed because he has not activated the correct rôle for performing credit card requests. On the other hand, the following clients do type-check:

$$r_1\{\!|\textbf{role}\ \texttt{rich\_client}.\mathit{enqueue}^s\langle r\rangle.\mathit{dequeue}(z).z\langle \mathit{creditcard\_req}\rangle.\mathit{cc}^s\langle \mathit{signature}\rangle.z\langle \mathit{stop}\rangle|\!\}_{\{\texttt{rich}\}}$$
$$r_2\{\!|\textbf{role}\ \texttt{client}.\mathit{enqueue}^s\langle r\rangle.\mathit{dequeue}(z).z\langle \mathit{withdrw\_req}\rangle.\mathit{wdrw}^s\langle \mathit{sum}\rangle.z\langle \mathit{stop}\rangle|\!\}_{\{\texttt{user}\}}$$
$$r_3\{\!|\mathit{enqueue}^s\langle r\rangle.\mathit{dequeue}(z).z\langle \mathit{open\_req}\rangle.\mathit{opn}^s\langle \mathit{personal\_data}\rangle.z\langle \mathit{stop}\rangle|\!\}_{\{\texttt{user}\}}$$

4 Observational Semantics

Often, the overall structure of a distributed system cannot be known statically. Thus, the typing approach described in the previous section, even if interesting from a theoretical point of view, may not be usable in practice. Thus, in this section, we introduce a labelled transition system (LTS, for short) equipped with dynamic checks that allows us to study (not necessarily well-typed) system components in isolation and compositionally. The LTS and its dynamic checks also provide a tight operational model for the minimal engine underlying any implementation of a RBAC-based run-time system. Then, we define a standard bisimulation over the LTS and show that it is adequate with a typed barbed congruence, a relevant result in at least two aspects. First, it signifies that the bisimulation is a sensible equivalence to consider, as it agrees with the (typed) contextual semantics derived from an elementary, natural class of observables. Second, it provides us with a powerful co-inductive proof technique for barbed congruence.

The standard way to describe the interactions a system can externally offer is by labelling the system evolution with this information. Thus, we define a labelled transition system, $\xrightarrow{\mu}$, that makes apparent the action performed (and, thus, the external interaction offered) to evolve. Since we do not require $\xrightarrow{\mu}$ to relate only well-typed systems, the LTS comes equipped with some runtime checks w.r.t. the RBAC schema considered to block the execution of illegal actions.

The LTS evolves from the π–calculus' early-style transition system. In order to account for the systems' rôles varying over time, the LTS relates *configurations*, i.e. pairs $(\mathcal{U}; \mathcal{P}) \triangleright A$ made up of a RBAC schema $(\mathcal{U}; \mathcal{P})$ and a system A. Configurations are ranged over by $D, E, \ldots$. The labels of the LTS are derived from those of the π–calculus and can be described as follows.

$$\mu \quad ::= \quad \tau \quad | \quad a^r n \quad | \quad a^r n : R \quad | \quad \overline{a}^r n \quad | \quad \overline{a}^r n : R$$

Label τ represents an internal computation of the system. Labels $\overline{a}^r n$ and $a^r n$ describe the intention of sending/receiving the value n, known to the environment, on/from channel a^r. Labels $\overline{a}^r n : R$ and $a^r n : R$ are similar to but the value sent/received is new (i.e. unknown to the environment) and has

group R. We easily extend functions F $(\_)$ and B $(\_)$ to labels.

Label	F $(\_)$	B $(\_)$
τ	$\emptyset$	$\emptyset$
$a^r n$	$\{a^r, n\} \cap C$	$\emptyset$
$\overline{a}^r n$	$\{a^r, n\} \cap C$	$\emptyset$
$a^r n : R$	$\{a^r\}$	$\{n\} \cap C$
$\overline{a}^r n : R$	$\{a^r\}$	$\{n\} \cap C$

The rules defining $\xrightarrow{\mu}$ are given in Table 4, where we write $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U};\mathcal{P}) \triangleright B$ as $A \xrightarrow{\mu} B$ whenever the schema is clear from the context or is irrelevant. The general implant of the system is similar to π–calculus' (see, e.g., [19]). We introduced rules (LTS-S), (LTS-E), (LTS-M), (LTS-B) and the symmetric versions of rules (LTS-C), (LTS-P) and (LTS-C) to avoid structural congruence; however, we still implicitly assume alpha conversion. The premises of rules (LTS-K-I), (LTS-F-I), (LTS-O), (LTS-R$\hat{}$) and (LTS-Y) adapt, resp., the premises of the typing rules (T-I), (T-O), (T-R$\hat{}$) and (T-Y) and block the evolution of ill-typed systems. Rule (LTS-K-I) can be applied when the received value is known to the schema, while (LTS-F-I) is used when a fresh value (i.e. unknown to the schema) is received. In this case, the schema is extended to record the group of the fresh value. Similarly, when extruding a restricted channel b^s, rule (LTS-O) enlarges the relation $\mathcal{U}$ of the current configuration by recording that b^s has the rôle declared in the restriction. The information about a fresh/extruded channel is deleted from the schema when the channel is communicated: indeed, the restriction is pushed back in the system and closes the scope of the channel (see rule (LTS-C)). Notice that a bound output can synchronize only with a fresh input (and vice versa), and the rôle declared for the extruded/fresh channel must be the same.

The semantics given in Definition 2.2 and the LTS just presented are related by the following

Proposition 4.1. *If* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\tau} (\mathcal{U};\mathcal{P}) \triangleright A'$ *then* $A \longmapsto A'$. *Moreover, if* A *is well-typed for* $(\mathcal{U};\mathcal{P})$, *then* $A \longmapsto A'$ *implies* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\tau} (\mathcal{U};\mathcal{P}) \triangleright B$ *for some* $B \equiv A'$.

Proof. The first statement is proved by a simple induction over the length of the inference for $\xrightarrow{\tau}$. The second statement is proved by induction over the length of the inference for $\longmapsto$. The most complicate case is when the last rule applied to infer the reduction involves structural congruence. In this case, we need a second induction, over the length of the inference for the structural judgment: all the cases are simple. Just notice that, when considering context closure, we need a further induction over the structure of the system context. □

Also observe that τ-moves do not modify the schema $(\mathcal{U};\mathcal{P})$. We build upon this LTS a standard bisimulation. As usual, $\Longrightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\mu}{\Longrightarrow}$ denotes $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$. Finally, $\overset{\hat{\mu}}{\Longrightarrow}$ is $\Longrightarrow$ if $\mu = \tau$, and $\overset{\mu}{\Longrightarrow}$ otherwise.

Definition 4.1 (Bisimilarity). A *bisimulation* is a binary symmetric relation $\mathcal{S}$ between configurations such that, if $(D, E) \in \mathcal{S}$ and $D \xrightarrow{\mu} D'$, there exists a configuration E' such that $E \overset{\hat{\mu}}{\Longrightarrow} E'$ and $(D', E') \in \mathcal{S}$. *Bisimilarity*, $\approx$, is the largest bisimulation.

