Algebraic Abstractions ⋆

Michel Bidoit and Alexandre Boisseau

Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan, France
{bidoit,boisseau}@lsv.ens-cachan.fr

Abstract. In this paper we study abstraction techniques for verification problems of the form $\mathcal{C} \models \phi$, where $\mathcal{C}$ is a first-order structure and ϕ is a first-order formula (both w.r.t. a given signature Σ). This study is motivated by the need of such abstractions for the automatic verification of properties of cryptographic protocols, which in our approach are modeled by first-order structures. Our so-called algebraic abstractions will be correct by construction and optimal in some certain technical sense. Moreover, we provide guidelines to design specific algebraic abstractions suited for verification problems corresponding to cryptographic protocols.

1 Introduction and Motivations

Formal verification of cryptographic protocols is nowadays an important issue due to their crucial rôle in the design of security or electronic commerce systems. A cryptographic protocol is usually described by a sequence of steps, where each step describes a message exchange between two principals.[1] Let us consider for instance the following simplified cryptographic protocol (see e.g. [4]):

$$\begin{array}{l} 1)\ A \to S : \langle A, B\rangle \\ 2)\ S \to A : \{\langle K_B, B\rangle\}_{K_S^{-1}} \end{array}$$

This protocol would be used in a public key cryptography system, where each participant has a private key (which should remain secret) and a public key (which can be provided to the other participants). To start a private session with B, the participant A should first obtain, from a server S, the public key of B. Hence A first sends a corresponding request to S (step 1 above). When S receives such a request, S computes his answer by encrypting the pair ⟨public key of B, B's name⟩ with his private key K_S^{-1} and sends it to A (step 2 above).

⋆ This work is partially supported by the RNTL project EVA funded by the French Ministry of Research.

[1] A principal corresponds to a specific rôle in the cryptographic system. For instance, in the following example, there are two principals, one for the key server, one for the participants requesting a key. There may be several participants interacting according to the same rôle, they are all instances of the same principal. In informal explanations it is very frequent not to distinguish between participants and principals.

The aim of this encryption step is not to ensure the secrecy of the message, since anybody can decrypt it using the public key of S, but to ensure authentication: S is the unique participant able to encrypt a message with K_S^{-1}.

Verification of such a cryptographic protocol amounts to prove that some properties about all its possible executions hold, for example:

– at the end of each execution, A receives the key he asked for and no other,[2]
– during each execution, K_S^{-1} remains secret.

The first property reflects in a certain sense the aim of the protocol. The second one ensures that the private key of S (which plays an important rôle in the authentication process) will not be compromised.

More precisely, one generally assumes that both the encryption algorithms used in the protocol as well as the implementation of the protocol itself are correct, but that the communication media are non-trustable, i.e., one assumes the presence of an intruder able to eavesdrop messages, divert them, produce new messages from the already exchanged messages, and even impersonate some other participants. Indeed, following a pessimistic approach, we assume that each message exchange is intercepted by an intruder I and possibly modified. This means that the first step of our simplified protocol will be split into two sub-steps:

$$\begin{array}{l} 1A)\ A \to I\ : m = \langle A, B\rangle \\ 1S)\ I\ \to S : m' \end{array}$$

where m' is a fake message (possibly equal to m) produced by the intruder according to his *knowledge* about the previously exchanged (and intercepted) messages (and the protocol). The second step of the protocol is split in a similar way:

$$\begin{array}{l} 2S)\ S \to I\ : m = \{\langle K_B, B\rangle\}_{K_S^{-1}} \\ 2A)\ I\ \to A : m' \end{array}$$

Model-checking techniques (see, e.g., [3]) cannot be directly used for the verification of cryptographic protocols since these should necessarily be modeled by infinite state systems:

– the number of participants, keys, nonces[3] and more generally atomic data is unbounded,
– moreover, messages can be built from atomic data using encryption and pairing in an arbitrary way, so the set of the potential messages exchanged is also infinite (and even if the number of participants, keys, nonces, ..., involved in the protocol is finite, the size of the exchanged messages is unbounded, at least for those possibly produced by the intruder),
– finally, several sessions of the protocol can be run (concurrently or sequentially, depending on the modeling).

[2] It is easy to see that this property may not hold for our simplified protocol in presence of malicious intruders.

[3] Nonces are random numbers incorporated into messages in order to avoid replay attacks.

A standard approach to overcome this problem is the use of abstraction techniques (see [6, 9, 8]) in order to reduce the infinite system modeling the cryptographic protocol into a finite (or into a still infinite, but simpler) one. The use of abstraction techniques is summarized by the following diagram:

$$\begin{array}{ccc} & \textit{Abstraction} & \\ M & \longrightarrow & M^a \\ & & \\ \overset{?}{\models} & & \overset{?}{\models} \\ & & \\ \phi & \longrightarrow & \phi^a \end{array}$$

Here, given a model M and a property ϕ, we want to know whether this property holds for M. Instead of working directly with M and ϕ, we use a function called *abstraction* that transforms M and ϕ into an abstract model M^a and an abstract property ϕ^a. Now, if the abstraction is *correct*, i.e., if we have:

$$M^a \models \phi^a \quad \Rightarrow \quad M \models \phi$$

then $M \models \phi$ can be deduced from $M^a \models \phi^a$. Using such an abstraction will solve our problem when $M^a \models \phi^a$ is decidable (in particular when the abstract model is finite). The abstraction will be called *adequate* if $M^a \models \phi^a$ holds. In general abstractions are introduced in a rather ad hoc way, that is, for each verification problem $M \models \phi$, one has to invent some (hopefully adequate) abstraction and to prove its correctness before using it.

The aim of this paper is to introduce a general setting where classes of abstractions, correct by construction, can be more easily defined. More precisely, we will define abstraction schemes and correctness criteria that will ensure that any abstraction defined according to these schemes and satisfying the corresponding criteria is correct. Then the adequacy problem will be made simpler by using the fact that we are not looking for arbitrary abstractions, but merely for abstractions suitable for the verification of cryptographic protocols. Our specific way of modeling these protocols will provide useful guidelines for choosing a relevant abstraction.

The rest of this paper is organized as follows. In Section 2 we sketch our modeling technique of cryptographic protocols by first-order structures. In Section 3 we detail abstraction schemes for the verification of first-order formula with respect to first-order structures, and we provide the corresponding correctness criteria. Then we prove in Section 4 that abstractions defined according to our method are (correct by construction and) optimal in a certain technical sense. In Section 5 we show how to apply our abstraction method in the context of cryptographic protocols by exhibiting some relevant special cases.

In the sequel we assume familiarity with the basic concepts of algebraic specification (see, e.g., [2]) and in particular with the usual institution of (many-sorted) first-order logic with equality **FOLeq**:

- An algebraic signature is a pair $\Theta = (S, \Omega)$, where S is a set of sorts and Ω is a set of function symbols (together with their arity). A model for such a signature is a Θ-algebra and the class of Θ-algebras is denoted by $\mathbf{Alg}(\Theta)$.
- A first-order signature is a triple $\Sigma = (S, \Omega, \Pi)$, where (S, Ω) is an algebraic signature called the algebraic signature underlying Σ and denoted by $[\Sigma]$, and Π is a set of predicate symbols (with their arity). First-order signatures form a category denoted by **Sign**. A model for a first-order signature $\Sigma = (S, \Omega, \Pi)$ is a first-order structure $\mathcal{C}$ which is made of a $[\Sigma]$-algebra called the $[\Sigma]$-algebra underlying $\mathcal{C}$ and denoted by $[\mathcal{C}]$ together with an interpretation for each predicate symbol in Π. $\mathbf{Str}(\Sigma)$ denotes the class of all Σ-structures.
- The set of first-order formulas (with equality) built from Σ and a set of variables X is denoted by $\mathbf{Sen}(\Sigma)$.
- $\models$ denotes the usual satisfaction relation between Σ-structures and Σ-formulas.

In the following, by default "signature" means "first-order signature".

2 Modeling Cryptographic Protocols by First-Order Structures

As any protocol, a cryptographic protocol is naturally modeled by a labeled transition system. However, in the specific context of cryptographic protocols, the content and structure of the messages exchanged is especially important, since, as explained in the previous section, to model the *knowledge of the intruder*, we will use the set of messages exchanged (and therefore intercepted by the intruder). More precisely, assuming that the intruder "knows" a set of messages s, he will be able to produce a (possibly fake) message m if the ingredients necessary to build m can be "derived" from s by the intruder. This is formally captured by a predicate $\Vdash$ which states under which conditions a message m can be produced by the intruder from his current knowledge represented by a set of messages s. Following [11], this predicate $\Vdash$ is defined inductively by the following rules:

$$\frac{m \in s}{s \Vdash m} \quad (\xi)$$

$$\frac{s \Vdash m_1 \qquad s \Vdash m_2}{s \Vdash \langle m_1, m_2 \rangle} \quad (\pi)$$

$$\frac{s \Vdash \langle m_1, m_2 \rangle}{s \Vdash m_1} \quad (\pi'_1) \qquad \frac{s \Vdash \langle m_1, m_2 \rangle}{s \Vdash m_2} \quad (\pi'_2)$$

$$\frac{s \Vdash m \quad s \Vdash k}{s \Vdash \{m\}_k} \quad (\gamma) \qquad \frac{s \Vdash \{m\}_k \quad s \Vdash k^{-1}}{s \Vdash m} \quad (\gamma')$$

The first four rules are obvious whereas (γ) and (γ') express a strong hypothesis about the encryption algorithms used in the protocol. For instance, (γ') states that if the intruder wants to produce m from $\{m\}_k$, he must know the inverse of k (hence these rules reflect our assumption that the encryption algorithms are perfect, as generally done in the literature [4, 13, 10]).

Hence, a first part of the model is a first-order structure corresponding to:

- atomic data (keys, nonces, principals, etc.), together with some operations (e.g., obtaining the inverse of a given key),
- messages built from atomic data using two constructions:
  - the pair construction, denoted $\langle m_1, m_2 \rangle$, to put together messages m_1 and m_2,
  - the encrypt construction, denoted $\{m\}_k$, to encrypt the message m using the key k,
- sets of messages with the usual operations and membership predicate,
- the predicate $\Vdash$.

Then the labeled transition system itself is also described as a first-order structure (enriching the above one), where the transition relation between states is modeled by a predicate $q \xrightarrow{(l,m)} q'$ (see e.g. [1]).

To summarize, the whole cryptographic protocol is modeled by a first-order structure, which is itself described by a specification structured as follows:

$$
\begin{tikzcd}
& \boxed{\text{Protocol (LTS)}} \arrow[d, dash] & \\
& \text{Intruder} \arrow[d, dash] & \\
& \text{Sets (of messages)} \arrow[d, dash] & \\
& \text{Messages} \arrow[d, dash] \arrow[dr, dash] & \\
\boxed{\text{Principals}} \arrow[r, dash] & \boxed{\text{Keys}} & \boxed{\text{Nonces}}
\end{tikzcd}
$$

In the above diagram, the boxed pieces of specification correspond to the parts that are specific to a given cryptographic protocol, while the other parts are independent of the protocol considered. The specification of the intruder is merely the axiomatization of the predicate $\Vdash$ according to the rules given above. For the specification of the labeled transition system, one has to keep in mind that, as explained in the previous section, each step of the protocol is split into two sub-steps. So the predicate $q \xrightarrow{(l,m)} q'$ is axiomatized by a formula $q \xrightarrow{(l,m)} q' \Leftrightarrow r$, where r is a disjunction of formulas, one for each sub-step of the protocol (the full description of the transition relation corresponding to our toy cryptographic protocol is given in the following example).

