Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 117 (2005) 183–207
www.elsevier.com/locate/entcs

Representing the MSR Cryptoprotocol Specification Language in an Extension of Rewriting Logic with Dependent Types

Iliano Cervesato[1]

ITT Industries, Inc., Advanced Engineering and Sciences Division
Alexandria, VA 22303, USA

Mark-Oliver Stehr[2]

University of Illinois at Urbana-Champaign,
Computer Science Department, Urbana, IL 61801, USA

Abstract

This paper presents a shallow and hence efficient embedding of the security protocol specification language MSR into rewriting logic with dependent types, an instance of the open calculus of constructions which integrates key concepts from equational logic, rewriting logic, and type theory. MSR is based on a form of first-order multiset rewriting extended with existential name generation and a flexible type infrastructure centered on dependent types with subsorting. This encoding is intended to serve as the basis for implementing an MSR specification and analysis environment using existing first-order rewriting engines such as Maude.

Keywords: MSR, protocol specification, open calculus of constructions, dependent types, Maude.

1 Introduction

MSR originated as a simple logic-oriented language aimed at investigating the decidability of protocol analysis under a variety of assumptions [8]. It evolved into a precise, powerful, flexible, and still relatively simple framework

[1] Email: iliano@itd.nrl.navy.mil
[2] Email: stehr@uiuc.edu

doi:10.1016/j.entcs.2004.06.023

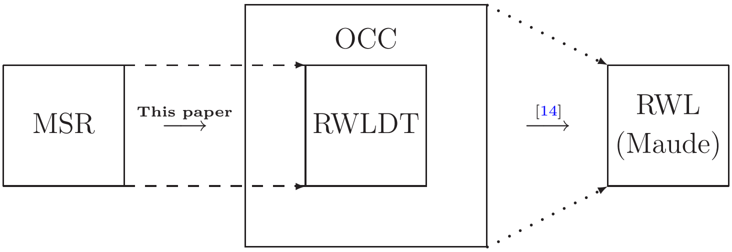

Figure 1. Architecture of the Embedding of MSR into Rewriting Logic

for the specification of complex cryptographic protocols, possibly structured as a collection of coordinated subprotocols [3,5]. It uses strongly-typed multiset rewriting rules over first-order atomic formulas to express protocol actions and relies on a form of existential quantification to symbolically model the generation of nonces and other fresh data. Dependent types are a useful abstraction mechanism not available in other languages. For instance, the dependency of public/private keys on their owner can be naturally expressed at the type level. Finally, MSR supports an array of useful static checks that include type-checking [4] and data access verification [6].

This work outlines an encoding of the core of MSR into rewriting logic (RWL), to be more precise into its extension with dependent types (RWLDT). Rewriting logic [11] draws on the observation that many paradigms can naturally be captured by conditional rewriting modulo an underlying equational theory, the theory of multisets being a particularly important case for the specification of concurrent systems and protocols. Recently a combination of equational logic and rewriting logic with dependent types has been studied in [14] under the name *open calculus of constructions* (OCC). In this paper we show that a restricted predicative instance of OCC, that we call rewriting logic with dependent types (RWLDT), can be used to represent typed MSR specifications in a way which preserves all type information. RWLDT does not natively support the expression of existential name generation: our encoding implements it with counters. Moreover, ensuring the executability of the resulting code required some care.

Composing the mapping from MSR into RWLDT with a mapping from RWLDT into RWL, which has already been implemented as part of the OCC prototype [14], we obtain an encoding from MSR into RWL, which can serve as the basis of an execution environment for MSR in a RWL-based language such as Maude [9]. This two-level approach, which is summarized in Figure 1, has some advantages over a direct mapping into RWL. The first is modularity and separation of concerns: the mapping from MSR into RWLDT is only concerned with the dynamics (given by the rules) but preserves the static part (given by declarations, types, and terms). The second advantage is that RWLDT seems to be the right level for user interaction, because terms and types closely correspond to those of MSR. Finally, the preservation of types and the fact that RWLDT is a sub-logic of OCC provides suitable level of abstraction for formal reasoning, a possibility that is not the subject of this paper, but that we hope to explore in the future.

This work serves as the basis for a forthcoming prototype of MSR, which will eventually run on top of Maude [9]. The linguistic affinity between MSR and RWLDT allow for a much simpler construction than a direct implementation. Mapping MSR into the popular CAPSL Intermediate Language [10] would have been more difficult, because MSR has a much richer typing infrastructure than CIL.

The remainder of this paper is organized as follows: We introduce MSR and RWLDT in Sections 3 and 4, respectively. In Section 5 we define the mapping from MSR into RWLDT, state its key properties, and outline some simple optimizations. It is applied to our running example, the Otway-Rees protocol, in Section 6. We conclude this paper with a discussion of limitations, implementation aspects, and possible extensions of our approach. First, some notation.

2 Notation

We use [] to denote the empty list and a comma to denote list concatenation. We write identifiers ranging over lists in bold, and indicate their length with a superscript. Therefore, $\boldsymbol{X}^n$ denotes a list of n elements. We will generally omit the length information when irrelevant or easily deducible from the context. Occasionally, we write $|\boldsymbol{X}|$ for the length of a list $\boldsymbol{X}$. We write $\boldsymbol{X}_i$ (or $\boldsymbol{X}^n_i$) for the i-th element of $\boldsymbol{X}$. We abbreviate constructions over all elements in a list as constructions over the list itself: for example, we may write $(M\ \boldsymbol{N}^n)$ for $(M\ \boldsymbol{N}_1 \dots \boldsymbol{N}_n)$, and $\forall \boldsymbol{X}^n : \boldsymbol{U}^n$ for $\forall \boldsymbol{X}_1 : \boldsymbol{U}_1 \dots \forall \boldsymbol{X}_n : \boldsymbol{U}_n$.

3 The MSR Cryptoprotocol Specification Language

The syntax of instances of MSR tailored to specific security protocols has been presented in [4,5]. Here, we will instead concentrate on a more abstract syntax, currently undergoing formalization in [7], which allows the user to declare operators such as message concatenation and encryption rather than having them hard-coded in the language. The core of the syntax of MSR is given in the following table:

$$\begin{array}{lll}
\textit{Terms} & M, N & ::= \; X \;\mid\; M\ N \\
\textit{Types} & T & ::= \; M \;\mid\; \texttt{state} \;\mid\; \texttt{princ} \;\mid\; \texttt{msg} \;\mid\; \{X : T\}T' \\
\textit{Kinds} & K & ::= \; \texttt{type} \;\mid\; \{X : T\}K \\
\textit{Contexts} & \mathcal{D} & ::= \; \cdot \;\mid\; \mathcal{D}, X : K \;\mid\; \mathcal{D}, X : T \\
\textit{States} & S & ::= \; \cdot \;\mid\; S, M \\
\textit{Rules} & \rho & ::= \; \texttt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} \,.\, (S \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} \,.\, S') \\
\textit{Roles} & \mathcal{P} & ::= \; \texttt{ROLE}\ i : \forall P : \texttt{princ} \,.\, \exists \boldsymbol{L} : \boldsymbol{T} \,.\, \boldsymbol{\rho} \\
 & & \quad\;\; \mid\; \texttt{ROLE}\ i : \ \texttt{FOR}\ P : \texttt{princ} \,.\, \exists \boldsymbol{L} : \boldsymbol{T} \,.\, \boldsymbol{\rho}
\end{array}$$

MSR is based on first-order terms, that for simplicity we limit to identifiers (X) and application. Here, X can be either a bound variable or a previously declared identifier. For conciseness, we describe atomic types (i.e., objects of kind `type`) as if they were terms. Reserved atomic types include the type of *states* (`state`), *principals* (`princ`) and *messages* (`msg`). We write $\{X : T\}T'$ for a dependent type, simplifying this syntax into $T \to T'$ when X does not occur free in T'. A *context* (called signature in [4,5]), is a list of declarations of term constants and type families. A *state* is a comma-separated multiset of terms (of type `state`). We later use the comma for message concatenation as well, an overloading that is disambiguated by the surrounding context. A *rule* relates two states S and S'. The latter can be prefixed by a sequence of existential declarations (e.g., for creating nonces), while other variables in the rule will often be universally quantified at its head. *Roles* are nonempty sequences of rules prefixed by zero or more existential predicate declarations. We assume that MSR specifications satisfy a *restricted format*, where the existential predicates are used to introduce local intermediate states for sequentializing the rules inside a role. A role has a distinguished *owner* P, which can be either an arbitrary principal in *generic roles*, or a fixed principal (e.g., a server) for *anchored roles*, that are introduced by the keyword `FOR` (which is not a binder).

MSR's actual syntax, as described in [4,5,7] has other constructions, that we either ignore for simplicity or leave out for future work. In particular, we assume that MSR's native subtyping is emulated by explicit coercions, that MSR's module structure has been expanded into a single sequence of declarations, and that all typing information is explicit, while MSR allows pointwise reconstruction. Simple preprocessing or standard techniques suffice to account for these discrepancies. In this paper, we do not treat other features of MSR, in particular guarded rules, equations, and a syntactic check known as data access specification. They will be the subject of future work.