We now state some properties of $\approx$; all the proofs are in Appendix A. As often happens in typed calculi, $\approx$ is *not* a congruence w.r.t. all system contexts. Indeed, due to the checks in the LTS for

(LTS-R$\hat{}$)
$$\frac{R \in \mathcal{U}(r) \qquad R^{\uparrow} \in \mathcal{P}(\rho)}{r\{\!|\mathbf{role}\, R.P|\!\}_\rho \xrightarrow{\tau} r\{\!|P|\!\}_{\rho \cup \{R\}}}$$

(LTS-Y)
$$\frac{R \in \rho}{r\{\!|\mathbf{yield}\, R.P|\!\}_\rho \xrightarrow{\tau} r\{\!|P|\!\}_{\rho - \{R\}}}$$

(LTS-K-I)
$$\frac{\mathcal{U}(a^r) = \{R\} \quad R^{?} \in \mathcal{P}(\rho) \quad n \in dom(\mathcal{U})}{r\{\!|a(x).P|\!\}_\rho \xrightarrow{a^r n} r\{\!| P[n/x] |\!\}_\rho}$$

(LTS-O)
$$\frac{\mathcal{U}(a^s) = \{R\} \qquad R^{!} \in \mathcal{P}(\rho)}{r\{\!|a^s\langle n\rangle.P|\!\}_\rho \xrightarrow{\overline{a}^s n} r\{\!|P|\!\}_\rho}$$

(LTS-C)
$$\frac{A \xrightarrow{a^r n} A' \qquad B \xrightarrow{\overline{a}^r n} B'}{A \parallel B \xrightarrow{\tau} A' \parallel B'}$$

(LTS-F-I)
$$\frac{\mathcal{U}(a^r) = \{R\} \qquad R^{?} \in \mathcal{P}(\rho) \qquad n \notin dom(\mathcal{U})}{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|a(x).P|\!\}_\rho \xrightarrow{a^r n:S} (\mathcal{U} \uplus \{n : S\}; \mathcal{P}) \triangleright r\{\!| P[n/x] |\!\}_\rho}$$

(LTS-O)
$$\frac{(\mathcal{U} \uplus \{b^s{:}S\}; \mathcal{P}) \triangleright A \xrightarrow{\overline{a}^r b^s} (\mathcal{U} \uplus \{b^s{:}S\}; \mathcal{P}) \triangleright A' \qquad a^r \neq b^s}{(\mathcal{U}; \mathcal{P}) \triangleright (\nu b^s{:}S)A \xrightarrow{\overline{a}^r b^s:S} (\mathcal{U} \uplus \{b^s{:}S\}; \mathcal{P}) \triangleright A'}$$

(LTS-C)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright A \xrightarrow{a^r b^s:S} (\mathcal{U}'; \mathcal{P}) \triangleright A' \qquad (\mathcal{U}; \mathcal{P}) \triangleright B \xrightarrow{\overline{a}^r b^s:S} (\mathcal{U}'; \mathcal{P}) \triangleright B'}{(\mathcal{U}; \mathcal{P}) \triangleright A \parallel B \xrightarrow{\tau} (\mathcal{U}; \mathcal{P}) \triangleright (\nu b^s{:}S)(A' \parallel B')}$$

(LTS-R)
$$\frac{(\mathcal{U} \uplus \{a^r{:}R\}; \mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}' \uplus \{a^r{:}R\}; \mathcal{P}) \triangleright A' \qquad a^r \notin \mathrm{F}(\mu)}{(\mathcal{U}; \mathcal{P}) \triangleright (\nu a^r{:}R)A \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright (\nu a^r{:}R)A'}$$

(LTS-P)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A' \qquad \mathrm{B}(\mu) \cap \mathrm{F}(B) = \emptyset}{(\mathcal{U}; \mathcal{P}) \triangleright A \parallel B \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A' \parallel B}$$

(LTS-E)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright (\nu a^r{:}R)r\{\!|P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|(\nu a{:}R)P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}$$

(LTS-S)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|P|\!\}_\rho \parallel r\{\!|Q|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|P|Q|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}$$

(LTS-M)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|[u = u]P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}$$

(LTS-B)
$$\frac{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|P \mid !P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}{(\mathcal{U}; \mathcal{P}) \triangleright r\{\!|!P|\!\}_\rho \xrightarrow{\mu} (\mathcal{U}'; \mathcal{P}) \triangleright A}$$

plus the symmetric version of rules of (LTS-P), (LTS-C) and (LTS-C)

Table 4: A Labelled Transition System

schema compliance, the application of ill-typed contexts can break equivalences. As an example, consider $A \triangleq r\{\!|a(x).b^r\langle\cdot\rangle \mid a^r\langle\cdot\rangle|\!\}_\rho$ and $B \triangleq \mathbf{0}$, and let $a^r \notin dom(\mathcal{U})$, $\mathcal{U}(b^r) = \{R\}$ and $R^?, R^! \in \mathcal{P}(\rho)$. Then $(\mathcal{U};\mathcal{P}) \triangleright A \approx (\mathcal{U};\mathcal{P}) \triangleright B$ but $(\mathcal{U};\mathcal{P}) \triangleright (\nu a^r : R)A \not\approx (\mathcal{U};\mathcal{P}) \triangleright (\nu a^r : R)B$. A similar problem arises also if $\mathcal{U}(a^r) = \{S\}$ but $S^?, S^! \notin \mathcal{P}(\rho)$. Moreover, some care must be spent when the configurations equated rely on different schemas. Indeed, it is easy to find a situation where $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A_1 \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright A_2$ but $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A_1 \parallel B \not\approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright A_2 \parallel B$ (it suffices to find a system B with an action enabled by $\mathcal{P}_1$ but disabled by $\mathcal{P}_2$).

Theorem 4.2 (Congruence Properties of $\approx$). *The following facts hold:*

1. *if* $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A_1 \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright A_2$ *and* $(\mathcal{U}_1;\mathcal{P}_1) \triangleright B \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright B$, *then* $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A_1 \parallel B \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright A_2 \parallel B$
2. *if* $(\mathcal{U}_1 \uplus \{a^r : R\};\mathcal{P}_1) \triangleright A_1 \approx (\mathcal{U}_2 \uplus \{a^r : R\};\mathcal{P}_2) \triangleright A_2$, *then* $(\mathcal{U}_1;\mathcal{P}_1) \triangleright (\nu a^r : R)A_1 \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu a^r : R)A_2$

This bisimulation is a sound semantic equivalence, in the sense that it produces no unreasonable equations. To substantiate this claim, we prove its adequacy for a standardly defined typed observational congruence. Namely, we introduce *reduction barbed congruence*, a touchstone equivalence defined in terms of the reduction relation and of a notion of observability, and that is closed under all possible system contexts [12]. The reason to consider a typed congruence is that only well-typed contexts guarantee a reduction behaviour abiding by the RBAC policy. Indeed, the reduction relation performs none of the legality checks hardcoded in the LTS. Hence, the right framework for comparison of $\approx$ and a barbed congruence is a typed one.

In its typed version, barbed congruence is tagged with an environment Γ and a permissions-to-rôles assignment $\mathcal{P}$ to mean that the systems equated are typeable under Γ and $\mathcal{P}$. Moreover, only contexts typeable under Γ and $\mathcal{P}$ are considered in the definition of congruence. Thus, following the style of [10], we write $\Gamma \models^{\mathcal{P}} A_1 \cong A_2$ to mean that $\Gamma \vdash^{\mathcal{P}} A_i$ for $i = 1, 2$ and that A_1 and A_2 exhibit the same behaviour when run in an environment constrained by Γ and $\mathcal{P}$.

Definition 4.2 (Barbs). The *observation predicate* $A \downarrow \eta$ holds true whenever either $\eta = a^r$ and $A \equiv (\nu\, \widetilde{a^r : R})(A' \parallel r\{\!|a(x).P|\!\}_\rho)$ for $a^r \notin \widetilde{a^r}$, or $\eta = \overline{a^r}$ and $A \equiv (\nu\, \widetilde{a^r : R})(A' \parallel s\{\!|a^r\langle n\rangle.P|\!\}_\rho)$ for $a^r \notin \widetilde{a^r}$. The predicate $A \Downarrow \eta$ is defined as $\exists A'.\, A \Longrightarrow A' \wedge A' \downarrow \eta$.