Example 1 (A CASL *specification of the toy cryptographic protocol).*

spec PRINCIPAL =
  sort *Princ*
  op $S : Princ$
end

spec KEY =
  PRINCIPAL
then **free type** $Key ::= k(Princ) \mid k^{-1}(Princ)$
  op $\_\_^{-1} : Key \to Key;$
  $\forall x : Princ$
  • $k(x)^{-1} = k^{-1}(x)$
  • $k^{-1}(x)^{-1} = k(x)$
end

spec MESSAGE =
  PRINCIPAL **and** KEY
then **free type** $Msg ::= \mathit{sort}\ Princ \mid \mathit{sort}\ Key$ %% Coercion of atomic data
  $\mid \langle \_\_, \_\_ \rangle(Msg;\ Msg)$ %% Pairs
  $\mid \{\_\_\}_{\_\_}(Msg;\ Key)$ %% Encryption
end

spec INTRUDER =
  MESSAGE **and** SET [**sort** *Msg*]
then **free** {
  pred $\_\_ \Vdash \_\_ : Set[Msg] \times Msg$
  $\forall m, m_1, m_2 : Msg;\ k : Key;\ s : Set[Msg]$
  • $m \in s \Rightarrow s \Vdash m$
  • $(s \Vdash m_1 \wedge s \Vdash m_2) \Rightarrow s \Vdash \langle m_1, m_2 \rangle$
  • $s \Vdash \langle m_1, m_2 \rangle \Rightarrow s \Vdash m_1$
  • $s \Vdash \langle m_1, m_2 \rangle \Rightarrow s \Vdash m_2$
  • $(s \Vdash \{m\}_k \wedge s \Vdash k^{-1}) \Rightarrow s \Vdash m$
  • $(s \Vdash m \wedge s \Vdash k) \Rightarrow s \Vdash \{m\}_k$ }
end

spec TOY_CRYPTOGRAPHIC_PROTOCOL =
  INTRUDER
then **free types**
  $Lbl ::= 1A \mid 1S \mid 2S \mid 2A;$
  %% As explained before, the two steps are split into four sub-steps
  $State ::= (\_\_.Adir : Set[Msg];$
  $\_\_.Sid : Princ;$
  $\_\_.I : Set[Msg])$

%% Each state is a tuple containing :
%% The directory where A records the public keys
%% The identity of the participant received by the server S
%% The set of messages known by the intruder

pred $\_\_ \xrightarrow{(\_\_,\_\_)} \_\_ : \mathit{State} \times \mathit{Lbl} \times \mathit{Msg} \times \mathit{State}$

$\forall q, q' : \mathit{State};\ m : \mathit{Msg};\ l : \mathit{Lbl}$

$$
\begin{aligned}
\bullet\ & q \xrightarrow{(l,m)} q' \ \Leftrightarrow \\
&\quad (l = \mathit{1A} \\
&\qquad \wedge\ \exists x : \mathit{Princ}\ \bullet\ \exists y : \mathit{Princ}\ \bullet \\
&\qquad\quad m\ =\ \langle x, y\rangle\ \wedge\ q'\ =\ (q.\mathit{Adir}, q.\mathit{Sid}, q.I \cup m)) \\
&\quad \vee\ (l = \mathit{1S} \\
&\qquad \wedge\ \exists x : \mathit{Princ}\ \bullet\ \exists y : \mathit{Princ}\ \bullet \\
&\qquad\quad m\ =\ \langle x, y\rangle\ \wedge\ q.I \Vdash m\ \wedge\ q'\ =\ (q.\mathit{Adir}, y, q.I)) \\
&\quad \vee\ (l\ =\ \mathit{2S} \\
&\qquad \wedge\ m\ =\ \{\langle k(q.\mathit{Sid}), q.\mathit{Sid}\rangle\}_{k^{-1}(S)} \\
&\qquad \wedge\ q'\ =\ (q.\mathit{Adir}, q.\mathit{Sid}, q.I \cup m)) \\
&\quad \vee\ (l\ =\ \mathit{2A} \\
&\qquad \wedge\ (\exists x : \mathit{Princ}\ \bullet\ \exists y : \mathit{Key}\ \bullet \\
&\qquad\quad m\ =\ \{\langle y, x\rangle\}_{k^{-1}(S)}\ \wedge\ q.I \Vdash m \\
&\qquad\quad \wedge\ q'\ =\ (q.\mathit{Adir} \cup \langle x, y\rangle, q.\mathit{Sid}, q.I))
\end{aligned}
$$

then **free** **{**

pred $\mathit{Reach} : \mathit{State} \times \mathit{State}$

$\forall q, q', q'' : \mathit{State};\ m : \mathit{Msg};\ l : \mathit{Lbl}$

- $\mathit{Reach}(q, q)$
- $\mathit{Reach}(q, q') \wedge q \xrightarrow{(l,m)} q'\ \Rightarrow\ \mathit{Reach}(q, q'')$ **}**

end

Now, we can formalize the properties of our toy protocol.

– The following sentence states that the server's private key remains secret:

$$\forall q_0, q : \mathit{State}\ \bullet\ \neg(q_0.I \Vdash k^{-1}(S)) \wedge \mathit{Reach}(q_0, q)\ \Rightarrow\ \neg(q \Vdash k^{-1}(S))$$

– A's directory contains only valid pairs $\langle$name,key$\rangle$:

$$
\begin{aligned}
&\forall q_0, q : \mathit{State}\ \bullet\ \forall b : \mathit{Princ}\ \bullet\ \forall k : \mathit{Key}\ \bullet \\
&\quad q_0.\mathit{Adir} = \mathit{empty} \wedge \mathit{Reach}(q_0, q) \wedge \langle b, k\rangle \in q.\mathit{Adir}\ \Rightarrow\ k = k(b)
\end{aligned}
$$

3 Algebraic Abstractions

In this section, we describe our general setting for defining abstractions suitable for verification problems of the form $\mathcal{C} \models \phi$ (see Section 1), where $\mathcal{C}$ is a first-order structure and ϕ is a first-order formula (both w.r.t. a given signature Σ).[4] The abstraction problem can be split into the following subtasks:

[4] Abstractions are usually described by means of *Galois connections* (see [8]). This can indeed be done as well for our algebraic abstractions. However, in our case, a direct presentation without explicit Galois connections is much more easily to follow, and this is why we will adopt the later style here.

1. Abstract $\mathcal{C}$ into $\mathcal{A}$.
2. Abstract ϕ into ϕ^a.
3. Check the correctness of this abstraction.

Let us focus on the first subtask. A crucial remark is that any Σ-structure $\mathcal{C}$ can be considered as a $[\Sigma]$-algebra $[\mathcal{C}]$ together with an interpretation for each predicate symbol in Σ. Thus, abstracting $\mathcal{C}$ can itself be decomposed into, first, abstracting the underlying $[\Sigma]$-algebra $[\mathcal{C}]$, and then, finding an adequate abstraction for each predicate symbol interpretation.

To abstract algebras, we use, as usual, homomorphisms. Let $\alpha : [\mathcal{C}] \to \mathcal{B}$ be a $[\Sigma]$-morphism between the given $[\Sigma]$-algebra $[\mathcal{C}]$ and some $[\Sigma]$-algebra $\mathcal{B}$. To understand how to extend $\mathcal{B}$ into a Σ-structure $\mathcal{A}$ abstracting $\mathcal{C}$, i.e., to understand how to provide relevant interpretations for the predicate symbols, let us consider the following illustrative example.

Example 2. Consider the following CASL specification [7], specifying natural numbers and lists of natural numbers in the usual way:

spec BASICNATLIST =
  types $Nat ::= 0 \mid suc(Nat);$
    $List ::= empty \mid cons(Nat;\ List)$
  pred $\_\_\ is\_in\ \_\_ : Nat \times List$
  $\forall x, y : Nat;\ l : List$
  • $\neg(x\ is\_in\ empty)$
  • $x\ is\_in\ cons(y, l) \Leftrightarrow (x = y \vee x\ is\_in\ l)$
end
spec NATLIST = **free {** BASICNATLIST **}**

The specification NATLIST has (up to isomorphism) only one model $\mathcal{C}$ defined by:

- $\mathcal{C}_{Nat} \stackrel{\text{def}}{=} \mathbb{N}$,
- $\mathcal{C}_{List} \stackrel{\text{def}}{=} \mathbb{N}^*$ (set of finite sequences of elements of $\mathbb{N}$),
- 0, *suc*, *empty*, and *cons* are interpreted in the usual way.

Now consider the specification NATLIST' given by:

spec NATLIST' =
free **{** BASICNATLIST **then**
    $\forall x, y : Nat;\ l : List$
    • $suc(suc(x)) = suc(x)$
    • $cons(x, cons(y, l)) = cons(y, cons(x, l))$
    • $cons(x, cons(x, l)) = cons(x, l)$ **}**

This specification also has a unique model $\mathcal{A}$ up to isomorphism, given by:

- $\mathcal{A}_{Nat} \stackrel{\text{def}}{=} \{\texttt{nul}, \texttt{pos}\}$,
- $0^{\mathcal{A}} \stackrel{\text{def}}{=} \texttt{nul}$, $suc^{\mathcal{A}}(\texttt{nul}) \stackrel{\text{def}}{=} \texttt{pos}$, and $suc^{\mathcal{A}}(\texttt{pos}) \stackrel{\text{def}}{=} \texttt{pos}$,
- $\mathcal{A}_{List}$ is the powerset of $\mathcal{A}_{Nat}$ and $is\_in^{\mathcal{A}}$ is the usual membership predicate.

Then, there is a unique homomorphism α from $[\mathcal{C}]$ to $\mathcal{B} = [\mathcal{A}]$. Given n in $\mathcal{C}_{Nat}$ and l in $\mathcal{C}_{List}$, we have $n\ is\_in^{\mathcal{C}}\ l$ implies $\alpha(n)\ is\_in^{\mathcal{A}}\ \alpha(l)$, because α is an homomorphism, so in order to prove that n is not in l, it is sufficient to prove that $\alpha(n)$ is not in $\alpha(l)$. However, $n\ is\_in^{\mathcal{C}}\ l$ cannot be deduced from $\alpha(n)\ is\_in^{\mathcal{A}}\ \alpha(l)$ (consider for instance $n = 2$ and $l = cons(1, empty)$). A stronger property, for instance $\alpha(n)\ is\_in^{\mathcal{A}}\ \alpha(l)$ and $\alpha(n) = \texttt{nul}$, is required.

The above example illustrates that in general, two (distinct) interpretations in the abstract domain should be associated with each interpretation of a predicate symbol in $\mathcal{C}$, one for validation and one for refutation. Hence, the idea is to replace each predicate symbol P interpreted in the concrete domain by two predicate symbols $P_{\oplus}$ and $P_{\ominus}$, which means using a new signature Σ^* for the abstract structure.