We will rely on the Otway-Rees authentication protocol [12] to illustrate the use of MSR. In this protocol, an initiator, A, wants to obtain a short-term secret key k_{AB} to communicate securely with a responder B. They rely on a server $\mathtt{S}$, with whom both share long-term secret keys k_{AS} and k_{BS} respectively, to generate this new key. The "usual notation" for this protocol is as follows:

(i) $A \to B : n, A, B, \{n_A, n, A, B\}_{k_{AS}}$

(ii) $B \to \mathtt{S} : n, A, B, \{n_A, n, A, B\}_{k_{AS}}, \{n_B, n, A, B\}_{k_{BS}}$

(iii) $\mathtt{S} \to B : n, \{n_A, k_{AB}\}_{k_{AS}}, \{n_B, k_{AB}\}_{k_{BS}}$

(iv) $B \to A : n, \{n_A, k_{AB}\}_{k_{AS}}$

Here, A and B range over arbitrary principals, $\mathtt{S}$ is a fixed principal, the key server. Moreover, n, n_A and n_B are nonces, freshly generated values aimed at avoiding the replay of old messages.

As mentioned earlier, we assume appropriate declarations of types other than `princ`, `state` and `msg` and their elements. For this example, we rely on the type `nonce`, type families `ltK` and `stK` (for long- and short-term keys, respectively), and concatenation (overloaded as the infix ",") and encryption (written $\{\_\}_\_$) as additional message constructor. We use the state predicate $\mathtt{N}$ for representing messages in transit on the network. The superscripts $\mathtt{p}$, $\mathtt{n}$ and $\mathtt{k}$ represent coercions from principals, nonces and short-term keys to messages, respectively and $\mathtt{l}$ denotes the coercion from long-term keys to the shared keys expected by the encryption function (full MSR uses subsorting instead). The complete MSR specification can be found in [4]. Here we show only the generic role for the responder (B):

$$
\begin{array}{l}
\mathtt{ROLE}\ 2 : \forall B : \mathtt{princ}\,.\\
\quad \exists L : \{B : \mathtt{princ}\}\mathtt{princ} \to \mathtt{nonce} \to \mathtt{nonce} \to (\mathtt{ltK}\ B\ \mathtt{S}) \to \mathtt{state}\,.\\
\quad \mathtt{RULE}\ 21 : \left(\begin{array}{l}
\forall A : \mathtt{princ}\,.\ \forall n : \mathtt{nonce}\,.\ \forall k_{BS} : (\mathtt{ltK}\ B\ S)\,.\ \forall X : \mathtt{msg}\,.\\
\mathtt{N}\ (n^{\mathtt{n}}, A^{\mathtt{p}}, B^{\mathtt{p}}, X) \longrightarrow \exists n_B : \mathtt{nonce}\,.\\
\qquad\qquad\qquad\qquad\quad \mathtt{N}\ (n^{\mathtt{n}}, A^{\mathtt{p}}, B^{\mathtt{p}}, X, \{n_B^{\mathtt{n}}, n^{\mathtt{n}}, A^{\mathtt{p}}, B^{\mathtt{p}}\}_{k_{BS}^{\mathtt{l}}}),\\
\qquad\qquad\qquad\qquad\quad (L\ B\ A\ n\ n_B\ k_{BS})
\end{array}\right)\\
\quad \mathtt{RULE}\ 22 : \left(\begin{array}{l}
\forall A : \mathtt{princ}\,.\ \forall n, n_A : \mathtt{nonce}\,.\ \forall Y : \mathtt{msg}\,.\\
\forall k_{BS} : (\mathtt{ltK}\ B\ S)\,.\ \forall k_{AB} : (\mathtt{stK}\ A\ B)\,.\\
\mathtt{N}\ (n^{\mathtt{n}}, Y, \{n_B^{\mathtt{n}}, k_{AB}^{\mathtt{k}}\}_{k_{BS}^{\mathtt{l}}}), (L\ B\ A\ n\ n_B\ k_{BS}) \longrightarrow \mathtt{N}\ (n^{\mathtt{n}}, Y)
\end{array}\right)
\end{array}
$$

This role is generic and has two rules which are both in the scope of the quantifiers for B and L. In contrast, the role for $\mathtt{S}$ would be an anchored role, since $\mathtt{S}$ is a fixed principal.

The operational semantics of MSR [4,5] uses transition judgments that transform *configurations* of the form $[S]^R_\Sigma$. Here S is an MSR state, R is an *active roles set* which collects the *active roles*, i.e., instantiated and possibly partially executed roles, available at the next step, and Σ is an MSR dynamic context (called signature in [4,5]), which accounts for all the dynamically generated fresh constants available to R in state S. Using a slightly richer syntax than [4,5] we write an active role in the form $\texttt{ACTROLE}\ i : \texttt{FOR}\ A : \texttt{princ}\,.\ \texttt{WITH}\ \boldsymbol{N} : \boldsymbol{T}\,.\ \boldsymbol{\rho}$ (again $\texttt{FOR}$ and $\texttt{WITH}$ are no binders), meaning that it is the instance of either a generic role $\texttt{ROLE}\ i : \forall P : \texttt{princ}\,.\ \exists \boldsymbol{L} : \boldsymbol{T}\,.\ \boldsymbol{\rho}$ with P instantiated by A and $\boldsymbol{L}$ instantiated by $\boldsymbol{N}$, or the instance of an anchored role $\texttt{ROLE}\ i : \texttt{FOR}\ A : \texttt{princ}\,.\ \exists \boldsymbol{L} : \boldsymbol{T}\,.\ \boldsymbol{\rho}$ with $\boldsymbol{L}$ instantiated by $\boldsymbol{N}$.

In this paper we initially rely on two judgment forms to describe transitions: Given declarations $\mathcal{D}$ and roles $\mathcal{P}$, the judgment $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I_i^{A,\boldsymbol{N}}} [S]^{R'}_{\Sigma'}$ denotes the full instantiation, i.e., instantiation of all outer universal and existential quantifiers, of the role i (A and $\boldsymbol{N}$ define the particular instantiation). We write $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{E_{i,j}^{A,\boldsymbol{N}}} [S']^{R'}_{\Sigma'}$ to denote a transition resulting in the application of a rule j from the active role i (the instance with A and $\boldsymbol{N}$), followed by the removal of the active i if it has been fully executed. If one of these transitions can be derived we simply write $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I,E} [S']^{R'}_{\Sigma'}$. The rules for a marginally more abstract version of this semantics can be found in [4,5].

4 Rewriting Logic with Dependent Types

The *open calculus of constructions (OCC)*, from which we derive the rewriting logic with dependent types (RWLDT) by instantiation and restriction, is a family of type theories that are concerned with three classes of terms: *elements*, *types* and *universes*. Types serve as an abstraction for collections of elements, and universes as an abstraction for collections of types.

OCC is parameterized by OCC signatures defining the universe structure. In this paper we use a fixed signature $\Sigma = (\mathcal{S}, \texttt{Type}, :, \mathcal{R}, \leq)$ with *predicative* universes $\mathcal{S} = \{\texttt{Type}, \texttt{Type}_1, \texttt{Type}_2, \ldots\}$, which form a cumulative predicative hierarchy. This means that we have $\texttt{Type} : \texttt{Type}_1 : \texttt{Type}_2 \ldots$, a subtyping relation $\texttt{Type} \leq \texttt{Type}_1 \leq \texttt{Type}_2 \ldots$ (also called subuniverse relation), and $(s, s', s \sqcup s') \in \mathcal{R}$ for all $s, s' \in \mathcal{S}$, where $\sqcup$ denotes the least upper bound w.r.t. $\leq$.[3]

The formal system of OCC is designed to make sense under the *propositions-as-types interpretation*, where propositions are interpreted as types and proofs

[3] The effect of this choice of $\mathcal{R}$, a standard parameter for pure type systems [1], is that for arbitrary types $S : s$ (in a context Γ) and $T : s'$ (in a context $\Gamma, X : S$) with $s, s' \in \mathcal{S}$ we can form the dependent type $\{X : S\}T : s''$ (in Γ) for $s'' = s \sqcup s'$.

are interpreted as elements of these types. Since in OCC there is no a priori distinction between *terms* and *types*, and furthermore between *types* and *propositions*, we use all these notions synonymously.

OCC has the standard constructs known from pure type systems (cf. [1,14,15]) and a few additional ones. An *OCC term* can be one of the following: a *universe* s, a *variable* X, a typed *λ-abstraction* $[X : S]M$, a *dependent function type* $\{X : S\}T$, a *type assertion* $M : T$, an *ϵ-construct* $\epsilon\ A$ to denote an irrelevant proof of a proposition A, a *propositional equality* $M = N$, or one of three flavors of *operational propositions*, written as $||\ A$, $!!\ A$, or $??\ A$. Here and in following we usually use M, N, P, Q, S, T, U, V, A, and B to range over OCC terms, and X, Y, Z to range over names. Operational propositions can either be *structural propositions* designated by $||$, *computational propositions* designated by $!!$, or an *assertional propositions* designated by $??$. Subsequently, we use τ to range over these three flavors $\{||, !!, ??\}$.