Definition 4.3 (Reduction Barbed Congruence). *Reduction barbed congruence* is the largest binary and symmetric typed relation over systems such that, whenever $\Gamma \models^{\mathcal{P}} A_1 \cong A_2$, the following properties do hold:

1. *Barb Preservation:* if $A_1 \downarrow \eta$, then $A_2 \Downarrow \eta$
2. *Reduction Closure:* if $A_1 \longmapsto A_1'$, then there exists a system A_2' such that $A_2 \Longrightarrow A_2'$ and $\Gamma \models^{\mathcal{P}} A_1' \cong A_2'$
3. *Contextuality:*
   (a) for all $\mathcal{P}'$ and $\widetilde{u : T}$ such that $\Gamma, \widetilde{u : T}$ is defined, it holds that $\Gamma, \widetilde{u : T} \models^{\mathcal{P} \cup \mathcal{P}'} A_1 \cong A_2$
   (b) for all systems B such that $\Gamma \vdash^{\mathcal{P}} B$ it holds that $\Gamma \models^{\mathcal{P}} A_1 \parallel B \cong A_2 \parallel B$
   (c) for all $a^r : R(T)$ such that $\Gamma = \Gamma', a^r : R(T)$, it holds that $\Gamma' \models^{\mathcal{P}} (\nu a^r : R)A_1 \cong (\nu a^r : R)A_2$

Before comparing $\approx$ and $\cong$, we remark that the chosen barbs only express the ability to interact over channels. Indeed, observing rôle activations/deactivations is not reasonable, because no context can determine whether a user can perform a **role**/**yield**: these operations only affect the thread performing them.

The fact that $\approx$ approximates $\cong$ only holds for well-typed configurations, i.e. configurations $(\mathcal{U};\mathcal{P}) \triangleright A$ such that A is well-typed for $(\mathcal{U};\mathcal{P})$. Given a typing environment Γ, we let $\mathcal{U}_\Gamma$ be the rôles-to-users assignment extracted from Γ, that is the least assignment such that, for any association $r : \rho[a : \widetilde{R(T)}]$ in Γ, it holds that $\mathcal{U}_\Gamma(r) = \rho$ and $\mathcal{U}_\Gamma(a^r) = \{R\}$ for any $a : R(T) \in a : \widetilde{R(T)}$.

Theorem 4.3 (Soundness of $\approx$). *Let $\Gamma \vdash^{\mathcal{P}} A$ and $\Gamma \vdash^{\mathcal{P}} B$. If $(\mathcal{U}_\Gamma;\mathcal{P}) \triangleright A \approx (\mathcal{U}_\Gamma;\mathcal{P}) \triangleright B$ then $\Gamma \models^{\mathcal{P}} A \cong B$.*

Theorem 4.3 shows that $\approx$ is a sound, while being effective, proof-technique for barbed congruence, that, on the contrary, is very hard to prove because of the contextuality requirement. We leave as a future work the development of finer techniques (as e.g. in [9, 15]) to prove the converse of Theorem 4.3, i.e. that bisimilarity is complete for barbed congruence.

To conclude, we now list some algebraic laws that illustrate the impact of RBAC on the π–calculus. In what follows, we fix a RBAC schema $(\mathcal{U};\mathcal{P})$. The first equation states that a terminated session of a user does not affect the evolution of a system. Indeed, it holds that

$$r\{\!|\mathbf{nil}|\!\}_\rho \approx \mathbf{0}$$

This is different from some distributed calculi, like, e.g., the Ambient calculus [5], where the presence of a user is relevant. Moreover, by letting α to range over action prefixes (i.e. inputs/outputs and **role**/**yield**), it holds that

$$r\{\!|\alpha.P|\!\}_\rho \approx \mathbf{0}$$

whenever α is not legal for a session $r\{\!|\cdot|\!\}_\rho$ w.r.t. the RBAC schema (i.e. if the premises of rules (LTS-R$\hat{}$), (LTS-Y), (LTS-K-I), (LTS-F-I) and (LTS-O), resp., are not satisfied). This law stresses the fact that the LTS and the types enforce the same requirements (compare the runtime checks of the LTS with Theorem 3.2). As a consequence, the following law differentiates our language from the π–calculus. Indeed, it holds that

$$(\nu a^r : R)(r\{\!|a(x).P|\!\}_\rho \parallel s\{\!|a^r\langle n\rangle.Q|\!\}_{\rho'}) \approx \mathbf{0},$$

whenever $R^? \notin \mathcal{P}(\rho)$ or $R^! \notin \mathcal{P}(\rho')$.

Differently from several distributed languages, the user performing an output action is irrelevant. The only relevant aspect is the set of permissions activated when performing the action. This is summarized in the following law:

$$r\{\!|b^s\langle n\rangle.\mathbf{nil}|\!\}_\rho \approx t\{\!|b^s\langle n\rangle.\mathbf{nil}|\!\}_\rho.$$

A similar situation also holds when the prefix is a **yield** action. On the contrary, relocating an input action usually breaks the equivalence between processes. In particular, it holds that

$$r\{\!|a(x).P|\!\}_\rho \approx t\{\!|a(x).P|\!\}_\rho$$

only if both input actions are disabled (and in this case both the systems are equivalent to **0**). Similarly, it is possible to move a **role** R prefix only when R is assigned to both or to none of the users involved. By exploiting these observations, we can find a relocation procedure to minimize the number of users in a system, while maintaining the system overall behaviour, as it will be described in the next section.

5 Applications

In this section, we exploit the theory developed so far in three non-trivial applications of the RBAC model. The first one deals with the problem of finding the 'minimal' schema to execute a given system. The second one is somehow symmetric: given a schema and a system, we aim at arranging **role**/**yield** operations within the system so that the resulting system can be executed w.r.t. the given schema (if possible). Finally, we give a simple but efficient procedure to determine whether a process can be executed by different users without compromising the functionality of the system. This can be useful to minimize the number of users in a system, while maintaining the overall system behaviour.

Minimal Schema. Let A be a system, $(\mathcal{U};\mathcal{P})$ a RBAC schema and A well-typed for $(\mathcal{U};\mathcal{P})$. Potentially, there are infinitely many schemas under which the system can run correctly; thus, it seems reasonable to look for a 'minimal' schema to execute the system. To this aim, we define the set $CONF_A = \{(\mathcal{U}';\mathcal{P}') \triangleright A : (\mathcal{U}';\mathcal{P}') \text{ is a RBAC schema}\}$ of configurations for A. Even if potentially infinite, the set $CONF_A$ can be effectively built up by considering only the rôles and identifiers occurring in the system A. We now partition $CONF_A$ w.r.t. $\approx$ and consider the equivalence class containing $(\mathcal{U};\mathcal{P}) \triangleright A$, called $CONF_A^{(\mathcal{U};\mathcal{P})}$. By fixing a metrics over schemas, the minimal schema to run the system A will be a minimal element of $CONF_A^{(\mathcal{U};\mathcal{P})}$ (if any).

Clearly, the existence of such a minimal element and the way in which it is chosen depend on a chosen *metrics*. For example, one can consider as a good metrics the value $|\mathcal{U}| + |\mathcal{P}|$, i.e., the size of the schema expressed in terms of the number of couples forming the relations $\mathcal{U}$ and $\mathcal{P}$. In this case, a minimal schema always exists. Other metrics could be based on the number of rôles used to define the schema, on the weight of the permissions associated to some designed users (once assumed a weight function to discriminate powerful permissions from common ones), on the average number of permissions associated to each rôle, and so on.

Notice that in general A may *not* be well-typed under the minimal schema. This is because the *static* typing procedure over-approximates the behaviour of a system (e.g., it also considers unreachable code). The bisimulation-based approach presented here is more accurate since it only considers the effective behaviour of the system. Thus, the schema obtained in this way describes the minimal requirements a schema should satisfy to *run* (and not to type) system A, while maintaining the behaviour of A under schema $(\mathcal{U};\mathcal{P})$.

Refining Systems to make them Executable. Usually, the task of properly putting **role**/**yield** operations within a system is tedious and error-prone; moreover, it assumes a full knowledge of the RBAC schema at programming time. We now describe a way to add rôle activations/deactivations within a system in such a way that the resulting system can be executed under a given schema (whenever possible). Notice that, given a RBAC schema $(\mathcal{U};\mathcal{P})$ and a system A without **role**/**yield** actions, we can simply refine A in a system A' by activating at the beginning of each session of a (generic) user r all the rôles in $\mathcal{U}(r)$. Intuitively, A' contains all the legal behaviours of A w.r.t. the RBAC schema given. However, the fact that all the rôles assigned to a user are always activated violates a basilar RBAC design principle: a rôle should be active only when needed.