Abstraction of a formula ϕ can then be done in a straightforward way: each positive (resp. negative) occurrence of a predicate symbol P in ϕ is replaced by $P_{\oplus}$ (resp. by $P_{\ominus}$). We can even construct a dual formula obtained by replacing the positive (resp. negative) occurrences of P in ϕ by $P_{\ominus}$ (resp. by $P_{\oplus}$). If these formulas are denoted $\phi_{\oplus}$ and $\phi_{\ominus}$ and if $\mathcal{B}$ is "correctly" embedded into a Σ^*-structure $\mathcal{A}$ (i.e., if some conditions on the interpretations of $P_{\oplus}$ and of $P_{\ominus}$ in $\mathcal{A}$ w.r.t. both the interpretation of P in $\mathcal{C}$ and the morphism α hold), then we will obtain the following properties:

$$\begin{array}{rl} & \mathcal{A} \models \phi_{\oplus} \;\Rightarrow\; \mathcal{C} \models \phi \\ \text{and} & \mathcal{A} \not\models \phi_{\ominus} \;\Rightarrow\; \mathcal{C} \not\models \phi \end{array}$$

It may obviously happen that neither $\mathcal{A} \models \phi_{\oplus}$ nor $\mathcal{A} \not\models \phi_{\ominus}$. In such a case we cannot conclude whether $\mathcal{C} \models \phi$ or not, which means the chosen abstraction is not adequate.

Example 3. For instance, continuing Example 2 above, a "correct" Σ^*-structure with respect to the Σ-structure associated to the specification NatList is given by the specification:

spec NatList" =
  NatList' **with pred** $is\_in \mapsto is\_in_{\ominus}$
  %% $is\_in_{\ominus}$ is the same as is_in
  %% Recall that $\alpha(n)\ is\_in\ \alpha(l) \;\Rightarrow\; n\ is\_in\ l$
then pred $\_\_\ is\_in_{\oplus}\ \_\_ : Nat \times List$
  $\forall x : Nat;\ l : List \bullet\ x\ is\_in_{\oplus}\ l \;\Leftrightarrow\; (x\ is\_in_{\ominus}\ l \wedge\ x = 0)$

Remark 1. Related ideas (duplicating predicate symbols) have been used in some works on abstractions for reactive systems [9, 6, 12], using automata and some kind of temporal logic. Let us stress that here, due to the crucial rôle of the messages exchanged in a cryptographic protocol, it seems more convenient to model a cryptographic protocol by a first-order structure in order to obtain a uniform framework for modeling both the data part and the control part of the protocol.

Remark 2. Assume we want to validate some formula of the form $\phi \stackrel{\text{def}}{=} \exists x.\ \phi'$ and suppose that this formula is valid in the abstract domain (i.e., $\mathcal{A} \models \phi_\oplus$). In this case, there exists $a \in \mathcal{A}$ such that the valuation $x \mapsto a$ satisfy ϕ'. In order to transpose this result in $\mathcal{C}$, it is natural to intend to use an object $c \in \mathcal{C}$ such that $\alpha(c) = a$. This is why in the sequel, arbitrary homomorphisms α will not be used but only epimorphisms.

We are now ready to formalize the above intuitive principles.

Definition 1. *Let $\Sigma = (S, \Omega, \Pi)$ be a signature. The split signature Σ^* associated with Σ is defined by:*

$$\Sigma^* \stackrel{\text{def}}{=} (S, \Omega, \Pi^*)$$

where Π^* is the disjoint union of two copies of Π and of a new family of predicate symbols $\{=_\oplus : s \times s \mid s \in S^*\}$ and we note:

$$\begin{aligned} \Pi^* \stackrel{\text{def}}{=}\ & \{P_\oplus : s_1 \times \ldots \times s_n \mid P : s_1 \times \ldots \times s_n \in \Pi\} \\ & \cup \{P_\ominus : s_1 \times \ldots \times s_n \mid P : s_1 \times \ldots \times s_n \in \Pi\} \\ & \cup \{=_\oplus : s \times s \mid s \in S\}. \end{aligned}$$

Definition 2. *Given a signature $\Sigma = (S, \Omega, \Pi)$ and the associated split signature Σ^*, a Σ-structure $\mathcal{C}$, a Σ^*-structure $\mathcal{A}$ and a $[\Sigma]$-epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$, $\mathcal{A}$ is* correct *with respect to $\mathcal{C}$ and α, denoted by $\mathcal{C} \sqsubset_\alpha \mathcal{A}$, if and only if:*

i) for each $P : s_1 \times \ldots \times s_n \in \Pi$, we have, for all $c_1 \in \mathcal{C}_{s_1}, \ldots, c_n \in \mathcal{C}_{s_n}$:

$$\begin{aligned} & P_\oplus^{\mathcal{A}}(\alpha(c_1), \ldots, \alpha(c_n)) \Rightarrow P^{\mathcal{C}}(c_1, \ldots, c_n) \\ \textit{and } & P^{\mathcal{C}}(c_1, \ldots, c_n) \Rightarrow P_\ominus^{\mathcal{A}}(\alpha(c_1), \ldots, \alpha(c_n)), \end{aligned}$$

ii) for each $s \in S$, we have, for all $c, c' \in \mathcal{C}_s$:

$$\alpha(c) =_\oplus^{\mathcal{A}} \alpha(c') \Rightarrow c = c'.$$

Given a Σ-structure $\mathcal{C}$ and a $[\Sigma]$-epimorphism $\alpha : [\mathcal{C}] \to \mathcal{B}$, there are in general several ways to extend the $[\Sigma]$-algebra $\mathcal{B}$ into a Σ-structure $\mathcal{A}$ (i.e., $[\mathcal{A}] = \mathcal{B}$) correct with respect to both $\mathcal{C}$ and α. This is why we introduce a notion of precision in the definition below in order to be able to compare such extensions. Of course, when two Σ^*-structures $\mathcal{A}$ and $\mathcal{A}'$ are correct with respect to $\mathcal{C}$ and α, we have $[\mathcal{A}] = [\mathcal{A}']$. Essentially, the notion of precision only concerns the respective interpretations of the predicate symbols on $\mathcal{A}$ and $\mathcal{A}'$. For the Σ^*-structure $\mathcal{A}$, to be more precise than $\mathcal{A}'$ intuitively means that $\mathcal{A}$ allows us to validate and refute at least all the formulas that can be validated or refuted using $\mathcal{A}'$.

Definition 3. *Let $\Sigma = (S, \Omega, \Pi)$ be a signature and $\mathcal{A}, \mathcal{A}' \in \mathbf{Str}(\Sigma^*)$. $\mathcal{A}$ is more precise than $\mathcal{A}'$ (denoted by $\mathcal{A} \preccurlyeq \mathcal{A}'$) if and only if $[\mathcal{A}] = [\mathcal{A}']$ and:*

i) for all $s \in S$ and $a_1, a_2 \in \mathcal{A}_s$, $a_1 =_\oplus^{\mathcal{A}'} a_2 \Rightarrow a_1 =_\oplus^{\mathcal{A}} a_2$, *ii) for all $P : s_1 \times \ldots \times s_n \in \Pi$, for all $a_1 \in \mathcal{A}_{s_1}, \ldots, a_n \in \mathcal{A}_{s_n}$:*

$$P_\oplus^{\mathcal{A}'}(a_1,\ldots,a_n) \Rightarrow P_\oplus^{\mathcal{A}}(a_1,\ldots,a_n)$$
$$\textit{and } P_\ominus^{\mathcal{A}}(a_1,\ldots,a_n) \Rightarrow P_\ominus^{\mathcal{A}'}(a_1,\ldots,a_n)$$

This relation clearly defines a partial order on $\mathbf{Str}(\Sigma^*)$.

Proposition 1 (Conservation of correctness with respect to precision). *Given a signature Σ, a Σ-structure $\mathcal{C}$, two Σ^*-structures $\mathcal{A}$ and $\mathcal{A}'$ and a $[\Sigma]$-epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$, if $\mathcal{C} \sqsubset_\alpha \mathcal{A}$ and $\mathcal{A} \preccurlyeq \mathcal{A}'$ then we have $\mathcal{C} \sqsubset_\alpha \mathcal{A}'$.*

Proof. Obvious since for each predicate symbol $P : s_1 \times \ldots \times s_n$ in Σ and any $c_1 \in \mathcal{C}_{s_1}, \ldots, c_n \in \mathcal{C}_{s_n}$, we have, using the above definitions of correctness and precision:

$$P_\oplus^{\mathcal{A}'}(\alpha(c_1),\ldots,\alpha(c_n)) \Rightarrow P_\oplus^{\mathcal{A}}(\alpha(c_1),\ldots,\alpha(c_n)) \Rightarrow P^{\mathcal{C}}(c_1,\ldots,c_n)$$
$$P^{\mathcal{C}}(c_1,\ldots,c_n) \Rightarrow P_\ominus^{\mathcal{A}}(\alpha(c_1),\ldots,\alpha(c_n)) \Rightarrow P_\ominus^{\mathcal{A}'}(\alpha(c_1),\ldots,\alpha(c_n))$$

and similarly for $=_\oplus$. □

Definition 4 (Canonical abstraction). *Assume given a signature $\Sigma = (S, \Omega, \Pi)$, a Σ-structure $\mathcal{C}$ and $\alpha : [\mathcal{C}] \to \mathcal{B}$ an epimorphism of $[\Sigma]$-algebras. Then there is a canonical Σ^*-structure associated to $\mathcal{C}$ and α, denoted by $\mathcal{C}^\alpha$ and defined by:*

- $[\mathcal{C}^\alpha] \stackrel{\text{def}}{=} \mathcal{B}$,
- *for all $P : s_1 \times \ldots \times s_n \in \Pi$, $a_1 \in \mathcal{C}_{s_1}^\alpha, \ldots, a_n \in \mathcal{C}_{s_n}^\alpha$:*
  - *$P_\oplus^{\mathcal{C}^\alpha}(a_1,\ldots,a_n)$ if and only if for all $c_1 \in \alpha^{-1}(a_1), \ldots, c_n \in \alpha^{-1}(a_n)$, we have $P^{\mathcal{C}}(c_1,\ldots,c_n)$,*
  - *$P_\ominus^{\mathcal{C}^\alpha}(a_1,\ldots,a_n)$ if and only if there exists $c_1 \in \alpha^{-1}(a_1), \ldots, c_n \in \alpha^{-1}(a_n)$ such that $P^{\mathcal{C}}(c_1,\ldots,c_n)$,*
- *for all $s \in S$ and $a_1, a_2 \in \mathcal{C}_s^\alpha$, $a_1 =_\oplus^{\mathcal{C}^\alpha} a_2$ if and only if $a_1 = a_2$ and $\alpha^{-1}(a_1)$ and $\alpha^{-1}(a_2)$ both contain exactly one element.*

This construction allows us to reformulate the notion of correctness in terms of precision.