OCC contexts are lists of *declarations* of the form $X : S$. The empty context is written as $[]$. Typically, we use Γ to range over OCC contexts.

An *OCC specification* is simply an OCC context Γ in this paper. Such a specification can introduce *rewrite predicates* by declarations of the form $R : \{Y : S\}\{Y' : S\}\ T \to \texttt{Type}$. The idea is that each rewrite predicate can be regarded as a labeled transition system, which can be executed in a very similar way as rewriting logic specifications. Note that $R : \{Y : S\}\{Y' : S\}\ T \to \texttt{Type}$ is the declaration of a ternary predicate R where S is the type of states and T is the type of actions, which could range from atomic labels to rewrite proofs, depending on the requirements of the application. In the case where the type T does not depend on Y and Y', this declaration takes the form $R : S \to S \to T \to \texttt{Type}$.

Since we are working with a predicative instance of OCC, it is entirely straightforward to define a model-theoretic semantics based on classical set theory with suitable universes [14]. The appropriate level of abstraction for this paper is, however, the operational semantics, which is given by the formal system of OCC. It is a direct generalization of the operational semantics of rewriting logic [11] and its membership-equational sublogic [2] as implemented in Maude [9].

The *formal system of OCC* defines *derivability* of OCC judgments $\Gamma \vdash J$ and has been shown to be sound w.r.t. the set-theoretic semantics [14]. For brevity we only give an informal explanation of all judgments and their intuitive operational meaning.

- The *type inference judgment* $\Gamma \vdash M \to: S$ asserts that the term M is an *element* of the *inferred type* S in the context Γ. Operationally, Γ and M are given and S is obtained by syntax-directed type inference and possible reduction using computational equations modulo the structural equations of Γ.

- The *typing judgment* $\Gamma \vdash M : S$ asserts that M is an *element* of *type* S in the context Γ. Operationally, Γ, M and S are given and verifying $\Gamma \vdash M : S$ amounts to type checking. Type checking is always reduced to type inference and the verification of an assertional subtyping judgment.
- The *structural equality judgment* $\Gamma \vdash || \ (M = N)$ is used to express that M and N are considered to be structurally equal elements in the context Γ. Operationally, structural equality is realized by a suitable term representation so that structurally equal terms cannot be distinguished when they participate in computations.
- The *computational equality judgment* $\Gamma \vdash !! \ (M = N)$ is the judgment that defines the notion of reduction for the simplification of terms. The judgment states that the element M can be reduced to the element N in the context Γ. Operationally, Γ and M are given and N is the result of reducing M using the computational equations in Γ modulo the structural equations in Γ.
- The *assertional judgment* $\Gamma \vdash ?? \ A$ states that A is provable by means of the operational semantics in the context Γ. Operationally, Γ and A are given and the judgment is verified by a combination of reduction using the computational equations and exhaustive goal-oriented search using the assertional propositions in Γ. Both processes take place modulo the structural equations in Γ.
- The *assertional equality judgment* $\Gamma \vdash ?? \ (M = N)$ states that M and N are assertionally equal in Γ, a notion that treats equality as a predicate and subsumes the structural and computational equality judgments. Operationally, Γ, M and N are given and the judgment is verified like other assertional judgments in a goal-oriented fashion.
- The *assertional subtyping judgment* $\Gamma \vdash ?? \ (S \leq T)$ subsumes the assertional equality judgment and states that S is a subtype of T in Γ as a consequence of the cumulativity of the universe hierarchy. Operationally, Γ, S and T are given and the judgment is verified like other assertional judgments in a goal-oriented fashion.
- The *computational rewrite judgment* $\Gamma \vdash !! \ (R \ M \ M' \ P)$ expresses that by means of the computational rewrite rules specified in Γ for the rewrite predicate R the element M can be rewritten to the element M' and this rewrite is labeled by the element P. Operationally, Γ and M are given and M' is computed by the application of a computational rewrite rule in Γ modulo the computational and structural equations in Γ. In addition, an

abstract witness P for this rewrite is constructed.

By fixing the signature at the beginning of this section, we have introduced a particular instance of OCC. For the purpose of this paper we further restrict this instance by requiring that specifications use a unique fixed rewrite predicate $\mathtt{R}$ which is declared as $\mathtt{R} : S \to S \to T \to \mathtt{Type}$. The idea is that this ternary rewrite predicate precisely corresponds to the labeled rewrite relation of rewriting logic. To remind us of this correspondence we refer to this restricted sublanguage of OCC in the following as *rewriting logic with dependent types* (RWLDT).[4] Since $\mathtt{R}$ is unique, we can use the usual notation $[P] : M \Rightarrow N$ instead of the less intuitive $(\mathtt{R}\ M\ N\ P)$. Similarly, we use $\Gamma \vdash !!\ [P] : M \Rightarrow N$ to denote the corresponding computational rewrite judgment.

5 Mapping MSR to RWLDT

In this section we give a precise definition of our mapping from MSR into RWLDT. The translations of kinds, types, terms, and states are very direct. The translation of roles and rules may appear somewhat technical, but the underlying idea is simple. To make it better accessible to the reader we introduce the mapping of roles and rules in three steps: In Sections 5.1–5.3, we give an initial mapping that is correct in a rather obvious way, and then we deal with some deficiencies of this mapping in two further steps in Sections 5.4 and 5.5. The result is a mapping which is not only correct but ensures executability of the resulting RWLDT specification (in the sense of ordinary rewrite systems). It furthermore avoids the introduction of any superfluous intermediate states that would lead to unnecessary inefficiencies, especially if we use the result of the translation for symbolic state space exploration.

5.1 Initial Context

The MSR multiset union constructs will be translated to an ordinary RWLDT function $\mathtt{union}$. To this end, we define *initial_context* as an OCC context that contains the following declarations.

There are the structural axioms for multisets:

$\mathtt{state} : \mathtt{Type},$

$\mathtt{empty} : \mathtt{state},$

$\mathtt{union} : \mathtt{state} \to \mathtt{state} \to \mathtt{state},$

$\mathtt{union\_comm} : ||\ \{X, Y : \mathtt{state}\}(\mathtt{union}\ X\ Y) = (\mathtt{union}\ Y X),$

[4] Compared with [14] we have omitted assertional rewrite judgments in our presentation of OCC, because we do not need rewrite conditions in this paper. Such conditions are admitted in RWL and hence in the most general version of RWLDT.

$$\texttt{union\_assoc} : \mid\mid \{X, Y, Z : \texttt{state}\}\ (\texttt{union}\ (\texttt{union}\ X\ Y) Z) = (\texttt{union}\ X\ (\texttt{union}\ Y\ Z)),$$

$$\texttt{union\_id} : \mid\mid \{X : \texttt{state}\}(\texttt{union}\,\texttt{empty}\ X) = X.$$

The initial context also contains the following declarations, which we describe only intuitively. Their purpose will become clear as we lay out the translation.

- $\texttt{princ} : \texttt{Type}$, $\texttt{msg} : \texttt{Type}$. The types of principals and messages, respectively.
- $\texttt{T}ij : \texttt{princ} \to \boldsymbol{T} \to \texttt{state}$. For each role i with existential quantifier types $\boldsymbol{T}$ and with a rule j, a token $(\texttt{T}ij\ A\ \boldsymbol{N})$ will be used to represent the fact that role i has been instantiated with values A and $\boldsymbol{N}$.
- $\texttt{nat} : \texttt{Type}$, and $\texttt{0} : \texttt{nat}$ and $\texttt{S} : \texttt{nat} \to \texttt{nat}$. Natural numbers are used to index fresh symbols, i.e., symbols that have not been used in the past.
- $\texttt{F} : \texttt{nat} \to \texttt{state}$. As an invariant of our representation there is a unique $(\texttt{F}\ N)$ that holds the next available fresh index N.
- $\bar{V} : \texttt{nat} \to V$. For each type V which can be generated, this function allows us to index (some of) its elements by natural numbers, a way to generate fresh symbols of this type.
- $\texttt{E} : \{T : \texttt{Type}\}T \to \texttt{state}$. The term $(\texttt{E}\ T\ M)$ expresses the fact that M is an element of type T, as part of the state. We will use this predicate only in Section 5.4.
- $\texttt{act} : \texttt{Type}$, $\texttt{A}i : \texttt{act}$ for each role i, and $\texttt{A}ij : \texttt{act}$ for each of its rules j. These constants are used to label the rewrite rules resulting from the translation.