We now give a procedure to refine a system to be well-typed for the given RBAC schema. The obtained system will be closer to the RBAC design principles. To this aim, we let $\vec{R}$ to denote a (possibly empty) ordered sequence of rôles and **role** $\vec{R}$ (resp., **yield** $\vec{R}$) to denote **role** $R_1.\cdots.$**role** R_n (resp., **yield** $R_1.\cdots.$**yield** R_n) whenever $\vec{R} = R_1,\cdots,R_n$. The refining procedure replaces any input/output prefix α occurring in session $r\{\!|\cdots|\!\}_\rho$ with the sequence of prefixes **role** $\vec{R}.\alpha.$**yield** $\vec{R}$ where $\vec{R}$ is formed

by rôles assigned to r, activable when having activated ρ and enabling the execution of α.[2] Moreover, since in principle there are several such $\vec{R}$, we chose one of the shortest, i.e. a sequence containing the minimum number of elements. Thus, we define function $en_a(\Gamma, \mathcal{P}, r, \rho)$ to return one of the shortest sequences of rôles (say, $\vec{R}$) such that $\Gamma; \rho \vdash_r^{\mathcal{P}} \mathbf{role}\, \vec{R}.a(x).\mathbf{nil}$. The function is undefined if such a $\vec{R}$ does not exists. Function $en_{\overline{u}}(\Gamma, \mathcal{P}, r, \rho)$ is defined similarly but for outputs over u. The refining procedure adapts the type system presented in Section 3; as an example, the following rule adapts (T-I).

$$\frac{\Gamma \vdash a^r : R(T) \qquad en_a(\Gamma, \mathcal{P}, \rho, r) = \vec{S} \qquad \Gamma, x : T; \rho \vdash_r^{\mathcal{P}} P \gg P'}{\Gamma; \rho \vdash_r^{\mathcal{P}} a(x).P \gg \mathbf{role}\, \vec{S}.a(x).\mathbf{yield}\, \vec{S}.P'}$$

The typing rules of Table 3 are adapted accordingly. Of course, when dealing with output prefixes, we use function $en_{\overline{u}}$ instead of en_a.

Functions $en\_(\ldots)$ can be easily calculated by reducing the problem to a breath first search in a direct acyclic graph. The nodes of the graph are labelled by permissions. The procedure to calculate $en_a(\Gamma, \mathcal{P}, \rho, r)$ starts from a node labelled with $R^?$, where $\Gamma \vdash a^r : R(T)$. Then, for all the rôles S assigned to r enabling inputs from a channel of rôle R (as established by Γ and $\mathcal{P}$), the procedure connects $R^?$ to a new node labelled by $S^\uparrow$. If $S \notin \rho$ the procedure will be iterated starting from this new node; otherwise, the procedure terminates and returns the sequence $R_n, \cdots, R_1$ such that $R^? \to R_1^\uparrow \to \ldots \to R_n^\uparrow \to S^\uparrow$ is one of the shortest path from $R^?$ to $S^\uparrow$. Intuitively, since $S \in \rho$, R_n can be activated in $r\{\!|\cdots|\!\}_\rho$; similarly, R_i can be activated in $r\{\!|\cdots|\!\}_{\rho \cup \{R_n, \cdots, R_{i+1}\}}$ for all $i = 1, \ldots, n-1$; finally, $a(x)$ can be executed in $r\{\!|\cdots|\!\}_{\rho \cup \{R_n, \cdots, R_1\}}$. Notice that n can be 0 (thus, the returned sequence can be empty); in this case, inputs from a^r are enabled by rôles in ρ. Finally, some care must be spent to avoid cycles generated, e.g., when a rôle R_1 can be activated only when R_2 is on, and R_2 can be activated only when R_1 is on, and neither R_1 nor R_2 are in ρ. These situations can be ruled out by ignoring all the paths containing multiple occurrences of the same label.

The soundness of the modified judgment $\Gamma \vdash^{\mathcal{P}} A \gg A'$ now follows.

Proposition 5.1. *Let $(\mathcal{U}; \mathcal{P})$ be a RBAC schema and Γ a typing environment respecting $\mathcal{U}$. Then, $\Gamma \vdash^{\mathcal{P}} A$ implies that $\Gamma \vdash^{\mathcal{P}} A \gg A$, while $\Gamma \vdash^{\mathcal{P}} A \gg A'$ implies that $\Gamma \vdash^{\mathcal{P}} A'$.*

Proof. To prove the first claim we proceed by induction over the length of the inference for judgment $\Gamma \vdash^{\mathcal{P}} A$ and we build a corresponding inference for judgment $\Gamma \vdash^{\mathcal{P}} A \gg A$. To this aim, notice that, if $\Gamma; \rho \vdash_r^{\mathcal{P}} a(x).P$ then $en_a(\Gamma, \mathcal{P}, \rho, r) = \epsilon$ (where ϵ denotes an empty sequence); this fact holds by soundness of function en_a. Thus $\Gamma; \rho \vdash_r^{\mathcal{P}} a(x).P \gg a(x).P$ since, by induction, $\Gamma, x : T; \rho \vdash_r^{\mathcal{P}} P \gg P$. Similar arguments hold when the prefix is an output or a **role**/**yield**, while the other cases follow by an easy induction.

The converse implication is dealt with symmetrically: from the inference for judgment $\Gamma \vdash^{\mathcal{P}} A \gg A'$ we build a (usually longer) inference for judgment $\Gamma \vdash^{\mathcal{P}} A'$. The proof is by induction over the length of $\Gamma \vdash^{\mathcal{P}} A \gg A'$. Let us consider just one case; the others are similar or easier. Let $\Gamma; \rho \vdash_r^{\mathcal{P}} a(x).P \gg \mathbf{role}\, \vec{S}.a(x).\mathbf{yield}\, \vec{S}.P'$. By induction, $\Gamma, x : T; \rho \vdash_r^{\mathcal{P}} P'$ and thus $\Gamma, x : T; \rho \cup \{\vec{S}\} \vdash_r^{\mathcal{P}} \mathbf{yield}\, \vec{S}.P'$ (indeed, it can be easily proved that $\rho \cap en_a(\Gamma, \mathcal{P}, \rho, r) = \emptyset$). By soundness of function en_a, it holds that $R^? \in \mathcal{P}(\rho \cup \{\vec{S}\})$; this implies that $\Gamma; \rho \cup \{\vec{S}\} \vdash_r^{\mathcal{P}} a(x).\mathbf{yield}\, \vec{S}.P'$. Again by soundness of function en_a, it holds that $\vec{S} \subseteq \mathcal{U}(r)$, $S_i^\uparrow \in \mathcal{P}(S_{i+1})$ for all $i = 1, \ldots, n-1$ and $S_n^\uparrow \in \mathcal{P}(\rho)$, where $\vec{S} = S_n, \cdots, S_1$. This suffices to conclude the desired $\Gamma; \rho \vdash_r^{\mathcal{P}} \mathbf{role}\, \vec{S}.a(x).\mathbf{yield}\, \vec{S}.P'$. □

[2] Notice that a lot of optimizations are implementable in order to reduce the number of actions **role**/**yield** introduced during the refining phase. For example, $\mathbf{role}\, R.a(x).\mathbf{yield}\, R.\mathbf{role}\, R.a(x).\mathbf{yield}\, R.P$ can be simplified in the equivalent (w.r.t. $\approx$) process $\mathbf{role}\, R.a(x).a(x).\mathbf{yield}\, R.P$. Similarly, $\mathbf{role}\, R.a(x).\mathbf{yield}\, R.\mathbf{role}\, S.b(y).\mathbf{yield}\, S.P$ can be simplified in $\mathbf{role}\, R.a(x).b(x).\mathbf{yield}\, R.P$, whenever rôle R enables inputs from both a and b. However, this optimizations require more complicated algorithms that we leave for future work.