Proposition 2 (Characterization of correctness). *Let be given a signature Σ, a Σ-structure $\mathcal{C}$, a Σ^*-structure $\mathcal{A}$ and a $[\Sigma]$-epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$. Then $\mathcal{C} \sqsubset_\alpha \mathcal{A}$ if and only if $\mathcal{C}^\alpha \preccurlyeq \mathcal{A}$ (and consequently, $\mathcal{C}^\alpha$ is the most precise Σ^*-structure correct with respect to $\mathcal{C}$ and α).*

Proof. See the Appendix. □

Note that, using the same notations as above, if $\mathcal{A}$ is correct with respect to $\mathcal{C}$ and α, then $\mathcal{A}$ satisfies the following conditions:

i) for all $P : s_1 \times \ldots \times s_n \in \Pi$, $P_\oplus^{\mathcal{A}} \subseteq P_\ominus^{\mathcal{A}}$,
ii) $=_\oplus$ is a subset of the natural equality on $\mathcal{A}$.

Such a Σ^*-structure $\mathcal{A}$, correct with respect to $\mathcal{C}$ and α, allows us to validate (or refute) a subset of formulas on Σ, namely the atomic formulas. This property has now to be extended to more general formulas in order to obtain the abstract counterpart of the concrete properties considered.

Definition 5. *Given a Σ-formula $\phi \in \mathbf{Sen}(\Sigma)$, we define two formulas in $\mathbf{Sen}(\Sigma^*)$, denoted respectively by $\phi_\oplus$ and $\phi_\ominus$, inductively on the structure of ϕ as follows:*

- *for $P : s_1 \times \ldots \times s_n \in \Pi$, $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$:*

$$(Pt_1 \ldots t_n)_\oplus \stackrel{\text{def}}{=} P_\oplus t_1 \ldots t_n \quad \textit{and} \quad (Pt_1 \ldots t_n)_\ominus \stackrel{\text{def}}{=} P_\ominus t_1 \ldots t_n,$$

- *for $s \in S$ and $t_1, t_1 \in T_\Sigma(X)_s$:*

$$(t_1 = t_2)_\oplus \stackrel{\text{def}}{=} t_1 =_\oplus t_2 \quad \textit{and} \quad (t_1 = t_2)_\ominus \stackrel{\text{def}}{=} t_1 = t_2,$$

- *for $\phi_1, \phi_2 \in \mathbf{Sen}(\Sigma)$:*

$$\begin{aligned}
(\phi_1 \wedge \phi_2)_\oplus &\stackrel{\text{def}}{=} \phi_{1\oplus} \wedge \phi_{2\oplus}, \\
(\phi_1 \wedge \phi_2)_\ominus &\stackrel{\text{def}}{=} \phi_{1\ominus} \wedge \phi_{2\ominus}, \\
(\phi_1 \vee \phi_2)_\oplus &\stackrel{\text{def}}{=} \phi_{1\oplus} \vee \phi_{2\oplus}, \\
(\phi_1 \vee \phi_2)_\ominus &\stackrel{\text{def}}{=} \phi_{1\ominus} \vee \phi_{2\ominus},
\end{aligned}$$

- *for $\phi \in \mathbf{Sen}(\Sigma)$ and $x : s \in X$:*

$$\begin{aligned}
(\forall x : s.\ \phi)_\oplus &\stackrel{\text{def}}{=} \forall x : s.\ \phi_\oplus, \\
(\forall x : s.\ \phi)_\ominus &\stackrel{\text{def}}{=} \forall x : s.\ \phi_\ominus, \\
(\exists x : s.\ \phi)_\oplus &\stackrel{\text{def}}{=} \exists x : s.\ \phi_\oplus, \\
(\exists x : s.\ \phi)_\ominus &\stackrel{\text{def}}{=} \exists x : s.\ \phi_\ominus, \\
(\neg\phi)_\oplus &\stackrel{\text{def}}{=} \neg(\phi_\ominus), \\
(\neg\phi)_\ominus &\stackrel{\text{def}}{=} \neg(\phi_\oplus).
\end{aligned}$$

We can now state our first main result.

Theorem 1 (Correctness result). *Assume given a signature Σ, a Σ-structure $\mathcal{C}$, and a Σ^*-structure $\mathcal{A}$ such that $\mathcal{C} \sqsubset_\alpha \mathcal{A}$, where $\alpha : [\mathcal{C}] \to [\mathcal{A}]$ is a $[\Sigma]$-epimorphism. Then for any $\phi \in \mathbf{Sen}(\Sigma)$ we have:*

$$\begin{aligned}
\mathcal{A} \models \phi_\oplus &\Rightarrow \mathcal{C} \models \phi \\
\textit{and } \mathcal{C} \models \phi &\Rightarrow \mathcal{A} \models \phi_\ominus\,.
\end{aligned}$$

Proof. See the Appendix. □

Proposition 3. *Assume given a signature Σ and two Σ^*-structures $\mathcal{A}$ and $\mathcal{A}'$ such that $\mathcal{A} \preccurlyeq \mathcal{A}'$. Then for any $\phi \in \mathbf{Sen}(\Sigma)$ we have:*

$$\begin{aligned}
\mathcal{A}' \models \phi_\oplus &\Rightarrow \mathcal{A} \models \phi_\oplus \\
\textit{and } \mathcal{A} \models \phi_\ominus &\Rightarrow \mathcal{A}' \models \phi_\ominus.
\end{aligned}$$

Proof. By induction on the structure of ϕ. $\square$

The framework developed above and the corresponding results mean that, to abstract a verification problem of the form $\mathcal{C} \models \phi$, where $\mathcal{C}$ is a first-order structure and ϕ is a first-order formula (both w.r.t. a given signature Σ), it is enough to, first, abstract the corresponding underlying algebra $[\mathcal{C}]$ by choosing some $[\Sigma]$-epimorphism $\alpha : [\mathcal{C}] \to \mathcal{B}$, and then, extend the $[\Sigma]$-algebra $\mathcal{B}$ into a Σ^*-structure $\mathcal{A}$ correct w.r.t. $\mathcal{C}$ and α, which amounts to choose a suitable interpretation for the duplicated predicate symbols $P_\oplus$ and $P_\ominus$ (and $\mathcal{C}^\alpha$ is a good candidate). Then we can use $\phi_\oplus$ and $\phi_\ominus$ which are syntactically derived from ϕ.

4 Optimality of Algebraic Abstractions

In the previous section, we have explained how to use algebraic abstractions which are, by construction, correct abstractions. However, a correct algebraic abstraction may turn out not to be adequate, and in this case one has to look for another better suited abstraction. We will show in this section that, given a verification problem of the form $\mathcal{C} \models \phi$, for each choice of an epimorphism $\alpha : [\mathcal{C}] \to \mathcal{B}$, there is a corresponding optimal algebraic abstraction: if it is not adequate, then any other abstraction built from the same epimorphism will not be adequate either. Hence, our optimality results will allow us to reduce the adequacy problem to the right choice of the underlying epimorphism of algebras. Intuitively, Proposition 3 shows that $\mathcal{C}^\alpha$ is the "best" structure we can use once α is chosen. It is however far from obvious whether our syntactic abstraction of formulas ϕ into $\phi_\oplus$, $\phi_\ominus$ is optimal or not. The main aim of this section is to formally establish that $(\phi_\oplus, \phi_\ominus)$ is indeed the best possible choice.

Hence, we will prove that, for each application:

$$\begin{array}{rl} \tau : \mathbf{Sen}(\Sigma) & \to \mathbf{Sen}(\Sigma^*)^2 \\ \phi & \mapsto (\tau^1(\phi), \tau^2(\phi)) \end{array}$$

satisfying for all $\phi \in \mathbf{Sen}(\Sigma)$ and for all $\mathcal{C} \in \mathbf{Str}(\Sigma)$ and $\mathcal{A} \in \mathbf{Str}(\Sigma^*)$ correct with respect to $\mathcal{C}$ and an epimorphism α:

$$\mathcal{A} \models \tau^1(\phi) \Rightarrow \mathcal{C} \models \phi \quad \text{and} \quad \mathcal{C} \models \phi \Rightarrow \mathcal{A} \models \tau^2(\phi)$$

we have:

$$\mathcal{A} \models \tau^1(\phi) \Rightarrow \mathcal{A} \models \phi_\oplus \quad \text{and} \quad \mathcal{A} \models \phi_\ominus \Rightarrow \mathcal{A} \models \tau^2(\phi)$$

for all $\phi \in \mathbf{Sen}(\Sigma)$. We will assume that these applications τ are compatible with signature morphisms, since in our framework it only makes sense to consider abstraction techniques generic w.r.t. the signatures of the structures to be abstracted.

To prove our optimality result, we will rely on a technical trick, namely we will abstract again $\mathcal{A}$, and we will therefore consider a further split signature Σ^{**}. Moreover, in a first step, we will work within the **FOL** institution (instead of the **FOLeq** institution), and then we will of course come back to **FOLeq**.

4.1 Optimality in FOL

The institution of (many-sorted) first-order logic is denoted **FOL**:

$$\mathbf{FOL} = (\mathbf{Sign}, \mathbf{Sen}_{\mathbf{FOL}}(), \mathbf{Str}(), \models)$$

(where **Sign** and **Str**() are the same as in **FOLeq**). Abstractions can be done in **FOL** in the same way as in **FOLeq**, but since there is no predicate $=_{\oplus}$ to add, we use, instead of Σ^*, the signature:

$$\Sigma^\bullet = (S, \Omega, \{P_\oplus \mid P \in \Pi\} \cup \{P_\ominus \mid P \in \Pi\})$$

where $\Sigma = (S, \Omega, \Pi)$.

In order to state and prove the optimality result in **FOL**, we are going to proceed as follow:

- abstraction of signatures will be turned into a functor (then it will be possible to define a translation which is compatible with respect to signature morphisms),
- a $\Sigma^{\bullet\bullet}$-structure will be associated to each $\Sigma^\bullet$-structure,
- then it will be possible to prove the optimality result using abstractions.

With each signature morphism $\sigma : \Sigma \to \Sigma'$ is associated a morphism $\sigma^\bullet : \Sigma^\bullet \to \Sigma'^\bullet$ defined by:

- $\sigma^\bullet$ acts as σ on the sorts and function symbols of Σ,
- for each predicate symbol P of Σ:

$$\sigma^\bullet(P_\oplus) \stackrel{\text{def}}{=} (\sigma(P))_\oplus \quad \text{and} \quad \sigma^\bullet(P_\ominus) \stackrel{\text{def}}{=} (\sigma(P))_\ominus.$$

Moreover, for each signature Σ there is a canonical homomorphism:

$$|.|_\Sigma : \Sigma^\bullet \to \Sigma$$

defined by:

- $|.|_\Sigma$ is the identity on sorts and function symbols,
- for each predicate symbol P of Σ:

$$|P_\oplus|_\Sigma \stackrel{\text{def}}{=} P \text{ and } |P_\ominus|_\Sigma \stackrel{\text{def}}{=} P.$$

Then, the following diagram is commutative:

$$\begin{tikzcd}
\Sigma \arrow[r, "\sigma"] & \Sigma' \\
\Sigma^\bullet \arrow[u, "|.|_\Sigma"] \arrow[r, "\sigma^\bullet"'] & \Sigma'^\bullet \arrow[u, "|.|_{\Sigma'}"']
\end{tikzcd}$$

Finally, in the proof of optimality, we are going to use the satisfaction condition in first-order logic, i.e., given a signature morphism $\sigma : \Sigma \to \Sigma'$, $\phi \in \mathbf{Sen_{FOL}}(\Sigma)$ and $\mathcal{C} \in \mathbf{Str_{FOL}}(\Sigma')$ we have:

$$\mathcal{C} \models \sigma(\phi) \Leftrightarrow \mathcal{C}|_\sigma \models \phi$$

(see [2] for details).