5.2 Translating Kinds, Types, Terms, States, and Contexts

For the following we assume that MSR specifications do not use names introduced by *initial_context* other than state, princ and msg. We also assume that all declared and bound variables are distinct. This allows a clear presentation of the main ideas without worrying about renaming and capturing. We then define the translation of MSR kinds, types, states, and contexts as follows:

- $\textit{kind}(\texttt{type}) = \texttt{Type}$
  $\textit{type}(\{X : T\}K) = \{X : \textit{type}(T)\}\textit{kind}(K)$
- $\textit{type}(X) = X$
  $\textit{type}(\texttt{state}) = \texttt{state}$
  $\textit{type}(\texttt{princ}) = \texttt{princ}$
  $\textit{type}(\texttt{msg}) = \texttt{msg}$
  $\textit{type}(T\ M) = (\textit{type}(T)\ \textit{term}(M))$
  $\textit{type}(\{X : T\}T') = \{X : \textit{type}(T)\}\textit{type}(T')$

- $\mathit{term}(X) = X$
  $\mathit{term}(M\ N) = (\mathit{term}(M)\ \mathit{term}(N))$
- $\mathit{state}(\cdot) = \texttt{empty}$
  $\mathit{state}(S, S') = (\texttt{union}\ \mathit{state}(S)\ \mathit{state}(S'))$
  $\mathit{state}(M) = \mathit{term}(M)$
- $\mathit{context}(\cdot) = \mathit{initial\_context}$
  $\mathit{context}(\mathcal{D}, X : K) = \mathit{context}(\mathcal{D}), X : \mathit{kind}(K)$
  $\mathit{context}(\mathcal{D}, X : T) = \mathit{context}(\mathcal{D}), X : \mathit{type}(T)$.

Subsequently, $(\texttt{union}\ (S_1, \ldots, S_n))$ abbreviates $(\texttt{union}\ S_1\ (\texttt{union}\ (S_2, \ldots, S_n)))$, and $(\texttt{union}\ ())$ abbreviates $\texttt{empty}$. We also refer to this term as the *formal multiset* of $S_1, \ldots, S_n$.

The adequacy of this translation is expressed by the following theorem:

Theorem 5.1 If $\mathcal{D}$ is a well-typed MSR context then:

(i) If K is an MSR kind, then
$\mathcal{D} \vdash K\ \mathit{kind}$ in MSR iff $\mathit{context}(\mathcal{D}) \vdash \mathit{kind}(K) : \texttt{Type}_1$ in RWLDT.

(ii) If in addition $\mathcal{D} \vdash K\ \mathit{kind}$ and T is an MSR type, then
$\mathcal{D} \vdash T : K$ in MSR iff $\mathit{context}(\mathcal{D}) \vdash \mathit{type}(T) : \mathit{kind}(K)$ in RWLDT.

(iii) If in addition $\mathcal{D} \vdash T : K$ and M is an MSR term, then
$\mathcal{D} \vdash M : T$ in MSR iff $\mathit{context}(\mathcal{D}) \vdash \mathit{term}(M) : \mathit{type}(T)$ in RWLDT.

(iv) If S is an MSR state, then
$\mathcal{D} \vdash S : \texttt{state}$ in MSR iff $\mathit{context}(\mathcal{D}) \vdash \mathit{state}(S) : \texttt{state}$ in RWLDT.

Furthermore, $\mathcal{D}$ is well-typed iff $\mathit{context}(\mathcal{D})$ is well-typed.

Proof Sketch. First of all, it is straightforward to verify that each MSR inference rule [7] can be simulated by one or more inference rules of RWLDT [14]. As a consequence, the ⇒ direction of the equivalences (i)–(iv) holds.

To deal with the more interesting ⇐ direction of these equivalences, we first observe that several features of RWLDT are not relevant for the purpose of this proof. Our representation does not exploit higher universes (beyond $\texttt{Type}_0$) or universe subtyping in any essential way (the only use of $\texttt{Type}_1$ in our representation is to serve as a type of kinds). Both, computational equations and assertional propositions of RWLDT are not used. Structural equations are only used to represent MSR states (multisets) and they are used in such a way that they precisely represent the MSR syntax. The computational rewrite axioms of RWLDT do not have any impact on type checking, so they can be ignored here. Another major simplification is that MSR and hence the representation in RWLDT does not use λ-abstractions, and the type assertions and the ϵ-operator of RWLDT are not used either. As a consequence, many of the inference rules of RWLDT [14] can be ignored or reduced to trivial cases, because they cannot have been used in the RWLDT derivation or have only been used in a trivial form. For instance, without λ-abstractions and computational equations the computational equality reduces to structural equality. Without assertional propositions, inference rules for assertional propositions other than assertional equality and assertional subtyping are superfluous. Without the use of higher universes, assertional subtyping coincides with assertional equality which reduces to structural equality and α-conversion. Now it is easy to verify the $\Leftarrow$ direction of (i)–(iv) by simulating each inference rule of the simplified RWLDT using inference rules of MSR [7]. A slight remaining difficulty is to overcome the gap between implicit α-conversion in MSR and explicit α-conversion in RWLDT (including its more general notion of context), but the proof techniques for pure type systems used in [15] can be easily adapted to our simple setting.

Finally, the proof that $\mathcal{D}$ is well-typed iff $context(\mathcal{D})$ is well-typed, can be done by induction over the length of $\mathcal{D}$ using the previous statements. □

It is important to note that this theorem does not imply that each well-typed RWLDT term in the context $context(\mathcal{D})$ is a representation of an MSR term, type, or kind. For instance, a counterpart of the RWLDT type $\texttt{Type} \rightarrow \texttt{Type} : \texttt{Type}_1$ does not exists in MSR. Similarly, we could use λ-abstractions and other constructs in RWLDT, but they do not have counterparts in MSR. In fact, the restricted syntax of MSR and our representation carves out a sublanguage of RWLDT, and only terms in this sublanguage are used inside the operational semantics. The specification itself, however, requires constructs such as structural equations and computational rewrite axioms, which are outside of this sublanguage.

5.3 Translating Roles and Rules

To further simplify the presentation, we assume that the identifier of the i-th role is i and the identifier of its j-th rule is j. We then define the translation of MSR roles and rules as follows (using $\mathcal{P}$ and $\mathcal{P}'$ to range over role sequences):

- $roles(\cdot) = [].$
  $roles(\texttt{ROLE}\ i\ \texttt{FOR}\ A : \texttt{princ}\,.\,\exists \boldsymbol{L} : \boldsymbol{T}\,.\,\boldsymbol{\rho}) =$
  $\qquad \texttt{R}i : role(i, A, \exists \boldsymbol{L} : \boldsymbol{T}\,.\,\boldsymbol{\rho}).$
  $roles(\texttt{ROLE}\ i : \forall P : \texttt{princ}\,.\,\exists \boldsymbol{L} : \boldsymbol{T}\,.\,\boldsymbol{\rho}) =$
  $\qquad \texttt{R}i : \{P : \texttt{princ}\} role(i, P, \exists \boldsymbol{L} : \boldsymbol{T}\,.\,\boldsymbol{\rho}).$
  $roles(\mathcal{P}, \mathcal{P}') = roles(\mathcal{P}), roles(\mathcal{P}').$
- $role(i, P, \exists \boldsymbol{L} : \boldsymbol{T}\,.\,\boldsymbol{\rho}^n) =$
  $\qquad \{Z, Z' : \texttt{nat}\}\{\boldsymbol{L} : type(\boldsymbol{T})\} fresh(Z, \boldsymbol{L}, \boldsymbol{T}, Z') \rightarrow$

$$[\texttt{A}i] : (\texttt{F}\ Z) \Rightarrow (\texttt{union}\ ((\texttt{T}i1\ P\ \boldsymbol{L}), \ldots, (\texttt{T}in\ P\ \boldsymbol{L})\ (\texttt{F}\ Z'))),$$
$$\mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \boldsymbol{\rho}^n_1), \ldots, \mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \boldsymbol{\rho}^n_n).$$

- $\mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \texttt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} \mathrel{.} M \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} \mathrel{.} N) =$
  $$\texttt{R}ij : \{P : \texttt{princ}\}\{Z, Z' : \texttt{nat}\}\{\boldsymbol{L} : \mathit{type}(\boldsymbol{T})\}$$
  $$\{\boldsymbol{X} : \mathit{type}(\boldsymbol{U})\}\{\boldsymbol{Y} : \mathit{type}(\boldsymbol{V})\}\mathit{fresh}(Z, \boldsymbol{Y}, \boldsymbol{V}, Z') \to$$
  $$[\texttt{A}ij] : (\texttt{union}\ ((\texttt{T}ij\ P\ \boldsymbol{L}), (\texttt{F}\ Z), \mathit{state}(M))) \Rightarrow$$
  $$(\texttt{union}\ ((\mathit{state}(N), (\texttt{F}\ Z')))).$$
- $\mathit{fresh}(Z, \boldsymbol{Y}^y, \boldsymbol{V}^y, Z') =$
  $$\boldsymbol{Y}^y_1{=}\bar{\boldsymbol{V}}^y_1(Z)), \boldsymbol{Y}^y_2{=}\bar{\boldsymbol{V}}^y_2(\texttt{s}(Z)), \ldots, \boldsymbol{Y}^y_y{=}\bar{\boldsymbol{V}}^y_y(\texttt{s}^{y-1}(Z)), Z'{=}\texttt{s}^y(Z).$$

In the last equation we assume that for type V there is an injection with the same name $\bar{V} : \texttt{nat} \to V$, a declaration that needs to be included in *initial_context*.

Above we use $A_1, \ldots, A_n \to B$ to abbreviate $A_1 \to \ldots \to A_n \to B$, which here means that $A_1, \ldots, A_n$ are conditions. It should also be noted, however, that the use of conditions here is not essential, because they are all of the form $Y{=}Q$ and hence can trivially be eliminated. We only use conditions for better readability and to maintain a more direct correspondence to the MSR syntax.