To conclude, notice that there are other possible ways to find rôle sequences enabling inputs/outputs. For example, we can enforce the *least privilege* property while refining the system. A system satisfies this property if, whenever it performs an action, only the minimal set of permissions enabling the action is activated in the corresponding session. The approach presented above can be adapted to the new scenario: the main change affects how functions $en\_(\ldots)$ are calculated. Now, the metrics to minimize is $|\mathcal{P}(\vec{R})|$ instead of $|\vec{R}|$. Thus, the graph is now weighted: an edge $\_ \to R^{\uparrow}$ is associated with the number of permissions the activation of R adds to the current session permissions. The best $\vec{R}$ is extracted from the minimal path between the starting node and a reserved node OK. Intuitively, all edges $\_ \to OK$ are weighted with 0 and record that $\_ \in \mathcal{P}(\rho)$ (i.e. the permission $\_$ is enabled by the session permissions ρ passed to function $en\_$).

Relocating Activities. We now investigate another application of our theory, i.e. the possibility to transfer a process from a user to another one. This can be useful in order to minimize the number of users in a system. Moreover, assigning the activity of a user to another user can have a relevant economical impact: in a corporation, the administrator usually tries to minimize the number of employees by exploiting as much as possible each employee's activity.

We now give an axiomatic way to infer judgments of the form

$$(\mathcal{U};\mathcal{P}) \triangleright r\{\!|P|\!\}_\rho \;\approx\; (\mathcal{U};\mathcal{P}) \triangleright s\{\!|P|\!\}_\rho$$

This judgment says that the process P can be executed by r and s without affecting the overall system behaviour. Thus, the session $r\{\!|P|\!\}_\rho$ can be removed. If no other session of r is left in the system, then r is a useless user and is erased. The procedure $\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}$ equates systems under the schema $(\mathcal{U};\mathcal{P})$. The key rules defining it are given below and generalize the equations given at the end of Section 4.

$$\frac{r\{\!|P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|P|\!\}_\rho}{r\{\!|u\langle v\rangle.P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|u\langle v\rangle.P|\!\}_\rho} \qquad \frac{\mathcal{U}(a^r) = \{R\} \qquad \mathcal{U}(a^s) = \{S\} \qquad \{R^?, S^?\} \cap \mathcal{P}(\rho) = \emptyset}{r\{\!|a(x).P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|a(x).P|\!\}_\rho}$$

$$\frac{R \notin \mathcal{U}(r) \cup \mathcal{U}(s)}{r\{\!|\mathbf{role}\, R.P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|\mathbf{role}\, R.P|\!\}_\rho} \qquad \frac{R \in \mathcal{U}(r) \cap \mathcal{U}(s) \qquad r\{\!|P|\!\}_{\rho\cup\{R\}} \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|P|\!\}_{\rho\cup\{R\}}}{r\{\!|\mathbf{role}\, R.P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|\mathbf{role}\, R.P|\!\}_\rho}$$

$$\frac{R \notin \rho}{r\{\!|\mathbf{yield}\, R.P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|\mathbf{yield}\, R.P|\!\}_\rho} \qquad \frac{R \in \rho \qquad r\{\!|P|\!\}_{\rho-\{R\}} \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|P|\!\}_{\rho-\{R\}}}{r\{\!|\mathbf{yield}\, R.P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|\mathbf{yield}\, R.P|\!\}_\rho}$$

As stated by the following Proposition, the procedure given above is a sound axiomatization for the judgment $(\mathcal{U};\mathcal{P}) \triangleright r\{\!|P|\!\}_\rho \;\approx\; (\mathcal{U};\mathcal{P}) \triangleright s\{\!|P|\!\}_\rho$, whenever P is finite. A process is *finite* if it is built up without using the replication operator. As usual, equivalences over recursive processes are not recursively enumerable and, thus, they cannot be finitely axiomatizable.

Proposition 5.2. *Let P be a finite process. Then $r\{\!|P|\!\}_\rho \;\overset{\mathcal{U}}{\underset{\mathcal{P}}{=}}\; s\{\!|P|\!\}_\rho$ implies that $(\mathcal{U};\mathcal{P}) \triangleright r\{\!|P|\!\}_\rho \;\approx\; (\mathcal{U};\mathcal{P}) \triangleright s\{\!|P|\!\}_\rho$.*

6 Related Work and Conclusion

To the best of our knowledge, no other process-calculi based study have been conducted on RBAC. A number of papers have instead dealt with the formal specification and verification of RBAC schema. In [13, 20] formal methods are used only to verify the correctness of the schema definition but not of the whole system. In [20], the ALLOY language is used to detect possible conflicts in RBAC schemas supporting simultaneously delegation of authority and separation of duty. A constraint analyzer allows the schema validation to be computed automatically. In [13, 14], the authors use graph transformation which combines an intuitive visual description of the RBAC schema with solid semantical foundations. Ahn et al. in [1] introduce a formal language for the specification of more sophisticated role-based authorization constraints, such as prohibition and obligation constraints. These approaches are complementary to ours: they can be integrated with our technique in order to verify the consistency of $(\mathcal{U}; \mathcal{P})$, but they do not give any hint about the correct execution of a system as our method does.

In [3], Bertino et al. develop a logical framework for reasoning about access control models. The framework is general enough to model discretionary, mandatory, and role-based access control models. Such a framework is useful for comparing the expressive power of the models, but it cannot be used to verify the correct execution of a system under a given schema.

Probably, the most related work, although not aiming at studying RBAC systems, is [4], insofar as rôles can be understood as (privilege) groups. *Groups* are introduced in *loc. cit.* as types for channels, and used to limit their visibility. A type system ensures that channels belonging to a fresh group can be only used by processes within the initial scope of the group. Thus, processes can access channels according to their physical distribution (w.r.t. group restrictions). In our work this feature is modified so that not only the place where the process runs (i.e., the user running the process) but also its execution history (i.e., the user session where the process runs) is relevant to execute an action. E.g., outputs over a^r of group R can be executed only by processes whose user r is such that $R^! \in \mathcal{P}(\mathcal{U}(r))$; moreover, such action must be enabled by at least one of the rôles active in r's session which contains it. The set of such sessions changes according to the computation and, thus, the processes enabled to access a channel change dynamically. In this sense, this work can be seen as a calculus of *dynamic* groups.

References

[1] G.-J. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, 3(4):207–226, 2000.

[2] R. Amadio. On modelling mobility. *Theoretical Computer Science*, 240(1):147–176, 2000.

[3] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A Logical Framework for Reasoning about Access Control Models. In *Proc. of 6th SACMAT*, pages 41–52. ACM Press, 2001.

[4] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. In *Proc. of CONCUR'00*, volume 1877 of *LNCS*, pages 365–379. Springer, 2000.

[5] L. Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[6] D. Ferraiolo and D. Kuhn. Role-Based Access Control. In *Proc. of the NIST-NSA National Computer Security Conference*, pages 554–563, 1992.

[7] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[8] C. Fournet, G. Gonthier, J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 406–421. Springer, 1996.

[9] C. Fournet and C. Laneve. Bisimulations for the join-calculus. *Theoretical Computer Science*, 266(1-2):569–603, 2001.

[10] M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. In *Proceedings of CATS '02*, volume 61 of *Electronic Notes on Theoretical Computer Science*. Elsevier Science Inc., New York, 2002. Full version to appear in *Mathematical Structures in Computer Science*.

[11] M. Hennessy and J. Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation*, 173:82–120, 2002.

[12] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[13] M. Koch, L. Mancini, and F. Parisi-Presicce. A Formal Model for Role-Based Access Control Using Graph Transformation. In *Proc. of 5th ESORICS*, volume 1895 of *LNCS*, pages 122–139. Springer, 2000.

[14] M. Koch, L. Mancini, and F. Parisi-Presicce. Decidability of Safety in Graph-based Models for Access Control. In *Proc. of 7th ESORICS*, volume 2502 of *LNCS*, pages 229–243. Springer, 2002.

[15] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. In *Proceedings of ICALP '98*, volume 1443 of *LNCS*, pages 856–867. Springer, 1998.

[16] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[17] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[18] D. Sangiorgi. The name discipline of uniform receptiveness. *Theoretical Computer Science*, 221(1–2):457–493, 1999. An abstract appeared in the *Proc. of ICALP '97*, LNCS 1256, pages 303–313.