So, there are two homomorphisms from $\Sigma^{\bullet\bullet}$ to $\Sigma^\bullet$:

- the first one is the canonical homomorphism $|.|_{\Sigma^\bullet} : \Sigma^{\bullet\bullet} \to \Sigma^\bullet$, which satisfies for each predicate P in Σ annotated by s_1 in $\Sigma^\bullet$ and s_2 in $\Sigma^{\bullet\bullet}$:
$$|(P_{s_1})_{s_2}|_{\Sigma^\bullet} = P_{s_1},$$
- the second one is the homomorphism associated to the canonical homomorphism $|.|_\Sigma : \Sigma^\bullet \to \Sigma$, denoted by $|.|^\bullet_\Sigma$, and we have:
$$|(P_{s_1})_{s_2}|^\bullet_\Sigma = (|P_{s_1}|_\Sigma)_{s_2} = P_{s_2}.$$

When we reduce a $\Sigma^\bullet$-structure $\mathcal{A}$ along $|.|_{\Sigma^\bullet}$, we obtain the structure defined in Section 3 (Definition 4) and denoted $\mathcal{A}^{id}$ (where id denotes the identity epimorphism). This structure is known to be correct with respect to $\mathcal{A}$, but, in the proof of optimality, we need to reduce $\mathcal{A}$ along the other homomorphism. So, we consider $\mathcal{A}|_{|.|^\bullet_\Sigma}$, and the first thing to do is to prove that this structure is also correct with respect to $\mathcal{A}$. This is the aim of the following lemma.

Lemma 1. *Let be given a signature Σ and a $\Sigma^\bullet$-structure $\mathcal{A} \in \mathbf{Str_{FOL}}(\Sigma^\bullet)$ such that for each predicate symbol $P : s_1 \times \ldots \times s_n$ in Σ:*

$$P^{\mathcal{A}}_\oplus \subseteq P^{\mathcal{A}}_\ominus.$$

Let $|.|^\bullet_\Sigma$ denote the signature homomorphism from $\Sigma^{\bullet\bullet}$ to $\Sigma^\bullet$ associated to the canonical homomorphism $|.|_\Sigma$ from $\Sigma^\bullet$ to Σ. Then $\mathcal{A} \sqsubset_{id} \mathcal{A}|_{|.|^\bullet_\Sigma}$, where id denote the identity from $[\mathcal{A}]$ to $[\mathcal{A}|_{|.|^\bullet_\Sigma}] = [\mathcal{A}]$.

Proof. See the Appendix. □

We now just have to define the class of translations considered and we can prove the optimality theorem.

Definition 6. *A translation (in* **FOL***) is a family $\tau = (\tau_\Sigma)_{\Sigma \in \mathbf{Sign}}$ where, for each $\Sigma \in \mathbf{Sign}$, τ_Σ is an application:*

$$\begin{array}{l} \tau_\Sigma : \mathbf{Sen_{FOL}}(\Sigma) \to \mathbf{Sen_{FOL}}(\Sigma^\bullet)^2 \\ \qquad\quad \phi \qquad\;\; \mapsto (\tau^1_\Sigma(\phi), \tau^2_\Sigma(\phi)) \end{array}$$

A translation is correct *if for all $\Sigma \in \mathbf{Sign}$, $\phi \in \mathbf{Sen_{FOL}}(\Sigma)$, $\mathcal{C} \in \mathbf{Str_{FOL}}(\Sigma)$ and $\mathcal{A} \in \mathbf{Str_{FOL}}(\Sigma^\bullet)$ correct with respect to $\mathcal{C}$ and an epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$ we have:*

$$\mathcal{A} \models \tau^1_\Sigma(\phi) \Rightarrow \mathcal{C} \models \phi \quad \textit{and} \quad \mathcal{C} \models \phi \Rightarrow \mathcal{A} \models \tau^2_\Sigma(\phi)$$

and natural *if for* $\Sigma, \Sigma' \in \mathbf{Sign}$*,* $\sigma : \Sigma \to \Sigma'$ *and* $\phi \in \mathbf{Sen_{FOL}}(\Sigma)$*, we have:*

$$\tau_{\Sigma'}(\sigma(\phi)) = \sigma^\bullet(\tau_\Sigma(\phi))$$

i.e., for each $\Sigma \in \mathbf{Sign}$ *the following diagram is commutative:*

$$\begin{tikzcd}
\mathbf{Sen_{FOL}}(\Sigma) \arrow[r, "\tau_\Sigma"] \arrow[d, "\sigma"'] & \mathbf{Sen_{FOL}}(\Sigma^\bullet)^2 \arrow[d, "\sigma^\bullet \times \sigma^\bullet"] \\
\mathbf{Sen_{FOL}}(\Sigma') \arrow[r, "\tau_{\Sigma'}"'] & \mathbf{Sen_{FOL}}(\Sigma'^\bullet)^2
\end{tikzcd}$$

Remark 3. Consider the functor $\mathbf{Sen^\bullet_{FOL}}(.)$ from **Sign** to **Set** (the category of sets) defined by:

– for $\Sigma \in \mathbf{Sign}$, $\mathbf{Sen^\bullet_{FOL}}(\Sigma) = \mathbf{Sen_{FOL}}(\Sigma^\bullet)^2$,
– for $\sigma : \Sigma \to \Sigma'$, $\mathbf{Sen^\bullet_{FOL}}(\sigma) = \sigma^\bullet \times \sigma^\bullet$,

then a translation is natural if and only if it defines a natural transformation from $\mathbf{Sen_{FOL}}(.)$ to $\mathbf{Sen^\bullet_{FOL}}(.)$.

Theorem 2 (Optimality in FOL). *Let be given a correct and natural translation* τ *and let* $\Sigma \in \mathbf{Sign}$ *and* $\mathcal{A} \in \mathbf{Str_{FOL}}(\Sigma^\bullet)$ *be such that for each predicate symbol* $P : s_1 \times \ldots \times s_n$ *in* Σ*, we have* $P^{\mathcal{A}}_\oplus \subseteq P^{\mathcal{A}}_\ominus$*. Then:*

$$\mathcal{A} \models \tau^1_\Sigma(\phi) \Rightarrow \mathcal{A} \models \phi_\oplus \quad \textit{and} \quad \mathcal{A} \models \phi_\ominus \Rightarrow \mathcal{A} \models \tau^2_\Sigma(\phi).$$

Proof. See the Appendix. □

Remark 4. This result holds in particular when $\mathcal{A}$ is correct with respect to $\mathcal{C} \in \mathbf{Str_{FOL}}(\Sigma)$ and an epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$.

4.2 Optimality in FOLeq

The optimality result in **FOLeq** will be proved using the previous theorem in **FOL** and that is why we first define an encoding (following [14]) of **FOLeq** into **FOL**. Since sentences and structures in **FOL** can be considered as sentences and structures in **FOLeq**, we finally obtain an encoding of **FOLeq** into **FOLeq**. Then, the class of translations considered in **FOLeq** is the class of translations correct, natural (as above) and commuting with this encoding.

For each $\Sigma = (S, \Omega, \Pi) \in \mathbf{Sign}$, we consider the signature:

$$\overline{\Sigma} = (S, \Omega, \Pi \cup \{eq_\Sigma : s \times s \mid s \in S\})$$

(where $\{eq_\Sigma : s \times s \mid s \in S\}$ is a fresh family of predicate symbols). Using $\overline{\Sigma}$, we obtain a one to one translation of formulas:

$$\begin{array}{lcl} \mathbf{Sen}(\Sigma) & \to & \mathbf{Sen_{FOL}}(\overline{\Sigma}) \\ \phi & \mapsto & \overline{\phi} \end{array}$$

and a mapping on the structures:

$$\begin{array}{lcl} \mathbf{Str}(\Sigma) & \to & \mathbf{Str}(\overline{\Sigma}) \\ \mathcal{C} & \mapsto & \overline{\mathcal{C}} \end{array}$$

satisfying the property:

$$\mathcal{C} \models \phi \;\Leftrightarrow\; \overline{\mathcal{C}} \models \overline{\phi}$$

These mappings define an encoding of **FOLeq** into **FOL**. Moreover, $\overline{\Sigma^*}$ and $\overline{\Sigma}^\bullet$ are clearly isomorphic and in the sequel we are going to identify them. Since $\overline{\Sigma}^\bullet \subseteq \overline{\Sigma}^*$, the following relations between sentences hold:

$$\begin{array}{l} \mathbf{Sen_{FOL}}(\Sigma) \subseteq \mathbf{Sen}(\Sigma) \cong \mathbf{Sen_{FOL}}(\overline{\Sigma}), \\ \mathbf{Sen}(\overline{\Sigma^*}) \subseteq \mathbf{Sen}(\overline{\Sigma}^*). \end{array}$$

In **FOLeq**, we need to consider translations commuting with the above encoding of **FOLeq** into itself. Such a translation is called homogeneous. The homogeneity condition states that the equality is processed like predicates, at the syntactic level.

Definition 7. *A translation (in* **FOLeq***) is a family $\tau = (\tau_\Sigma)_{\Sigma \in \mathbf{Sign}}$, where for each $\Sigma \in \mathbf{Sign}$, τ_Σ is an application $\tau_\Sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma^*)^2$. Correctness and naturality of such a translation can be defined like in* **FOL***. Moreover, τ is said to be* homogeneous *if for each $\Sigma \in \mathbf{Sign}$ the following diagram is commutative :*

$$\begin{tikzcd} \mathbf{Sen}(\Sigma) \arrow[r, "\simeq"] \arrow[d, "\tau_\Sigma"'] & \mathbf{Sen_{FOL}}(\overline{\Sigma}) \arrow[r, hook] & \mathbf{Sen}(\overline{\Sigma}) \arrow[d, "\tau_{\overline{\Sigma}}"] \\ \mathbf{Sen}(\Sigma^*)^2 \arrow[r, "\simeq"] & \mathbf{Sen}(\overline{\Sigma^*})^2 \arrow[r, hook] & \mathbf{Sen}(\overline{\Sigma}^*)^2 \end{tikzcd}$$

(i.e., for all $\phi \in \mathbf{Sen}(\Sigma)$, $\tau_{\overline{\Sigma}}(\overline{\phi}) = \overline{\tau_\Sigma(\phi)}$).

The following lemma states that a correct, natural and homogeneous translation in **FOLeq** induces (and is entirely defined by) a correct and natural translation in **FOL**.