The idea behind the definition of *role* is that it maps each MSR role i to several RWLDT rewrite axioms: There is one rewrite axiom labelled $\texttt{A}i$, representing the instantiation of this role. In addition, there is one rewrite axiom labelled $\texttt{A}ij$ (generated by *rule*) for each of its rules j. The first axiom $\texttt{A}i$, apart from the generation of fresh terms needed for the new instance, generates tokens $(\texttt{T}i1\ P\ \boldsymbol{L}), \ldots, (\texttt{T}in\ P\ \boldsymbol{L})$, representing the fact that none of the rules of this role have been executed yet. Each of the remaining axioms $\texttt{A}ij$ simulates the corresponding MSR rule j, so that each application of a rule removes its corresponding token. This realizes the MSR policy that rules of active roles can only be executed once.

The following lemma expresses that the generation of fresh symbols is correctly represented in RWLDT.

Lemma 5.2 (Freshness Invariance) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification. The representation in RWLDT maintains the following invariant: If there is a term of the form $(\texttt{F}\ (\texttt{s}^n\texttt{0}))$ in the RWLDT state and no other occurrence of $\texttt{F}$ then for each k (including 0) the term $(\texttt{s}^{n+k}\texttt{0})$, and consequently $V(\texttt{s}^{n+k}\texttt{0})$, is fresh, i.e., it does not occur in any other part of the state.

Proof Sketch. Can be directly verified as an inductive invariant for each of the RWLDT rewrite axioms in our representation, using our earlier assumption on the initial context that `0` and `s` are not used in the MSR specification. □

To express the relationship between MSR and its representation we also need a representation of dynamic entities such as active roles:

- $\mathit{actroles}(\cdot) = \texttt{empty}$.
  $\mathit{actroles}(\texttt{ACTROLE}\, i :\ \texttt{FOR}\ A : \texttt{princ}\,.\ \texttt{WITH}\ \boldsymbol{N} : \boldsymbol{T}\,.\ \boldsymbol{\rho}) =$
  the formal multiset of all $(\texttt{T}ij\ A\ \boldsymbol{N})$ s.t. $\boldsymbol{\rho}$ contains rule j.
  $\mathit{actroles}(R, R') = (\texttt{union}\ \mathit{actroles}(R)\ \mathit{actroles}(R'))$,
  where R and R' range over active role sets.

Recall that $\texttt{ACTROLE}\ i :\ \texttt{FOR}\ A : \texttt{princ}\,.\ \texttt{WITH}\ \boldsymbol{N} : \boldsymbol{T}\,.\ \boldsymbol{\rho}$ is the form of an active role, i.e., one that has been (fully) instantiated and possibly partially executed. The fact that the active role set contains an active role of this form corresponds to the fact that for each rule j of the active role (i.e., a rule that has not been executed yet) the term $(\texttt{T}ij\ A\ \boldsymbol{N})$ is part of the distributed state in the representation.

The subsequent theorem justifies the use of representations of MSR configurations of a particular form in all the remaining theorems.

Theorem 5.3 (Representation Invariance) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification. If $\mathit{context}(\mathcal{D}),\ \mathit{roles}(\mathcal{P}) \vdash\ !!\ [P] : M \Rightarrow M'$ and M is a representation of an MSR configuration, i.e., of the form $(\texttt{union}\ ((\texttt{F}\ (\texttt{s}^n\ \texttt{0})), \mathit{state}(S), \mathit{actroles}(R)))$ for some n, some MSR state S, and some MSR active role set R, then M' is a representation of an MSR configuration as well.

Proof Sketch. Again this is an inductive invariant that obviously holds for each of the RWLDT rewrite axioms in our representation. □

For the proof of the main theorem, we need the following lemma.

Lemma 5.4 (Representation Uniqueness) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification. Then each MSR active role set R reachable in the operational semantics of MSR can be uniquely reconstructed from its representation $\mathit{actroles}(R)$.

Proof Sketch. Observe that active role sets can contain only elements that can actually be obtained by (full) instantiation of known roles followed by removal of some of its rules (after they are executed). We need to consider two cases:

(i) If there is at least one token (left) that represents the active role, then this token carries the full information, namely i, A, and $\boldsymbol{N}$, to determine the initial role instance. Unfortunately, we need to argue that together all tokens of the form $(\texttt{T}ij\ A\ \boldsymbol{N})$ uniquely determine the rules of this active role, i.e., the rules that still need to be executed. The only potential source of confusion is that the representation may contain several instances of the same active role i executing concurrently with the same values A and $\boldsymbol{N}$. Since $\boldsymbol{N}$ is generated fresh the confusion can only occur if $\boldsymbol{N}$ is the empty list, i.e., when the role does not have any existential quantifiers. Because of the restricted format of MSR specifications (existential predicates are used to sequentialize the rules inside a role) this means that the role can only have a single rule, and hence it can be in only one state, namely the state after it has been instantiated but not executed.

(ii) In case there is no token (left) that represents the active role, the role must have been fully executed, and hence by definition of our operational semantics for MSR it cannot be part of the active role set. Hence, again its contribution to the active role set is uniquely determined. □

Lemma 5.5 Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification, let S, S' be MSR states, and let R, R' be MSR active roles sets. Then for all n, k we have the following equivalences:

There are MSR contexts Σ^n, Σ'^{n+k} s.t. $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I_i^{A,\boldsymbol{N}}} [S']^{R'}_{\Sigma'}$ iff

$$\mathit{context}(\mathcal{D}),\ \mathit{roles}(\mathcal{P}) \vdash\ !!\ [\mathtt{A}i] : (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{S}^n\ \mathtt{0})), \mathit{state}(S), \mathit{actroles}(R))) \Rightarrow \\ (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{S}^{n+k}\ \mathtt{0})), \mathit{state}(S'), \mathit{actroles}(R'))),$$

There are MSR contexts Σ^n, Σ'^{n+k} s.t. $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{E_{i,j}^{A,\boldsymbol{N}}} [S']^{R'}_{\Sigma'}$ iff

$$\mathit{context}(\mathcal{D}),\ \mathit{roles}(\mathcal{P}) \vdash\ !!\ [\mathtt{A}ij] : (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{S}^n\ \mathtt{0})), \mathit{state}(S), \mathit{actroles}(R))) \Rightarrow \\ (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{S}^{n+k}\ \mathtt{0})), \mathit{state}(S'), \mathit{actroles}(R'))).$$

In both statements we identify terms that are structurally equal in $\mathit{context}(\mathcal{D})$.

Proof Sketch. First, an observation that simplifies the proof of both statements of the lemma: We can verify using the previous lemma that

$$(\mathtt{union}\ ((\mathtt{F}\ (\mathtt{S}^n\mathtt{0})), \mathit{state}(S), \mathit{actroles}(R)))$$

can only represent the MSR state S, the MSR active role set R, and hence only an MSR configuration $[S]^R_\Sigma$ for some dynamic context Σ. Similarly,

$$(\mathtt{union}((\mathtt{F}\ (\mathtt{S}^{n+k}\ \mathtt{0})), \mathit{state}(S'), \mathit{actroles}(R')))$$

can only represent a configuration $[S']^{R'}_{\Sigma'}$ again for some dynamic context Σ'.

To prove the first equivalence of the lemma, note that the left-hand side expresses that role i is instantiated for some principal A (if it is generic, otherwise A is already fixed), the existential quantifiers are instantiated with fresh symbols, and the corresponding role instance is added to the active role set. According to the change in the MSR dynamic context on the left-hand side (Σ^n becomes Σ'^{n+k}) k fresh symbols are generated, which means that the role has k existential quantifiers. We need to verify that this step in the operational semantics of MSR is equivalent to the right-hand side, i.e., to the application of the rewrite axiom labeled $\mathtt{A}i$ (see definition of *role*) in RWLDT. Using the freshness invariance lemma is it easy to see that the existential quantifiers of role i are correctly instantiated using k fresh terms $\boldsymbol{N}$ which are generated in this process. Apart from maintaining the freshness information, the only effect of the rule is that the terms $(\mathtt{T}i1\ A\ \boldsymbol{N}), \ldots, (\mathtt{T}in\ A\ \boldsymbol{N})$ are added to the RWLDT state. This can only correspond to the addition of a new instance of role i to the active role set.

For the second equivalence of the lemma, note that the left-hand side expresses that an instance of role i is selected from the active role set and its rule j, which is of the form $\mathtt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} .\ M \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} .\ N$, is executed. According to the change in the dynamic MSR context on the left-hand side (Σ^n becomes Σ'^{n+k}) k fresh symbols are generated, which means that the rule has k existential quantifiers. We again need to verify that this step in the operational semantics of MSR is equivalent to the right-hand side, i.e., the application of the rewrite axiom labeled $\mathtt{A}ij$ (see definition of *rule*) in RWLDT. Using the freshness lemma it is easy to see that the existential quantifiers of rule j are correctly instantiated using k fresh terms. Apart from maintaining the freshness information, the rule has two effects: It replaces the term $state(M')$ by $state(N')$ (the terms M' and $'N'$ are instances of M and N, respectively), a step precisely corresponding to the execution of the MSR rule, and it removes the token $(\mathtt{T}ij\ A\ \boldsymbol{N})$, which can only correspond to the fact that the rule is removed from the active role, because it has been executed.