[19] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[20] A. Schaad and J. Moffett. A Lightweight Approach to Specification and Analysis of Role-based Access Control Extensions. In *Proc. of 7th SACMAT*, pages 13–22. ACM Press, 2002.

A Technical Proofs

In this section, we give details for the proofs of the main results of the paper.

A.1 Proofs of Section 4

To prove Theorem 4.2 and 4.3, we first need some preliminary results for the LTS and for relation $\approx$. We write $a^r \in \mathrm{F}(A, \mu, \widetilde{b^s}, dom(\mathcal{U}))$ to mean that $a^r \in \mathrm{F}(A) \cup \mathrm{F}(\mu) \cup \widetilde{b^s} \cup dom(\mathcal{U})$; derived notations have a similar meaning. A channel a^r is said to be *fresh* for $\_$ if it does not occur in $\_$, i.e. if $a^r \notin \mathrm{F}(\_)$; we shall not specify $\_$ whenever it refers all the entities (i.e. systems, schemas, sets of channels, ...) involved, and simply say that a^r is fresh. Moreover, $\_\, b^s : S$ denotes either $a^r b^s : S$ or $\overline{a}^r b^s : S$; similarly, $a^r \_$ denotes either $a^r n$ or $a^r n : S$.

Proposition A.1. *$\approx$ is an equivalence relations. Moreover, $A \equiv B$ implies that $(\mathcal{U}; \mathcal{P}) \triangleright A \approx (\mathcal{U}; \mathcal{P}) \triangleright B$, for all RBAC schemas.*

Proof. The first part is obvious from Definition 4.1. The second part is proved by an easy induction over the length of the inference for $A \equiv B$. □

Lemma A.2. *The following facts hold:*

1. *if* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{a^r n} (\mathcal{U};\mathcal{P}) \triangleright A'$ *then* $A \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel r\{\!|a^r(x).P|\!\}_\rho)$ *for* $a^r, n \notin \widetilde{a^r}$, $n \in dom(\mathcal{U})$ *and* $A' \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel r\{\!|P[^n/_x]|\!\}_\rho)$. *The converse holds true whenever* $R^? \in \mathcal{P}(\rho)$, *for* $\mathcal{U}(a^r) = \{R\}$

2. *if* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{a^r n:R} (\mathcal{U} \uplus \{n:R\};\mathcal{P}) \triangleright A'$ *then* $A \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel r\{\!|a^r(x).P|\!\}_\rho)$ *for* $a^r \notin \widetilde{a^r}$, $n \notin \widetilde{a^r} \cup dom(\mathcal{U})$ *and* $A' \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel r\{\!|P[^n/_x]|\!\}_\rho)$. *The converse holds true whenever* $R^? \in \mathcal{P}(\rho)$, *for* $\mathcal{U}(a^r) = \{R\}$

3. *if* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\overline{a^r} n} (\mathcal{U};\mathcal{P}) \triangleright A'$ *then* $A \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel s\{\!|a^r\langle n\rangle.P|\!\}_\rho)$ *for* $a^r, n \notin \widetilde{a^r}$ *and* $A' \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel s\{\!|P|\!\}_\rho)$. *The converse holds true whenever* $R^! \in \mathcal{P}(\rho)$, *for* $\mathcal{U}(a^r) = \{R\}$

4. *if* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\overline{a^r} n:R} (\mathcal{U} \uplus \{n:R\};\mathcal{P}) \triangleright A'$ *then* $n = b^s$, $A \equiv (\nu b^s{:}R)(\nu\, \widetilde{a^r{:}R})(B \parallel t\{\!|a^r\langle b^s\rangle.P|\!\}_\rho)$ *for* $a^r \notin \widetilde{a^r} \cup \{b^s\}$ *and* $A' \equiv (\nu\, \widetilde{a^r{:}R})(B \parallel t\{\!|P|\!\}_\rho)$. *The converse holds true whenever* $R^! \in \mathcal{P}(\rho)$, *for* $\mathcal{U}(a^r) = \{R\}$

Proof. The four points are all proved in the same way. The first implication is done by induction over the length of the inference for $\xrightarrow{-}$; the second implication is based on the definition of the LTS given in Table 4. □

Lemma A.3. *The following facts hold:*

1. *if* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}';\mathcal{P}) \triangleright A'$ *and* a^r *fresh, then* $(\mathcal{U}[^{a^r}/_{b^r}];\mathcal{P}) \triangleright A[^{a^r}/_{b^r}] \xrightarrow{\mu[^{a^r}/_{b^r}]} (\mathcal{U}'[^{a^r}/_{b^r}];\mathcal{P}) \triangleright A'[^{a^r}/_{b^r}]$

2. *Let* $(\mathcal{U} \uplus \{a^r:R\};\mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}' \uplus \{a^r:R\};\mathcal{P}) \triangleright A'$ *and* $a^r \notin \mathrm{F}(A)$. *If* $a^r \in \mathrm{B}(\mu)$ *then* $\mu = b^s a^r$ *and* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{b^s a^r:R} (\mathcal{U} \uplus \{a^r:R\};\mathcal{P}) \triangleright A'$; *if* $a^r \notin \mathrm{B}(\mu)$ *then* $(\mathcal{U};\mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}';\mathcal{P}) \triangleright A'$

3. *if* $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright B$ *and* a^r *is fresh, then* $(\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A \approx (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B$

Proof. The first two claims are proved by an easy induction over the length of the inference for $\xrightarrow{\mu}$. The last one is proved by showing that the relation

$$\mathfrak{R} \triangleq \quad \approx \quad \cup \quad \{((\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A\,,\ (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B)\ : \\ (\mathcal{U}_2;\mathcal{P}_2) \triangleright A \approx (\mathcal{U}_2;\mathcal{P}_2) \triangleright B \ \wedge\ a^r \notin \mathrm{F}(A,B)\}$$

is a bisimulation. Now, let $(\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A \xrightarrow{\mu} (\mathcal{U}_1' \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A'$. Because of case (2) of this Proposition, if a^r occurs in μ then μ must be $b^s a^r$ for some b^s and $(\mathcal{U}_1;\mathcal{P}_1) \triangleright A \xrightarrow{b^s a^r:R} (\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A'$. Thus, $(\mathcal{U}_2;\mathcal{P}_2) \triangleright B \xRightarrow{b^s a^r:R} (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B'$ and $(\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A' \approx (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B'$. This implies that $(\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B \xRightarrow{b^s a^r} (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B'$ and $((\mathcal{U}_1 \uplus \{a^r:R\};\mathcal{P}_1) \triangleright A'\,,\ (\mathcal{U}_2 \uplus \{a^r:R\};\mathcal{P}_2) \triangleright B') \in \mathfrak{R}$. The case when a^r does not occur in μ is simpler; just notice that, upon transitions, a^r still remains fresh. □

We can now consider the two main theorems. We start with Theorem 4.2.

Theorem 4.2 (Congruence Properties of $\approx$)

Proof. Both the clauses of the theorem are proved once shown that relation

$$\begin{aligned}\mathfrak{R} \triangleq \{\, ((\mathcal{U}_1;\mathcal{P}_1) \triangleright (\nu\, \widetilde{a^r{:}R})(A_1 \parallel B)\,,\ (\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A_2 \parallel B)) \ :\\ (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_1 \approx (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_2 \ \wedge\\ (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \approx (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \,\}\end{aligned}$$

is a bisimulation. Let $(\mathcal{U}_1;\mathcal{P}_1) \triangleright (\nu\, \widetilde{a^r{:}R})(A_1 \parallel B) \xrightarrow{\mu} (\mathcal{U}'_1;\mathcal{P}_1) \triangleright \bar{A}_1$; by definition of the LTS, it can only be one of the following cases.