Lemma 2. *Let τ be a natural and homogeneous translation in* **FOLeq***, then the following properties hold:*

i) for each $\Sigma \in \mathbf{Sign}$, $\tau_\Sigma(\mathbf{Sen_{FOL}}(\Sigma)) \subseteq \mathbf{Sen_{FOL}}(\Sigma^*)^2$ (hence τ induces a natural translation $\tau^{\mathbf{FOL}}$ in **FOL** *which is correct when τ is correct),*
ii) for each $\Sigma \in \mathbf{Sign}$, the following diagram is commutative:

$$\begin{tikzcd} \mathbf{Sen_{FOL}}(\Sigma) \arrow[r, hook] \arrow[d, "\tau^{\mathbf{FOL}}_\Sigma"'] & \mathbf{Sen}(\Sigma) \arrow[r, "\simeq"] \arrow[d, "\tau_\Sigma"'] & \mathbf{Sen_{FOL}}(\overline{\Sigma}) \arrow[d, "\tau^{\mathbf{FOL}}_{\overline{\Sigma}}"] \\ \mathbf{Sen_{FOL}}(\Sigma^\bullet)^2 \arrow[r, hook] & \mathbf{Sen}(\Sigma^*)^2 \arrow[r, "\simeq"] & \mathbf{Sen_{FOL}}(\overline{\Sigma}^\bullet)^2 \end{tikzcd}$$

Proof. See the Appendix. □

Theorem 3 (Optimality in FOLeq). *Let τ be a correct, natural and homogeneous translation and let $\Sigma \in$ **Sign** and $\mathcal{A} \in \mathbf{Str}(\Sigma^*)$ be such that, for each predicate symbol $P : s_1 \times \ldots \times s_n$ in Σ, we have $P_\oplus^\mathcal{A} \subseteq P_\ominus^\mathcal{A}$, and such that $=_\oplus^\mathcal{A}$ is a subset of the natural equality on $\mathcal{A}$. Then:*

$$\mathcal{A} \models \tau_\Sigma^1(\phi) \Rightarrow \mathcal{A} \models \phi_\oplus \quad \textit{and} \quad \mathcal{A} \models \phi_\ominus \Rightarrow \mathcal{A} \models \tau_\Sigma^2(\phi).$$

Proof. See the Appendix. □

Combining this result and Proposition 3, we obtain the following result about optimality of algebraic abstractions.

Corollary 1. *Assume given a correct, natural and homogeneous translation τ, a signature Σ, a Σ-structure $\mathcal{C}$, a Σ^*-structure $\mathcal{A}$ correct with respect to $\mathcal{C}$ and an epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$. Then for any $\phi \in \mathbf{Sen}(\Sigma)$ we have:*

$$\begin{array}{c} \mathcal{A} \models \tau_\Sigma^1(\phi) \;\Rightarrow\; \mathcal{C}^\alpha \models \phi_\oplus \\ \textit{and}\;\; \mathcal{C}^\alpha \models \phi_\ominus \;\Rightarrow\; \mathcal{A} \models \tau_\Sigma^2(\phi). \end{array}$$

Proof (of Corollary 1). Since $\mathcal{C} \sqsubset_\alpha \mathcal{A}$, we can deduce $\mathcal{A} \models \phi_\oplus$ from $\mathcal{A} \models \tau_\Sigma^1(\phi)$ using Theorem 3. Then, applying Proposition 3, we obtain $\mathcal{C}^\alpha \models \phi_\oplus$. The proof is the same for the second implication. □

5 Algebraic Abstractions at Work

According to the previous section, the main task to design an abstraction for a verification problem $\mathcal{C} \models \phi$ is to abstract $\mathcal{C}$. For this, as already explained, the non trivial part is to find suitable interpretations for the predicate symbols $P_\oplus$ and $P_\ominus$. In this section, we consider specific important cases for which we provide relevant guidelines.

The first important case is when the predicate to be abstracted is specified by a formula not involving this predicate. Abstracting it is in that case pretty obvious.

Proposition 4 (Predicate defined by a formula). *Assume given a structure $\mathcal{C}$ defined on a signature Σ of the form $\Sigma = \Sigma' \cup \{$**pred** $P : s_1 \times \ldots \times s_n\}$, such that $\mathcal{C} \models P x_1 \ldots x_n \Leftrightarrow \phi$, with $\phi \in \mathbf{Sen}(\Sigma')$. Moreover, assume the abstraction problem solved for $\mathcal{C}' = \mathcal{C}|_{\Sigma'}$ (i.e., assume given a Σ'^*-structure $\mathcal{A}'$ and an epimorphism $\alpha : [\mathcal{C}'] \to [\mathcal{A}']$ such that $\mathcal{A}'$ is correct with respect to $\mathcal{A}$ and α). Then any Σ-structure $\mathcal{A}$ satisfying:*

i) $\mathcal{A}|_{\Sigma'^*} = \mathcal{A}'$,
ii) $\mathcal{A} \models P_\oplus x_1 \ldots x_n \Leftrightarrow \phi_\oplus$,
iii) $\mathcal{A} \models P_\ominus x_1 \ldots x_n \Leftrightarrow \phi_\ominus$.

is correct with respect to $\mathcal{C}$ and α.

Proof. Follows directly from Theorem 1. □

Example 4. In our cryptographic protocol models, the transition relation is defined by an axiom of the form:

$$q \xrightarrow{(l,m)} q' \Leftrightarrow r$$

and consequently we can use $r_\oplus$ and $r_\ominus$ in order to define $\overset{(,)}{\rightarrow}_\oplus$ and $\overset{(,)}{\rightarrow}_\ominus$ (disregarding how the rest of the abstraction is defined).

Remark 5. The above result will prove useful for structures defined by a specification (Σ, Φ) which can be stratified as follows:

- the set of predicate symbols of Σ is a disjoint union $\Pi \cup \{P_1, \ldots, P_n\}$,
- Φ is a set of Σ-sentences of the form:

  $$\Phi = \Phi_0 \cup \Phi_1 \cup \ldots \cup \Phi_n$$

  where, for $i \in \{0, \ldots, n\}$, Φ_i contains no occurrence of a predicate symbol P_j for $j \geq i$ and for $i \geq 1$, Φ_i is of the form:

  $$\Phi_i = \{P_i x_1 \ldots x_k \Leftrightarrow \phi_i\}.$$

The second special case is when the predicate to be abstracted is defined inductively by some set of rules.

Proposition 5 (Predicate defined by a set of rules). *Assume given a structure $\mathcal{C}$ defined on a signature Σ of the form $\Sigma = \Sigma' \cup \{\mathbf{pred}\ P : s_1 \times \ldots \times s_n\}$, such that the interpretation of P in $\mathcal{C}$ is the least predicate satisfying a set $\mathcal{R}$ of rules of the form:*

$$R = \frac{Pt_1^1 \ldots t_n^1 \quad \ldots \quad Pt_1^m \ldots t_n^m \quad \phi}{Pt_1^0 \ldots t_n^0}$$

with $\phi \in \mathbf{Sen}(\Sigma)$, $t_1^0, \ldots, t_1^m \in T_\Sigma(X)_{s_1}$ and $t_n^0, \ldots, t_n^m \in T_\Sigma(X)_{s_n}$. Moreover, assume the abstraction problem solved for $\mathcal{C}' = \mathcal{C}|_{\Sigma'}$ (i.e., assume given a Σ'^*-structure $\mathcal{A}'$ and an epimorphism $\alpha : [\mathcal{C}'] \to [\mathcal{A}']$ such that $\mathcal{A}'$ is correct with respect to $\mathcal{A}$ and α). Then any Σ-structure $\mathcal{A}$ satisfying:

i) $\mathcal{A}|_{\Sigma'^*} = \mathcal{A}'$,
ii) for all $c_1 \in \mathcal{C}_{s_1}, \ldots, c_n \in \mathcal{C}_{s_n}$, $P_\oplus^{\mathcal{A}}(\alpha(c_1), \ldots, \alpha(c_n)) \Rightarrow P^{\mathcal{C}}(c_1, \ldots, c_n)$,
iii) $P_\ominus$ *is the least predicate on $\mathcal{A}$ satisfying each rule in $\mathcal{R}_\ominus$, obtained from $\mathcal{R}$ by replacing each rule R by:*

$$R_\ominus \stackrel{\text{def}}{=} \frac{P_\ominus t_1^1 \ldots t_n^1 \quad \ldots \quad P_\ominus t_1^m \ldots t_n^m \quad \phi_\ominus}{P_\ominus t_1^0 \ldots t_n^0}$$

is correct with respect to $\mathcal{C}$ and α.

Proof. See the Appendix. $\square$

Example 5. For cryptographic protocols, the capacities of the intruder are modeled by a predicate $\Vdash$ defined by a set of rules. So $\Vdash_\ominus$ can be defined by the following rules:

$$\frac{m \in_\ominus s}{s \Vdash_\ominus m} \quad (\xi)$$

$$\frac{s \Vdash_\ominus m_1 \qquad s \Vdash_\ominus m_2}{s \Vdash_\ominus \langle m_1, m_2\rangle} \quad (\pi)$$

$$\frac{s \Vdash_\ominus \langle m_1, m_2\rangle}{s \Vdash_\ominus m_1} \quad (\pi'_1) \qquad \frac{s \Vdash_\ominus \langle m_1, m_2\rangle}{s \Vdash_\ominus m_2} \quad (\pi'_2)$$

$$\frac{s \Vdash_\ominus m \quad s \Vdash_\ominus k}{s \Vdash_\ominus \{m\}_k} \quad (\gamma) \qquad \frac{s \Vdash_\ominus \{m\}_k \quad s \Vdash_\ominus k^{-1}}{s \Vdash_\ominus m} \quad (\gamma')$$

Example 6 (An abstraction for the study of cryptographic protocols). Let $\Sigma = (S, \Omega, \Pi)$ denote the signature of the specification of our toy cryptographic protocol given in Example 1 and let $\mathcal{C}$ be a model of this specification. We are going to describe a possible abstraction, wich will be useful to prove security properties about $\mathcal{C}$. The abstract structure $\mathcal{A}$ and the epimorphism $\alpha : [\mathcal{C}] \to [\mathcal{A}]$ will be described by the mean of the carrier sets of $\mathcal{A}$ together with a function $\alpha_s : \mathcal{C}_s \to \mathcal{A}_s$ for each $s \in S$ (recall that $S = \{Princ, Key, Msg, Set[Msg], Lbl, State\}$):

- $\mathcal{A}_{Princ}$ is a set containing two values, S and A, and α_{Princ} is defined by:

$$\begin{array}{rll} \alpha_{Princ} : \mathcal{C}_{Princ} & \to \mathcal{A}_{Princ} \\ S & \mapsto S \\ x & \mapsto A \text{ if } x \neq S \end{array}$$

- $\mathcal{A}_{Key}$ is a set containing four values, $k(S)$, $k^{-1}(S)$, $k(A)$ and $k^{-1}(A)$, and α_{Key} is defined by:

$$\begin{array}{rll} \alpha_{Key} : \mathcal{C}_{Key} & \to \mathcal{A}_{Key} \\ k(S) & \mapsto k(S) \\ k^{-1}(S) & \mapsto k^{-1}(S) \\ k(x) & \mapsto k(A) \text{ if } x \neq S \\ k^{-1}(x) & \mapsto k^{-1}(A) \text{ if } x \neq S \end{array}$$

- from an idea of D. Le Métayer, an element of $\mathcal{A}_{Msg}$ is a non-empty set of "rules" r of the form:

$$r = k_1, \ldots, k_n \vdash d$$

  where $k_1, \ldots, k_n \in \mathcal{A}_{Key}$ and $d \in \mathcal{A}_{Princ} \cup \mathcal{A}_{Key}$. For each $m \in \mathcal{C}_{Msg}$, $\alpha_{Msg}(m)$ is defined inductively on the structure of m:
  - if $m = d$ is atomic, $\alpha_{Msg}(m) = \{\vdash \alpha(d)\}$,