Obviously, for both equivalences the detailed proof would establish a bijection between the fresh symbols generated by MSR and the fresh terms generated in RWLDT. □

The following theorem summarizes the statements of the previous lemma:

Theorem 5.6 (Soundness and Completeness) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification, let S, S' be MSR states, and let R, R' be MSR active roles sets. Then for all n, k we have the following equivalence:
There are MSR contexts Σ^n,Σ'^{n+k} s.t. $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I,E} [S']^{R'}_{\Sigma'}$ iff
there exists P s.t. $context(\mathcal{D}),\ roles(\mathcal{P}) \vdash$
$!!\ [P] : (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{s}^n\ \mathtt{0})), state(S), actroles(R))) \Rightarrow$
$\quad (\mathtt{union}\ ((\mathtt{F}\ (\mathtt{s}^{n+k}\ \mathtt{0})), state(S'), actroles(R'))),$
where we identify terms that are structurally equal in $context(\mathcal{D})$.

5.4 Achieving Executability

Unfortunately, the resulting RWLDT specification is not necessarily executable in the ordinary sense of rewriting, since there may be rules with variables on the right-hand side that do not appear on the left-hand side and hence cannot be bound by matching. Therefore, we apply another simple transformation which makes certain types and their elements explicit in the state by making use of the predicate $\mathtt{E} : \{T : \mathtt{Type}\}T \rightarrow \mathtt{state}$. This leads to the following modifications:

- $\mathit{role}(i, P, \exists \boldsymbol{L} : \boldsymbol{T} . \boldsymbol{\rho}^n) =$
  $\{Z, Z' : \mathtt{nat}\}\{\boldsymbol{L} : \mathit{type}(\boldsymbol{T})\}\mathit{fresh}(Z, \boldsymbol{L}, \boldsymbol{T}, Z') \rightarrow$
  $[\mathtt{A}i] : (\mathtt{union}((\mathtt{E}\ \mathtt{princ}\ P), (\mathtt{F}\ Z)) \Rightarrow$
  $(\mathtt{union}((\mathtt{E}\ \mathtt{princ}\ P), (\mathtt{T}i1\ P\ \boldsymbol{L}), \ldots, (\mathtt{T}in\ P\ \boldsymbol{L}), (\mathtt{F}\ Z')))$
  $\mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \boldsymbol{\rho}^n_1),\ \ldots,\ \mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \boldsymbol{\rho}^n_n).$
- $\mathit{rule}(i, P, \boldsymbol{L}, \boldsymbol{T}, \mathtt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} . M \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} . N) =$
  $\mathtt{R}ij : \{P : \mathtt{princ}\}\{Z, Z' : \mathtt{nat}\}\{\boldsymbol{L} : \mathit{type}(\boldsymbol{T}ij)\}$
  $\{\boldsymbol{X} : \mathit{type}(\boldsymbol{U})\}\{\boldsymbol{Y} : \mathit{type}(\boldsymbol{V})\}\mathit{fresh}(Z, \boldsymbol{Y}, \boldsymbol{V}, Z') \rightarrow$
  $[\mathtt{A}ij] : (\mathtt{union}\ (ES, (\mathtt{T}ij\ P\ \boldsymbol{L}), (\mathtt{F}\ Z), \mathit{state}(M))) \Rightarrow$
  $(\mathtt{union}\ (ES, \mathit{state}(N), (\mathtt{F}\ Z')))$
  where ES is a formal multiset containing $(\mathtt{E}\ \boldsymbol{U}^x_1\ \boldsymbol{X}^x_1), \ldots, (\mathtt{E}\ \boldsymbol{U}^x_x\ \boldsymbol{X}^x_x)$.

The theorem is as before except for that we have to provide a sufficient amount of explicit typing information (see ES below) to perform a simulation step:

Theorem 5.7 (Soundness and Completeness) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification, let S, S' be MSR states, and let R, R' be MSR active roles sets. Then for all n, k we have the following equivalence:
There are MSR contexts Σ^n, Σ'^{n+k} s.t. $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I,E} [S']^{R'}_{\Sigma'}$ iff
there exists P, ES s.t. $\mathit{context}(\mathcal{D}),\ \mathit{roles}(\mathcal{P}) \vdash$
$!!\ [P] : (\mathtt{union}\ (ES, (\mathtt{F}\ (\mathtt{s}^n\mathtt{0}))), \mathit{state}(S), \mathit{actroles}(R))) \Rightarrow$
$(\mathtt{union}\ (ES, (\mathtt{F}\ (\mathtt{s}^{n+k}\ \mathtt{0})), \mathit{state}(S'), \mathit{actroles}(R')))$
where we identify terms that are structurally equal in $\mathit{context}(\mathcal{D})$, and ES is a formal multiset containing $(\mathtt{E}\ U\ Q)$ only for terms Q of type U.

Proof Sketch. The only modification to our previous representation (Section 5.3) is that we have added terms of the form $(\mathtt{E}\ U\ Q)$ to the rewrite axioms, such that as an obvious invariant these terms are preserved by applications of rewrite axioms. These terms cannot be confused or interact with any of the terms representing the MSR configuration, so that the original behavior is preserved (disregarding the newly introduced terms), assuming that the applicability of rewrite axioms is not compromised. To guarantee the latter, the theorem has been relaxed w.r.t. the previous one (Section 5.3) by adding the formal multiset ES to the state on the left-hand side of the rewrite judgment (and since ES is preserved it is added on the right-hand side as well). Since ES is existentially quantified it can be instantiated by any sufficient number of terms compensating for the ($\texttt{E}\ U\ Q$) that are now needed to apply the rewrite axioms. □

As a slight optimization, the term ($\texttt{E princ}\ A$) in the translation above can be dropped in the rewrite axiom $\texttt{A}i$ if it is the translation of an anchored rule, because in this case A is a constant declared in $\mathcal{D}$ and not a variable that needs to be determined by matching. Furthermore, ($\texttt{E}\ \boldsymbol{U}^x_j\ \boldsymbol{X}^x_j$) can be dropped from ES in the translation if $\boldsymbol{X}^x_j$ occurs in M, because in this case it can be bound by matching.

5.5 Eliminating Intermediate States

A drawback of the operational semantics of MSR defined in terms of the transition judgment $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I,E} [S']^{R'}_{\Sigma'}$ and our representation above is that role instantiation can occur anytime and arbitrarily often even if there is no subsequent use of the role. This is an unnecessary source of nondeterminism and nontermination and without any other means to control the execution it would prevent symbolic execution and analysis.

By considering a slightly more abstract semantics that composes role instantiation with the execution of a rule of this role, we can eliminate such superfluous intermediate states. For the modified operational semantics of MSR we write $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{(I)E} [S'']^{R''}_{\Sigma''}$ iff there exists a role i with a rule j (and A, $\boldsymbol{N}$) s.t.

- $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I_i^{A,\boldsymbol{N}}} [S']^{R'}_{\Sigma'}$ and $\mathcal{D}, \mathcal{P} \vdash [S']^{R'}_{\Sigma'} \longrightarrow_{E_{i,j}^{A,\boldsymbol{N}}} [S'']^{R''}_{\Sigma''}$ or
- $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{E_{i,j}^{A,\boldsymbol{N}}} [S'']^{R''}_{\Sigma''}$.

Our representation will be modified correspondingly as follows:

- $\mathit{role}(i, P, \exists \boldsymbol{L} : \boldsymbol{T} \,.\, \boldsymbol{\rho}^n) =$
  $\mathit{rule}_1(i, n, P, \boldsymbol{L}, \boldsymbol{\rho}^n_1), \mathit{rule}_2(i, n, P, \boldsymbol{L}, \boldsymbol{\rho}^n_n), \ldots,$
  $\mathit{rule}_1(i, n, P, \boldsymbol{L}, \boldsymbol{\rho}^n_n), \mathit{rule}_2(i, n, P, \boldsymbol{L}, \boldsymbol{\rho}^n_n).$
- $\mathit{rule}_1(i, n, P, \boldsymbol{L}, \boldsymbol{T}, \texttt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} \,.\, M \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} \,.\, N) =$
  $\texttt{R}ij1 : \{P : \texttt{princ}\}\{Z, Z', Z'' : \texttt{nat}\}\{\boldsymbol{L} : \mathit{type}(\boldsymbol{T})\}$
  $\{\boldsymbol{X} : \mathit{type}(\boldsymbol{U})\}\{\boldsymbol{Y} : \mathit{type}(\boldsymbol{V})\}$
  $\mathit{fresh}(Z, \boldsymbol{L}, \boldsymbol{T}, Z'), \mathit{fresh}(Z', \boldsymbol{Y}, \boldsymbol{V}, Z'') \rightarrow$
  $[\texttt{A}ij1] : (\texttt{union}\ ((\texttt{E princ}\ P), ES, (\texttt{F}\ Z), \mathit{state}(M))) \Rightarrow$
  $(\texttt{union}\ ((\texttt{E princ}\ P), ES, \mathit{state}(N), TS, (\texttt{F}\ Z'')))$
  where TS is the formal multiset containing
  ($\texttt{T}i1\ P\ \boldsymbol{L}$) $\ldots$ ($\texttt{T}in\ P\ \boldsymbol{L}$) with ($\texttt{T}ij\ P\ \boldsymbol{L}$) removed.
- $\mathit{rule}_2(i, n, P, \boldsymbol{L}, \boldsymbol{T}, \texttt{RULE}\ j : \forall \boldsymbol{X} : \boldsymbol{U} \,.\, M \longrightarrow \exists \boldsymbol{Y} : \boldsymbol{V} \,.\, N) =$
  $\texttt{R}ij2 : \{P : \texttt{princ}\}\{Z, Z', Z'' : \texttt{nat}\}\{\boldsymbol{L} : \mathit{type}(\boldsymbol{T})\}$