1. $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_1 \xrightarrow{\mu} (\mathcal{U}'_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1$ for $\mathrm{B}\ (\mu) \cap \mathrm{F}\ (B) = \emptyset$. Thus, $\bar{A}_1 \triangleq (\nu\, \widetilde{a^r{:}R})(A'_1 \parallel B)$. By hypothesis, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A_2 \stackrel{\hat{\mu}}{\Longrightarrow} (\mathcal{U}'_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$ and $(\mathcal{U}'_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1 \approx (\mathcal{U}'_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$. This implies that $(\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A_2 \parallel B) \stackrel{\hat{\mu}}{\Longrightarrow} (\mathcal{U}'_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_2 \parallel B)$ and $((\mathcal{U}'_1;\mathcal{P}_1) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_1 \parallel B)\,,\ (\mathcal{U}'_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_2 \parallel B)) \in \mathfrak{R}$. Indeed, if $\mathrm{B}\ (\mu) = \emptyset$ then $\mathcal{U}'_i = \mathcal{U}_i$ and the claim is trivial. Otherwise, $\mu = \_\, b^s{:}S$ for some S and $\mathcal{U}'_i = \mathcal{U}_i \uplus \{b^s{:}S\}$. By hypothesis, b^s is fresh for B and, thus, by Proposition A.3(3) it holds that $((\mathcal{U}'_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \approx (\mathcal{U}'_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright B$.

2. $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \xrightarrow{\mu} (\mathcal{U}'_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B'$ for $\mathrm{B}\ (\mu) \cap \mathrm{F}\ (A_1) = \emptyset$. By Proposition A.3(1), the transition can be inferred also upon renaming of the (possible) bound channel in μ into a channel fresh also for A_2. The proof then proceeds like in the previous case.

3. $\mu = \overline{a}^r b^s : S$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_1 \xrightarrow{\overline{a}^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1$ for $b^s : S \in \widetilde{a^r{:}R}$ and $b^s \notin \mathrm{F}\ (B)$. By letting $\widetilde{c^t{:}R'}$ be the set $\widetilde{a^r{:}R} - \{b^s : S\}$, we have that $\bar{A}_1 \triangleq (\nu\, \widetilde{c^t{:}R'})(A'_1 \parallel B)$ and $\mathcal{U}'_1 = \mathcal{U} \uplus \{b^s{:}S\}$. By hypothesis, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A_2 \stackrel{\overline{a}^r b^s}{\Longrightarrow} (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1 \approx (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$. This implies that $(\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A_2 \parallel B) \stackrel{\overline{a}^r b^s{:}S}{\Longrightarrow} (\mathcal{U}'_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{c^t{:}R'})(A'_2 \parallel B)$, for $\mathcal{U}'_2 = \mathcal{U}_2 \uplus \{b^s{:}S\}$, and $((\mathcal{U}'_1;\mathcal{P}_1) \triangleright (\nu\, \widetilde{c^t{:}R'})(A'_1 \parallel B)\,,\ (\mathcal{U}'_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{c^t{:}R'})(A'_2 \parallel B) \in \mathfrak{R}$.

4. The case when $\mu = \overline{a}^r b^s : S$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \xrightarrow{\overline{a}^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B'$ for $b^s{:}S \in \widetilde{a^r{:}R}$ is similar to the previous one.

5. $\mu = \tau$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_1 \xrightarrow{a^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \xrightarrow{\overline{a}^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B'$. Thus, $\bar{A}_1 \triangleq (\nu\, \widetilde{a^r{:}R})(A'_1 \parallel B')$. By hypothesis, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A_2 \stackrel{a^r b^s}{\Longrightarrow} (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1 \approx (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright A'_2$. Similarly, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright B \stackrel{\overline{a}^r b^s}{\Longrightarrow} (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright B'$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B' \approx (\mathcal{U}_2 \uplus \widetilde{a^r{:}R};\mathcal{P}_2) \triangleright B'$. Hence, $(\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A_2 \parallel B) \Rightarrow (\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_2 \parallel B')$ and $((\mathcal{U}_1;\mathcal{P}_1) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_1 \parallel B')\,,\ (\mathcal{U}_2;\mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A'_2 \parallel B')) \in \mathfrak{R}$.

6. The case for $\mu = \tau$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A_1 \xrightarrow{\overline{a}^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright A'_1$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B \xrightarrow{a^r b^s} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R};\mathcal{P}_1) \triangleright B'$ is similar to the previous one.

7. $\mu = \tau$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R}; \mathcal{P}_1) \triangleright A_1 \xrightarrow{a^r b^s:S} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_1) \triangleright A_1'$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R}; \mathcal{P}_1) \triangleright B \xrightarrow{\overline{a}^r b^s:S} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_1) \triangleright B'$. Thus, $\overline{A}_1 \triangleq (\nu\, \widetilde{a^r{:}R}, b^s : S)(A_1' \parallel B')$. By hypothesis, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R}; \mathcal{P}_2) \triangleright A_2 \xRightarrow{a^r b^s:S} (\mathcal{U}_2 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_2) \triangleright A_2'$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_1) \triangleright A_1' \approx (\mathcal{U}_2 \uplus \widetilde{a^r{:}R} \uplus \{b^s : S\}; \mathcal{P}_2) \triangleright A_2'$. Similarly, $(\mathcal{U}_2 \uplus \widetilde{a^r{:}R}; \mathcal{P}_2) \triangleright B \xRightarrow{\overline{a}^r b^s:S} (\mathcal{U}_2 \uplus \widetilde{a^r{:}R} \uplus \{b^s : S\}; \mathcal{P}_2) \triangleright B'$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_1) \triangleright B' \approx (\mathcal{U}_2 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_2) \triangleright B'$. Hence, $(\mathcal{U}_2; \mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R})(A_2 \parallel B) \Rightarrow (\mathcal{U}_2; \mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R}, b^s{:}S)(A_2' \parallel B')$ and $((\mathcal{U}_1; \mathcal{P}_1) \triangleright (\nu\, \widetilde{a^r{:}R}, b^s{:}S)(A_1' \parallel B')\ ,\ (\mathcal{U}_2; \mathcal{P}_2) \triangleright (\nu\, \widetilde{a^r{:}R}, b^s{:}S)(A_2' \parallel B')) \in \mathfrak{R}$.

8. The case for $\mu = \tau$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R}; \mathcal{P}_1) \triangleright A_1 \xrightarrow{\overline{a}^r b^s:S} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s : S\}; \mathcal{P}_1) \triangleright A_1'$ and $(\mathcal{U}_1 \uplus \widetilde{a^r{:}R}; \mathcal{P}_1) \triangleright B \xrightarrow{a^r b^s:S} (\mathcal{U}_1 \uplus \widetilde{a^r{:}R} \uplus \{b^s{:}S\}; \mathcal{P}_1) \triangleright B'$ is similar to the previous one. □

We now turn to Theorem 4.3. For notational convenience, we write $(\mathcal{U}; \mathcal{P}) \triangleright A \approx (\mathcal{U}; \mathcal{P}) \triangleright B$ as $(\mathcal{U}; \mathcal{P}) \triangleright A \approx B$. The proof relies on the following lemma.

Lemma A.4 (Weakening for $\approx$). *If $(\mathcal{U}; \mathcal{P}) \triangleright A \approx B$ and A and B are well-typed for $(\mathcal{U}; \mathcal{P})$, then $(\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright A \approx B$ for all $\mathcal{U}'$ and $\mathcal{P}'$.*

Proof. We have to prove that the relation

$$\mathfrak{R} \triangleq \{((\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright A\ ,\ (\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B)\ :\ (\mathcal{U}; \mathcal{P}) \triangleright A \approx B\}$$

is a bisimulation. Let $(\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright A \xrightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}' \uplus \mathcal{U}''; \mathcal{P} \cup \mathcal{P}') \triangleright A'$. We now distinguish on μ.

1. $\mu = a^r n$. In this case, $\mathcal{U}'' = \emptyset$ and $n \in dom(\mathcal{U} \uplus \mathcal{U}')$. We now distinguish whether $n \in dom(\mathcal{U})$ or $n \in dom(\mathcal{U}')$.