- if $m = \langle m_1, m_2 \rangle$, $\alpha_{Msg}(m) = \alpha_{Msg}(m_1) \cup \alpha_{Msg}(m_2)$,
- if $m = \{m'\}_k$, $\alpha_{Msg}(m) = \{\alpha_{Key}(k).r \mid r \in \alpha_{Msg}(m')\}$ where, for each $r = k_1, \ldots, k_n \vdash d \in \alpha_{Msg}(m')$, $\alpha_{Key}(k).r$ denotes the rule $\alpha_{Key}(k), k_1, \ldots, k_n \vdash d$

For example:

$$\alpha_{Msg}(\langle \{d_0\}_k, \{\{d_1\}_{k_2}\}_{k_1} \rangle) = \{k \vdash d_0;\ k_1, k_2 \vdash d_1\}$$

Formally, this part of the abstract structure is described by the following specification:

spec MESSAGE' =
PRINCIPAL' **and** KEY'
and SET[**sort** *Key*] **and** NON-EMPTY-SET[**sort** *Rule*]
then **free types** $Atom ::= sort\ Key |\ sort\ Princ;$
$Rule\ ::= \_\_ \vdash \_\_(Set[Key];\ Atom);$
$Msg\ ::= Non - Empty - Set[Rule]$
ops $\langle \_\_, \_\_ \rangle : Msg \times Msg \to Msg;$
$\{\_\_\}_{\_\_} : Msg \times Key \to Msg;$
then %def
$\forall m, m' : Msg;\ k : Key;\ s : Set[Key];\ d : Atom$
- $\langle m, m' \rangle\ =\ m \cup m'$
- $\{\{s \vdash d\}\}_k\ =\ \{\{k\} \cup s \vdash d\}$
- $\{m \cup \{s \vdash d\}\}_k\ =\ \{m\}_k \cup \{\{k\} \cup s \vdash d\}$

end

– $\mathcal{A}_{Set[Msg]}$ is the powerset of $\mathcal{A}_{Msg}$ and $\alpha_{Set[Msg]}$ is the map induced by α_{Msg}:

$$\begin{aligned} \alpha_{Set[Msg]} : \mathcal{C}_{Set[Msg]} &\to \mathcal{A}_{Set[Msg]} \\ \{m_1, \ldots, m_n\} &\mapsto \{\alpha_{Msg}(m_1), \ldots, \alpha_{Msg}(m_n)\} \end{aligned}$$

– $\mathcal{A}_{Lbl} = \mathcal{C}_{Lbl}$ and α_{Lbl} is the identity map,

– $\mathcal{A}_{State} = \mathcal{A}_{Set[Msg]} \times \mathcal{A}_{Princ} \times \mathcal{A}_{Set[Msg]}$ (recall that $\mathcal{C}_{State} = \mathcal{C}_{Set[Msg]} \times \mathcal{C}_{Princ} \times \mathcal{C}_{Set[Msg]}$) and α_{State} is defined by:

$$\begin{aligned} \alpha_{State} : \mathcal{C}_{State} &\to \mathcal{A}_{State} \\ (dir, id, I) &\mapsto (\alpha_{Set[Msg]}(dir), \alpha_{Princ}(id), \alpha_{Set[Msg]}(I)) \end{aligned}$$

Now we would have to define interpretations for the function symbols of Σ in order to turn the family $(\mathcal{A}_s)_{s \in S}$ into a $[\Sigma]$-algebra denoted $[\mathcal{A}]$, but we are going to skip this step since it is pretty obvious. So, we just have to give the interpretations of the predicate symbols of Σ^*, which are:

$$
\begin{array}{ll}
\_\_ =_\oplus \_\_ & : s \times s \text{ (for each } s \in S), \\
\_\_ \in_\oplus \_\_ & : Msg \times Set[Msg], \\
\_\_ \in_\ominus \_\_ & : Msg \times Set[Msg], \\
\_\_ \Vdash_\oplus \_\_ & : Set[Msg] \times Msg, \\
\_\_ \Vdash_\ominus \_\_ & : Set[Msg] \times Msg, \\
\_\_ \xrightarrow{(\_\_,\_\_)}_\oplus \_\_ & : State \times Lbl \times Msg \times State, \\
\_\_ \xrightarrow{(\_\_,\_\_)}_\ominus \_\_ & : State \times Lbl \times Msg \times State, \\
Reach_\oplus & : State \times State, \\
Reach_\ominus & : State \times State
\end{array}
$$

in order to obtain a Σ^*-structure correct with respect to $\mathcal{C}$ and α.

- $=_\oplus$ is defined on $\mathcal{A}_{Princ}$ and $\mathcal{A}_{key}$ using the following axioms:

$$
\begin{array}{l}
\forall x, y : Princ.\ x =_\oplus y \ \Leftrightarrow \ x \ = \ y \ = \ S \\
\forall x, y : Key.\ x =_\oplus y \ \Leftrightarrow \ x \ = \ y \ = \ k(S) \vee x \ = \ y \ = \ k^{-1}(S)
\end{array}
$$

- concerning the sort Msg, for $m, m' \in \mathcal{A}_{Msg}$, $m =_\oplus m'$ is true iff m and m' are equal to the same atomic data $d \in \mathcal{A}_{Princ} \cup \mathcal{A}_{Key}$ and, as an atomic data, $d =_\oplus d$,
- for two sets of messages $s, s' \in \mathcal{A}_{Set[Msg]}$, $s =_\oplus s'$ is true iff s and s' are equal to the same set $\{m_1, \ldots, m_n\}$ with $m_1 =_\oplus m_1, \ldots, m_n =_\oplus m_n$,
- concerning the sorts Lbl and $State$, $=_\oplus$ is defined on $\mathcal{A}_{Lbl}$ and $\mathcal{A}_{State}$ using the following axioms:

$$
\begin{array}{l}
\forall x, y : Lbl.\ x =_\oplus y \ \Leftrightarrow \ x \ = \ y, \\
\forall x, y : State.\ x =_\oplus y \ \Leftrightarrow \ x.Adir =_\oplus y.Adir \wedge x.Sid =_\oplus y.Sid \wedge x.I =_\oplus y.I
\end{array}
$$

- $\in_\ominus$ is the usual membership predicate for sets of messages whereas $\in_\oplus$ is defined by:

$$\forall m : Msg. \forall s : Set[Msg].\ m \in_\oplus s \ \Leftrightarrow \ m \in_\ominus s \wedge m =_\oplus m$$

- $\Vdash_\ominus$ is defined by the same rules as $\Vdash$ (see Example 5) whereas $\Vdash_\oplus$ is defined using the rule:

$$\frac{k_1, \ldots, k_n \vdash d \in_\oplus s \quad s \Vdash_\oplus k_1 \quad \ldots \quad s \Vdash_\oplus k_n}{s \Vdash_\oplus d}$$

  (this definition is the only one, among all the others, that needs a non trivial proof of correctness),
- $\xrightarrow{(,)}_\oplus$ and $\xrightarrow{(,)}_\ominus$ are defined as in Example 4,
- using Proposition 5, $Reach_\ominus$ is defined by the two rules:

$$\frac{}{Reach_\ominus(q, q)} \qquad \frac{Reach_\ominus(q, q') \quad q' \xrightarrow{(l,m)}_\ominus q''}{Reach_\ominus(q, q'')}$$

whereas the following definition of $Reach_\oplus$ can be proved to be correct:

$$\frac{q =_\oplus q}{Reach_\oplus(q,q)} \qquad \frac{Reach_\oplus(q,q') \quad q' \xrightarrow{(l,m)}_\oplus q''}{Reach_\oplus(q,q'')}$$

Finally, we obtain a Σ^*-structure $\mathcal{A}$ which is correct with respect to $\mathcal{C}$ and α and finite. So, if we want to prove a property of the protocol, for instance if we want to check that:

$$\mathcal{C} \models \forall q_0, q : State \bullet \quad \neg(q_0.I \Vdash k^{-1}(S)) \wedge Reach(q_0, q) \;\Rightarrow\; \neg(q \Vdash k^{-1}(S))$$

we just have to show that:

$$\mathcal{A} \models \forall q_0, q : State \bullet \quad \neg(q_0.I \Vdash_\oplus k^{-1}(S)) \wedge Reach_\ominus(q_0, q) \;\Rightarrow\; \neg(q \Vdash_\ominus k^{-1}(S))$$

and this can be done using model-checking techniques (recall that $\mathcal{A}$ is finite).

6 Conclusion

In this paper, we have provided a general framework for defining correct abstractions for verification problems of the form $\mathcal{C} \models \phi$, where $\mathcal{C}$ is a first-order structure and ϕ is a first-order formula (both w.r.t. a given signature Σ). We have proven that our algebraic abstractions are in some sense optimal, and moreover we have provided useful guidelines to design abstractions in practice, in particular for first-order structures modeling cryptographic protocols.

We now intend to improve our guidelines in order to automatize as much as possible the abstraction step and to build libraries of abstractions suited for some classes of cryptographic protocols.

Proofs

Proof (of Proposition 2). Concerning $=_\oplus$, for $s \in S$ and $a_1, a_2 \in \mathcal{A}_s$, if we have $a_1 =_\oplus^{\mathcal{A}} a_2$, then for any $c_1 \in \alpha^{-1}(a_1)$ and $c_2 \in \alpha^{-1}(a_2)$, we have $c_1 = c_2$, since $a_1 = a_2$ and $\alpha^{-1}(a_1) = \alpha^{-1}(a_2)$ contains exactly one element. For a predicate $P : s_1 \times \ldots \times s_n$ in Σ and $a_1 \in \mathcal{A}_{s_1}, \ldots, a_n \in \mathcal{A}_{s_n}$, we have:

$$P_\oplus^{\mathcal{A}}(a_1, \ldots, a_n) \Rightarrow P^{\mathcal{C}}(c_1, \ldots, c_n)$$

for any choice of $c_1 \in \alpha^{-1}(a_1), \ldots, c_n \in \alpha^{-1}(a_n)$, and then by definition of $P_\oplus^{\mathcal{C}^\alpha}$:

$$P_\oplus^{\mathcal{A}}(a_1, \ldots, a_n) \Rightarrow P_\oplus^{\mathcal{C}^\alpha}(a_1, \ldots, a_n).$$

Similarly, if we have:

$$P_\ominus^{\mathcal{C}^\alpha}(a_1, \ldots, a_n)$$

then there is a choice of $c_1 \in \alpha^{-1}(a_1), \ldots, c_n \in \alpha^{-1}(a_n)$ such that $P^{\mathcal{C}}(c_1, \ldots, c_n)$ which implies $P_\ominus^{\mathcal{A}}(a_1, \ldots, a_n)$. $\square$

Proof (of Theorem 1). First, we prove that for any $\phi \in \mathbf{Sen}(\Sigma)$ and for any valuation $\nu : X \to \mathcal{C}$:

$$\mathcal{A}, \alpha\nu \models \phi_\oplus \Rightarrow \mathcal{C}, \nu \models \phi \text{ and } \mathcal{C}, \nu \models \phi \Rightarrow \mathcal{A}, \alpha\nu \models \phi_\ominus$$

by induction on the structure of ϕ. Wlog, we can consider only formulas containing no occurrence of $\vee$ nor of $\exists$.