$$\begin{array}{l} \{\boldsymbol{X} : \mathit{type}(\boldsymbol{U})\}\{\boldsymbol{Y} : \mathit{type}(\boldsymbol{V})\}\mathit{fresh}(Z, \boldsymbol{Y}, \boldsymbol{V}, Z') \rightarrow \\ [\mathtt{A}ij2] : \mathtt{union}\ (ES, (\mathtt{T}ij\ P\ \boldsymbol{L}), (\mathtt{F}\ Z), \mathit{state}(M)) \Rightarrow \\ \qquad \mathtt{union}\ (ES, \mathit{state}(N), (\mathtt{F}\ Z')) \end{array}$$

The idea behind these definitions is that the rewrite axiom labelled $\mathtt{R}ij1$ generated by rule_1 simulates the effect of instantiating role i immediated followed by the execution of one of its roles, which is j in this case. As a consequence it generates the formal multiset of tokens $(\mathtt{T}i1\ P\ \boldsymbol{L})\ \ldots\ (\mathtt{T}in\ P\ \boldsymbol{L})$ with $(\mathtt{T}ij\ P\ \boldsymbol{L})$ removed, because the corresponding rule has already been executed. The rewrite axiom labelled $\mathtt{R}ij2$ generated by rule_2 takes care of the execution of remaining rules, and hence remains unchanged compared with the previous section.

Now we obtain a result entirely analogous to the previous theorem:

Theorem 5.8 (Soundness and Completeness) Let $(\mathcal{D}, \mathcal{P})$ be an MSR specification, let S, S' be MSR states, and let R, R' be MSR active roles sets. Then for all n, k we have the following equivalence:
There are MSR contexts Σ^n, Σ'^{n+k} s.t. $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{(I)E} [S']^{R'}_{\Sigma'}$ iff
there exists P, ES s.t. $\mathit{context}(\mathcal{D}),\ \mathit{roles}(\mathcal{P}) \vdash$
$!!\ [P] : (\mathtt{union}\ (ES, (\mathtt{F}\ (\mathtt{s}^n\mathtt{0})), \mathit{state}(S), \mathit{actroles}(R))) \Rightarrow$
$\qquad (\mathtt{union}\ (ES, (\mathtt{F}\ (\mathtt{s}^{n+k}\ \mathtt{0})), \mathit{state}(S'), \mathit{actroles}(R'))),$
where we identify terms that are structurally equal in $\mathit{context}(\mathcal{D})$, and ES is a formal multiset containing $(\mathtt{E}\ U\ Q)$ only for terms Q of type U.

Proof Sketch. The only difference w.r.t. the previous theorem is that we use the new judgment $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{(I)E} [S'']^{R''}_{\Sigma''}$ as the operational semantics of MSR. By definition there are two cases to consider:

(i) There is a role i with a rule j such that

$$\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{I_i^{A,N}} [S']^{R'}_{\Sigma'} \text{ and } \mathcal{D}, \mathcal{P} \vdash [S']^{R'}_{\Sigma'} \longrightarrow_{E_{i,j}^{A,N}} [S'']^{R''}_{\Sigma''}.$$

Observe that we are concerned with a sequential composition of the judgments that we represented in our previous representation (Section 5.4). Omitting quantifiers and conditions for clarity, the first judgment was represented by the rewrite axiom

$$\begin{array}{l} [\mathtt{A}i] : (\mathtt{union}((\mathtt{E\ princ}\ P), (\mathtt{F}\ Z)) \Rightarrow \\ \qquad (\mathtt{union}((\mathtt{E\ princ}\ P), (\mathtt{T}i1\ P\ \boldsymbol{L}), \ldots, (\mathtt{T}in\ P\ \boldsymbol{L}), (\mathtt{F}\ Z'))), \end{array}$$

and the second judgment was represented by the rewrite axiom

$$\begin{array}{l} [\mathtt{A}ij] : (\mathtt{union}\ (ES, (\mathtt{T}ij\ P\ \boldsymbol{L}), (\mathtt{F}\ Z'), \mathit{state}(M))) \Rightarrow \\ \qquad (\mathtt{union}\ (ES, \mathit{state}(N), (\mathtt{F}\ Z''))). \end{array}$$

Consequently, our new representation (see definition of rule_1) uses the sequential composition of these two:

$[\mathtt{A}ij1] : (\mathtt{union}\ ((\mathtt{E\ princ}\ P), (\mathtt{F}\ Z), ES, \mathit{state}(M))) \Rightarrow$
$\qquad (\mathtt{union}\ ((\mathtt{E\ princ}\ P), TS, ES, \mathit{state}(N), TS, (\mathtt{F}\ Z''))),$

where TS is the formal multiset containing $(\mathtt{T}i1\ P\ \boldsymbol{L})\ \ldots\ (\mathtt{T}in\ P\ \boldsymbol{L})$ with $(\mathtt{T}ij\ P\ \boldsymbol{L})$ removed.

(ii) There is a role i with a rule j such that $\mathcal{D}, \mathcal{P} \vdash [S]^R_\Sigma \longrightarrow_{E^{A,\boldsymbol{N}}_{i,j}} [S'']^{R''}_{\Sigma''}$.

For this case, we just need to simulate the execution of a rule. Hence, the rewrite axiom (see definition of rule_2) is as in the previous representation:

$[\mathtt{A}ij2] : \mathtt{union}\ (ES, (\mathtt{T}ij\ P\ \boldsymbol{L}), (\mathtt{F}\ Z), \mathit{state}(M)) \Rightarrow$
$\qquad \mathtt{union}\ (ES, \mathit{state}(N), (\mathtt{F}\ Z')).$

□

As an optimization, the rewrite axiom $\mathtt{R}ij1$ can be omitted if M contains any of the variables in $\boldsymbol{L}$. The reason is that we have the invariant (for the reachable states we are concerned with in the theorem) that the variables $\boldsymbol{L}$ are instantiated by objects which do not exist in the state, so that this axiom could never be applied.

Another obvious optimization is to omit the rewrite axiom $\mathtt{R}ij$ when the role i contains only a single rule. This optimization relies on the fact that in this case $\mathtt{T}ij$ can never appear in the state, an invariant that holds for the reachable states we are concerned with in the theorem. More generally, we drop any rule that depends on a $\mathtt{T}ij$ that is never generated. This can happen, because the only rule that generates $\mathtt{T}ij$ has been eliminated by previous optimizations.

6 Translation in our Example

Returning to the Otway-Rees protocol and its specification in MSR, we now illustrate how its responder role (role number 2) is translated into RWLDT. For brevity, we omit all declarations, except the ones for the network predicate, message concatenation (denoted $\_,\_$ in MSR), and the encryption function, (which was written as $\{\_\}_\_$ in MSR):

```
N : msg -> state
append : msg -> msg -> msg
encrypt : {A,B : princ} msg -> (shK A B) -> msg
```

As for `union`, we write $(\mathtt{append}\ (M_1, \ldots, M_n))$ to abbreviate $(\mathtt{append}\ M_1\ (\mathtt{append}\ (M_2, \ldots, M_n)))$.