   (a) Let $n \in dom(\mathcal{U})$; thus, because of well-typedness, $(\mathcal{U}; \mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U}; \mathcal{P}) \triangleright A'$. By hypothesis, $(\mathcal{U}; \mathcal{P}) \triangleright B \xRightarrow{\mu} (\mathcal{U}; \mathcal{P}) \triangleright B'$ and $(\mathcal{U}; \mathcal{P}) \triangleright A' \approx B'$. Then, $(\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B \xRightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B'$ and $((\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright A'\ ,\ (\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B') \in \mathfrak{R}$, as required.

   (b) Let $n \in dom(\mathcal{U}')$; thus, $(\mathcal{U}; \mathcal{P}) \triangleright A \xrightarrow{a^r n:R} (\mathcal{U} \uplus \{n : R\}; \mathcal{P}) \triangleright A'$ for any R; in particular, we can choose $R \in \mathcal{U}'(n)$ and let $\overline{\mathcal{U}} = \mathcal{U}' - \{(n, R)\}$. By hypothesis, $(\mathcal{U}; \mathcal{P}) \triangleright B \xRightarrow{a^r n:R} (\mathcal{U} \uplus \{n : R\}; \mathcal{P}) \triangleright B'$ and $(\mathcal{U} \uplus \{n : R\}; \mathcal{P}) \triangleright A' \approx B'$. Then, $(\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B \xRightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B'$ and $((\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright A'\ ,\ (\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B') \in \mathfrak{R}$, because $\mathcal{U} \uplus \mathcal{U}' = \mathcal{U} \uplus \{n : R\} \uplus \overline{\mathcal{U}}$.

2. $\mu = a^r n : R$. In this case, $\mathcal{U}'' = \{n : R\}$ since $n \notin dom(\mathcal{U} \uplus \mathcal{U}')$. This implies that $(\mathcal{U}; \mathcal{P}) \triangleright A \xrightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}''; \mathcal{P}) \triangleright A'$. By hypothesis, $(\mathcal{U}; \mathcal{P}) \triangleright B \xRightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}''; \mathcal{P}) \triangleright B'$ and $(\mathcal{U} \uplus \mathcal{U}''; \mathcal{P}) \triangleright A' \approx B'$. Then, $(\mathcal{U} \uplus \mathcal{U}'; \mathcal{P} \cup \mathcal{P}') \triangleright B \xRightarrow{\mu} (\mathcal{U} \uplus \mathcal{U}' \uplus \mathcal{U}''; \mathcal{P} \cup \mathcal{P}') \triangleright B'$ and $((\mathcal{U} \uplus \mathcal{U}' \uplus \mathcal{U}''; \mathcal{P} \cup \mathcal{P}') \triangleright A'\ ,\ (\mathcal{U} \uplus \mathcal{U}' \uplus \mathcal{U}''; \mathcal{P} \cup \mathcal{P}') \triangleright B') \in \mathfrak{R}$.

3. $\mu = \overline{a}^r n$. This case is dealt with similarly to case 1.

4. $\mu = \overline{a}^r n : R$. This case is dealt with similarly to case 2.

5. $\mu = \tau$. This case is dealt with similarly to case 1(a). □

Theorem 4.3 (Soundness of $\approx$)

Proof. It sufficies to prove that the relation

$$\mathfrak{R} \triangleq \{\, \Gamma \models^{\mathcal{P}} (A, B) \;:\; \Gamma \vdash^{\mathcal{P}} A \;\wedge\; \Gamma \vdash^{\mathcal{P}} B \;\wedge\; (\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A \approx B \}$$

is barb preserving, reduction closed and contextual.

1. Let $A \downarrow a^r$. By Definition 4.2 and well-typedness, it holds that $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A \xrightarrow{a^r\_}$ by exploiting Proposition A.2(1)/(2)). Then, $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright B \Rightarrow (\mathcal{U}_\Gamma; \mathcal{P}) \triangleright B' \xrightarrow{a^r\_}$. Again by Proposition A.2(1)/(2) and Definition 4.2, $B \Downarrow a^r$. The case for $A \downarrow \overline{a}^r$ is similar, but uses Proposition A.2(3)/(4).

2. Let $A \longmapsto A'$. By Proposition 4.1 and well-typedness, this implies that $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A \xrightarrow{\tau} (\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A''$ for some $A'' \equiv A'$. Thus, $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright B \Rightarrow (\mathcal{U}_\Gamma; \mathcal{P}) \triangleright B'$ and $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A'' \approx B'$. Again by Proposition 4.1, $B \Longmapsto B'$ and $\Gamma \models^{\mathcal{P}} A' \;\mathfrak{R}\; B'$. Indeed, by Theorem 3.1, it holds that $\Gamma \vdash^{\mathcal{P}} A'$ and $\Gamma \vdash^{\mathcal{P}} B'$. Moreover, by using Proposition A.1, we have that $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A' \approx A''$ and, thus, $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A' \approx B'$.

3. We pick up a pair $\Gamma \models^{\mathcal{P}} A \;\mathfrak{R}\; B$ and analize the three clauses defining the contextuality property.

   (a) Let $\mathcal{P}'$ be a permissions-to-rôles assignement and $\widetilde{u : T}$ be such that $\Gamma, \widetilde{u : T}$ is defined. By Lemma **??** we know that $\Gamma, \widetilde{u : T} \vdash^{\mathcal{P}\cup\mathcal{P}'} A$ and $\Gamma, \widetilde{u : T} \vdash^{\mathcal{P}\cup\mathcal{P}'} B$. Moreover, we let $\mathcal{U}_{\widetilde{u:T}}$ to be the rôles-to-users assignement such that $\mathcal{U}_{\widetilde{u:T}}(r) = \rho$ whenever $r : \rho[\widetilde{a : C}] \in \widetilde{u : T}$ and $\mathcal{U}_{\widetilde{u:T}}(a^r) = \{R\}$ whenever $a^r : R(T) \in \widetilde{u : T}$. It is easy to check that $\mathcal{U}_{\Gamma,\widetilde{u:T}} = \mathcal{U}_\Gamma \uplus \mathcal{U}_{\widetilde{u:T}}$. Thus, by Lemma A.4, $(\mathcal{U}_{\Gamma,\widetilde{u:T}}; \mathcal{P} \cup \mathcal{P}') \triangleright A \approx B$. This sufficies to conclude that $\Gamma, \widetilde{u : T} \models^{\mathcal{P}\cup\mathcal{P}'} A \;\mathfrak{R}\; B$.

   (b) Let $\overline{A}$ be a system such that $\Gamma \vdash^{\mathcal{P}} \overline{A}$. By Theorem 4.2(1) and by reflexivity of $\approx$, we can state that $(\mathcal{U}_\Gamma; \mathcal{P}) \triangleright A \parallel \overline{A} \approx B \parallel \overline{A}$. Moreover, by rule (T-S P), it holds that $\Gamma \vdash^{\mathcal{P}} A \parallel \overline{A}$ and $\Gamma \vdash^{\mathcal{P}} B \parallel \overline{A}$. Thus, $\Gamma \models^{\mathcal{P}} A \parallel \overline{A} \;\mathfrak{R}\; B \parallel \overline{A}$, as required.

   (c) Let $\Gamma = \Gamma', a^r{:}R(T)$. It is easy to check that $\mathcal{U}_\Gamma = \mathcal{U}_{\Gamma'} \uplus a^r{:}R$ and, thus, $(\mathcal{U}_{\Gamma'} \uplus a^r{:}R; \mathcal{P}) \triangleright A \approx B$. By Theorem 4.2(2), this implies that $(\mathcal{U}_{\Gamma'}; \mathcal{P}) \triangleright (\nu a^r : R)A \approx (\nu a^r : R)B$; moreover, by rule (T-S R), $\Gamma' \vdash^{\mathcal{P}} (\nu a^r : R)A$ and $\Gamma' \vdash^{\mathcal{P}} (\nu a^r : R)B$. Thus, $\Gamma' \models^{\mathcal{P}} (\nu a^r : R)A \;\mathfrak{R}\; (\nu a^r : R)B$, as required. □