We have the following coercions (to which we have given longer names here):

```
nonce-msg : nonce -> msg
```

```
princ-msg : princ -> msg
ltK-shK :  {A,B : princ} (ltK A B) -> (shK A B)
stK-shK :  {A,B : princ} (stK A B) -> (shK A B)
```

We also declare the following injections as required by the translation to generate fresh symbols of the target type:

```
NONCE : nat -> nonce
L2 : nat -> ({B : princ} princ ->
              nonce -> nonce -> (ltK B S) -> state)
```

Finally, we declare token constructors relevant for the responder role:

```
T21 : princ -> ({B : princ} princ ->
                 nonce -> nonce -> (ltK B S) -> state) -> state .
T22 : princ -> ({B : princ} princ ->
                 nonce -> nonce -> (ltK B S) -> state) -> state .
```

The translation of the responder role produces four rewrite rules, but two of them can be eliminated by our optimizations:

```
R211 : !! {B:princ}
  {L:{B:princ} princ -> nonce -> nonce -> (ltK B S) -> state}
  {A:princ}{kBS:(ltK B S)}{X:msg}
  {fresh,fresh':nat}{n,nB:nonce}

  (L := (L2 fresh)) ->
  (nB := (NONCE (suc fresh))) ->
  (fresh' := (suc fresh)) ->

  [A211] : (union ((E (ltK B S) kBS), (F fresh),
                   (N (append ((nonce-msg n),
                               (princ-msg A),
                               (princ-msg B),
                               X)))))
        => (union ((E (ltK B S) kBS), (F fresh'),
                   (N (append (n,A,B,X,
                      (encrypt (append ((nonce-msg nB),
                                        (nonce-msg n),
                                        (princ-msg A),
                                        (princ-msg B)))
                               (ltK-shK B S kBS))))),
                   (L B A n nB kBS),
                   (T22 B L)))

R222 : !! {B:princ}
  {L:{B:princ} princ -> nonce -> nonce -> (ltK B S) -> state}
  {A:princ}{kBS:(ltK B S)}{kAB:(stK A B)}{Y:msg}{n,nB:nonce}

  [A222] : (union ((N (append ((nonce-msg n),Y,
                       (encrypt (append ((nonce-msg nB),
                                         (stK-msg A B kAB)))
                                (ltK-shK B S kBS))))),
                  (L B A n nB kBS),
                  (T22 B L)))
        => (N (append ((nonce-msg n),Y)))
```

The justification for eliminating R221 is that it contains (L B A n nB kBS) with L fresh on its left-hand side. Since R212 depends on (T21 B L), which could only be generated by R221, this rule can be dropped as well.

The full RWLDT specification successfully passes the OCC type checker, which implies that the original MSR specification is type-correct as well. The OCC prototype can further be used to explore the dynamics of the protocol. For example, to restrict the protocol execution to one instance of each role and to observe termination we add STARTi and TERMINATEDi tokens to respectively the first and last rules of each role i.

```
A:princ . B:princ . kAS:(ltK A S) . kBS:(ltK B S) .

rew (union ((F 0),(E P A),(E P B),
       (E (ltK A S) kAS),(E (ltK B S) kBS),
       (START1 A), (START2 B), (START3 S))) .
trace:
    A111 A211 A311 A222 A122
result:
    (union ((F 6),(E P A),(E P B),
       (E (ltK A S) kAS),(E (ltK B S) kBS),
       (TERMINATED1 A), (TERMINATED2 B), (TERMINATED3 S)))
```

After starting the symbolic execution the system performs a series of actions each corresponding to the application of a rule. Finally, the terminating state is reached, the explicit type information is preserved, and six fresh constants have been used. An exploration of the state space using Maude shows that the above execution is the only possible one from the given initial state.

7 Final Remarks

In this paper we have presented a shallow and hence efficient embedding from MSR into rewriting logic with dependent types (RWLDT), which has been introduced as a restricted instance of the open calculus of constructions (OCC). This mapping forms the basis for an ongoing implementation of an MSR execution and analysis environment. A mapping from RWLDT into RWL has already been implemented as part of the OCC prototype in Maude. This enabled us to perform symbolic execution of the translated MSR specification in our example. The user interaction takes place at the level of RWLDT terms, which directly correspond to MSR terms, and hence the user does not need to be concerned with the resulting translation into RWL. A similar interface for symbolic search and model checking would be easy to implement. At the moment, we can however already export the RWL translation of the RWLDT specification and perform symbolic search and model checking at the level of

Maude.

For the sake of clarity we made a number of simplifying assumptions in this paper. We decoupled the issue of inferring implicit parts of an MSR specification from the actual translation phase, which is exactly the way we would like to organize the architecture of the translator. We also assumed the absence of name clashes, an assumption that is not necessary if we use the CINNI explicit substitution calculus [13,14] and its term representation. In fact the theory and prototype for OCC are already based on this calculus.

An additional feature of MSR that may require changes to our representation are constraints, i.e., conditions attached to MSR rules. Constraints do not appear in [4,5], but have proved useful, for instance, in the Kerberos analysis [3]. Among the options are the direct translation into conditional rules of RWLDT, the extension of the linear state by a non-linear counterpart (as in standard sequent presentations of linear logic) and its use to verify the constraints, or a combination of these two possibilities. Equations are a recent addition to MSR, inspired by this collaboration. They can be directly mapped to the computational equations of RWLDT. Further extensions of MSR, such as moving to richer executable fragments of linear logic in the style of CLF [16], a direction currently investigated by the first author, seem to require deeper embedding of MSR into RWLDT, an interesting topic that we leave for future research.

An important part of MSR, the data access specification [6], has not been treated in this paper, because a sufficiently generic and concise formulation is still subject of ongoing work. Our most recent idea to formalize the data access specification is to use predicates inside the type theory to express the accessibility of information relative to principals. In combination with the assertional propositions of RWLDT this may simplify the representation of data access rules significantly and would provide a great deal of flexibility. Furthermore, the proof that the data access specification is satisfied would become an object inside the type theory. In fact, the logical nature of RWLDT is far from being fully exploited so far, which leads us to the last point of the conclusion.

In addition to the automatic symbolic analysis techniques mentioned above, our two-level architecture opens the door to performing formal reasoning at the level of RWLDT, which contains all the type information of the original MSR specification and uses practically the same term syntax. Indeed, interactive theorem proving is supported by OCC [14], but to make use of it our translation needs to be enriched to make explicit the inductive nature of MSR, which can be achieved essentially by adding suitable elimination/induction principles. Formal reasoning would ultimately rely on the model-theoretic semantics of OCC, but it can use its operational semantics to enhance the expressivity of types and to provide partial automation in proofs.

See http://formal.cs.uiuc.edu/stehr/msr.html for the complete specification of the Otway-Rees example, other examples, and recent progress on the project.

Acknowledgment. The research reported has been conducted in the scope of the CONTESSA project http://formal.cs.uiuc.edu/contessa. The first author is partially supported by NRL under contract N00173-00-C-2086. The second author is supported by ONR Grant N00014-02-1-0715.

References

[1] Barendregt, H. P., *Lambda-calculi with types*, in: S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, editors, *Background: Computational Structures*, Handbook of Logic in Computer Science **2**, Claredon Press, Oxford, 1992 .

[2] Bouhoula, A., J.-P. Jouannaud and J. Meseguer, *Specification and proof in membership equational logic*, Theoretical Computer Science **236** (2000), pp. 35–132.

[3] Butler, F., I. Cervesato, A. D. Jaggard and A. Scedrov, *A Formal Analysis of Some Properties of Kerberos 5 Using MSR*, in: *Fifteenth Computer Security Foundations Workshop* (2002), pp. 175–190.

[4] Cervesato, I., *A specification language for crypto-protocols based on multiset rewriting, dependent types and subsorting.*, in: *Workshop on Specification, Analysis and Validation for Emerging Technologies*, 2001, pp. 1–22.

[5] Cervesato, I., *Typed MSR: Syntax and examples*, in: *1st International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security* (2001), pp. 159–177.

[6] Cervesato, I., *Data Access Specification and the Most Powerful Symbolic Attacker in MSR*, in: *Software Security, Theories and Systems* (2003), pp. 384–416.

[7] Cervesato, I., *MSR: Language definition and programming environment* (2003), draft available from http://theory.stanford.edu/~iliano/MSR/.

[8] Cervesato, I., N. Durgin, P. D. Lincoln, J. C. Mitchell and A. Scedrov, *A Meta-Notation for Protocol Analysis*, in: *12th Computer Security Foundations Workshop* (1999), pp. 55–69.

[9] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and J. F. Quesada, *Maude: Specification and programming in rewriting logic*, Theoretical Computer Science (2001).

[10] Denker, G. and J. K. Millen, *CAPSL Intermediate Language*, in: N. Heintze and E. Clarke, editors, *Proceedings of the Workshop on Formal Methods and Security Protocols — FMSP*, Trento, Italy, 1999.

[11] Meseguer, J., *Conditional rewriting logic as a unified model of concurrency*, Theoretical Computer Science **96** (1992), pp. 73–155.

[12] Otway, D. and O. Rees, *Efficient and timely mutual authentication*, Operating Systems Rewiew **21** (1987), pp. 8–10.

[13] Stehr, M.-O., *CINNI – A Generic Calculus of Explicit Substitutions and its Application to λ-, σ- and π-calculi*, in: K. Futatsugi, editor, *3rd International Workshop on Rewriting Logic and its Applications*, ENTCS **36** (2000), pp. 71 – 92, http://www.elsevier.nl/locate/entcs/volume36.html.

[14] Stehr, M.-O., *Programming, Specification, and Interactive Theorem Proving — Towards a Unified Language based on Equational Logic, Rewriting Logic, and Type Theory*, Doctoral Thesis, Universität Hamburg, Fachbereich Informatik, Germany (2002), http://www.sub.uni-hamburg.de/disse/810/.

[15] Stehr, M.-O. and J. Meseguer, *Pure type systems in rewriting logic*, in: *From Object-Orientation to Formal Methods: Dedicated to The Memory of Ole-Johan Dahl*, LNCS **2635** (2004).

[16] Watkins, K., I. Cervesato, F. Pfenning and D. Walker, *A Concurrent Logical Framework I: Judgments and Properties*, Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University (2003).