- if ϕ is of the form $t_1 = t_2$ and if $\mathcal{A}, \alpha\nu \models \phi$, then $\alpha\nu(t_1) =_\oplus^\mathcal{A} \alpha\nu(t_2)$ and $\nu(t_1) = \nu(t_2)$. Now if $\mathcal{C}, \nu \models \phi$ then we have $\alpha\nu(t_1) = \alpha\nu(t_2)$ since $\nu(t_1) = \nu(t_2)$ and α is an homomorphism,
- if ϕ is of the form $Pt_1 \dots t_n$, and if $\mathcal{A}, \alpha\nu \models \phi_\oplus$, then:

  $$P_\oplus^\mathcal{A}(\alpha(\nu(t_1)), \dots, \alpha(\nu(t_n)))$$

  hence $P^\mathcal{C}(\nu(t_1), \dots, \nu(t_n))$ (because $\mathcal{C} \sqsubset_\alpha \mathcal{A}$) and $\mathcal{C}, \nu \models \phi$. Now if we suppose that $\mathcal{C}, \nu \models \phi$, then:

  $$P^\mathcal{C}(\nu(t_1), \dots, \nu(t_n))$$

  and, because $\mathcal{C} \sqsubset_\alpha \mathcal{A}$, $P_\ominus^\mathcal{A}(\alpha(\nu(t_1)), \dots, \alpha(\nu(t_n)))$ i.e., $\mathcal{A}, \alpha\nu \models \phi_\ominus$,
- if $\phi = \phi_1 \wedge \phi_2$ and $\mathcal{A}, \alpha\nu \models \phi_\oplus$ then $\mathcal{A}, \alpha\nu \models \phi_{1\oplus}$ and $\mathcal{A}, \alpha\nu \models \phi_{2\oplus}$ and by induction hypothesis $\mathcal{C}, \nu \models \phi_1$ and $\mathcal{C}, \nu \models \phi_2$, so $\mathcal{C}, \nu \models \phi$. Now if we suppose that $\mathcal{C}, \nu \models \phi$ then $\mathcal{C}, \nu \models \phi_1$ and $\mathcal{C}, \nu \models \phi_2$ and by induction hypothesis $\mathcal{A}, \alpha\nu \models \phi_{1\ominus}$ and $\mathcal{A}, \alpha\nu \models \phi_{2\ominus}$ and so $\mathcal{A}, \alpha\nu \models \phi_\ominus$,
- if $\phi = \neg\phi_1$ and $\mathcal{A}, \alpha\nu \models \phi_\oplus$ then by definition of $\neg\phi_{1\oplus}$ we have $\mathcal{A}, \alpha\nu \not\models \phi_{1\ominus}$. Then by induction hypothesis $\mathcal{C}, \alpha\nu \not\models \phi_1$ and $\mathcal{C}, \alpha\nu \models \phi$. Now if we suppose that $\mathcal{C}, \nu \models \phi$, then $\mathcal{C}, \nu \not\models \phi_1$ and by induction hypothesis $\mathcal{A}, \alpha\nu \not\models \phi_{1\oplus}$, so $\mathcal{A}, \alpha\nu \models \phi_\ominus$,
- if $\phi = \forall x : s.\ \phi_1$ and $\mathcal{A}, \alpha\nu \models \phi_\oplus$ then given any $c \in \mathcal{C}_s$ we have by induction hypothesis:

  $$\mathcal{C}, \nu[c/x] \models \phi_1$$

  and then $\mathcal{C}, \nu \models \phi$. Now if we suppose that $\mathcal{C}, \nu \models \phi$ and if $a \in \mathcal{A}_s$ then there exists $c \in \mathcal{C}_s$ such that $\alpha(c) = a$ (because α is an epimorphism) and $\mathcal{A}, (\alpha\nu)[a/x] \models \phi_1$ is a consequence of $\mathcal{C}, \nu[c/x] \models \phi_1$ using the induction hypothesis, so $\mathcal{A}, \alpha\nu \models \phi_\ominus$.

Now, if $\mathcal{A} \models \phi_\oplus$ and if $\nu : X \to \mathcal{C}$ is a valuation, then $\mathcal{A}, \alpha\nu \models \phi_\oplus$ and we can conclude that $\mathcal{C}, \nu \models \phi$, so $\mathcal{C} \models \phi$. When $\mathcal{C} \models \phi$ and $\nu' : X \to \mathcal{A}$ is a valuation, there is a valuation ν such that $\nu' = \alpha\nu$ (because α is an epimorphism) and then $\mathcal{C}, \nu \models \phi$ so $\mathcal{A}, \nu' \models \phi_\ominus$ and $\mathcal{A} \models \phi_\ominus$. □

Proof (of Lemma 1). Let $\mathcal{A}^\bullet$ denote $\mathcal{A}|_{|.|^\bullet_\Sigma}$. We recall that each predicate symbol P in Σ gives two predicate symbols $P_\oplus$ and $P_\ominus$ in $\Sigma^\bullet$ and these predicates give four predicate symbols $(P_\oplus)_\oplus$, $(P_\oplus)_\ominus$, $(P_\ominus)_\oplus$, and $(P_\ominus)_\ominus$ in $\Sigma^{\bullet\bullet}$. So, we just have to prove that:

$$(P_\oplus)_\oplus^{\mathcal{A}^\bullet} \subseteq P_\oplus^\mathcal{A} \subseteq (P_\oplus)_\ominus^{\mathcal{A}^\bullet} \quad (P_\ominus)_\oplus^{\mathcal{A}^\bullet} \subseteq P_\ominus^\mathcal{A} \subseteq (P_\ominus)_\ominus^{\mathcal{A}^\bullet}.$$

Using the definition of $|.|_\Sigma$ and $|.|^\bullet_\Sigma$ we have:

$$\begin{array}{l}
|(P_\oplus)_\oplus|^\bullet_\Sigma = |P_\oplus|_{\Sigma\oplus} = P_\oplus, \\
|(P_\oplus)_\ominus|^\bullet_\Sigma = |P_\oplus|_{\Sigma\ominus} = P_\ominus, \\
|(P_\ominus)_\oplus|^\bullet_\Sigma = |P_\ominus|_{\Sigma\oplus} = P_\oplus, \\
|(P_\ominus)_\ominus|^\bullet_\Sigma = |P_\ominus|_{\Sigma\ominus} = P_\ominus.
\end{array}$$

Then, using the assumption $P^\mathcal{A}_\oplus \subseteq P^\mathcal{A}_\ominus$:

$$\begin{array}{l}
(P_\oplus)^{\mathcal{A}^\bullet}_\oplus = P^\mathcal{A}_\oplus \subseteq P^\mathcal{A}_\ominus = (P_\oplus)^{\mathcal{A}^\bullet}_\ominus, \\
(P_\ominus)^{\mathcal{A}^\bullet}_\oplus = P^\mathcal{A}_\oplus \subseteq P^\mathcal{A}_\ominus = (P_\ominus)^{\mathcal{A}^\bullet}_\ominus.
\end{array}$$

□

Proof (of Theorem 2). Let $|.|$ denote the canonical homomorphism from $\Sigma^\bullet$ to Σ and $|.|^\bullet$ the associated homomorphism from $\Sigma^{\bullet\bullet}$ to $\Sigma^\bullet$. Then:

$$\begin{array}{rll}
\mathcal{A} \models \tau^1_\Sigma(\phi) & \Leftrightarrow \mathcal{A} \models \tau^1_\Sigma(|\phi_\oplus|) & \text{because } \phi = |\phi_\oplus| \\
& \Leftrightarrow \mathcal{A} \models |\tau^1_{\Sigma^\bullet}(\phi_\oplus)|^\bullet & \text{naturality} \\
& \Leftrightarrow \mathcal{A}|_{|.|^\bullet} \models \tau^1_{\Sigma^\bullet}(\phi_\oplus) & \\
& \Rightarrow \mathcal{A} \models \phi_\oplus & \text{correctness, using the above lemma.}
\end{array}$$

And for the second implication:

$$\begin{array}{rll}
\mathcal{A} \models \phi_\ominus & \Rightarrow \mathcal{A}|_{|.|^\bullet} \models \tau^2_{\Sigma^\bullet}(\phi_\ominus) & \text{correctness} \\
& \Rightarrow \mathcal{A} \models |\tau^2_{\Sigma^\bullet}(\phi_\ominus)|^\bullet & \\
& \Rightarrow \mathcal{A} \models \tau^2_\Sigma(|\phi_\ominus|) & \text{naturality} \\
& \Rightarrow \mathcal{A} \models \tau^2_\Sigma(\phi) & \text{because } |\phi_\ominus| = \phi.
\end{array}$$

Proof (of Lemma 2). For $i)$, it is sufficient to prove that if $\phi \in \mathbf{Sen}(\Sigma)$ is such that $\overline{\phi}$ contains no occurrence of a predicate eq_Σ, then $\overline{\tau_\Sigma(\phi)}$ contains no occurrence of either eq_{Σ^*} or $eq_{\Sigma\oplus}$. If ι denotes the signature inclusion of Σ into $\overline{\Sigma}$ we have by naturality of τ:

$$\overline{\tau_\Sigma(\phi)} = \tau_{\overline{\Sigma}}(\overline{\phi}) = \tau_{\overline{\Sigma}}(\iota(\phi)) = \iota^*(\tau_\Sigma(\phi))$$

and $\iota^*(\tau_\Sigma(\phi))$ can contain no occurrence of eq_{Σ^*} nor of $eq_{\Sigma\oplus}$ since they are not in $\iota(\Sigma)$. $ii)$ directly results from the definition of $\tau^{\mathbf{FOL}}$.

Proof (of Theorem 3). For the first implication, we have:

$$\begin{array}{rll}
\mathcal{A} \models \tau_\Sigma(\phi)_\oplus & \Leftrightarrow \overline{\mathcal{A}} \models \overline{\tau_\Sigma(\phi)_\oplus} & \\
& \Leftrightarrow \overline{\mathcal{A}} \models \tau^{\mathbf{FOL}}_{\overline{\Sigma}}(\overline{\phi})_\oplus & \text{by homogeneity} \\
& \Rightarrow \overline{\mathcal{A}} \models \overline{\phi}_\oplus & \text{applying optimality in } \mathbf{FOL} \\
& \Rightarrow \mathcal{A} \models \phi_\oplus &
\end{array}$$

and for the second one:

$$\begin{array}{rll}
\mathcal{A} \models \phi_\ominus & \Leftrightarrow \overline{\mathcal{A}} \models \overline{\phi_\ominus} & \\
& \Leftrightarrow \overline{\mathcal{A}} \models \overline{\phi}_\ominus & \\
& \Rightarrow \overline{\mathcal{A}} \models \tau^{\mathbf{FOL}}_{\overline{\Sigma}}(\phi)_\ominus & \text{applying optimality in } \mathbf{FOL} \\
& \Rightarrow \overline{\mathcal{A}} \models \overline{\tau_\Sigma(\phi)_\ominus} & \text{using homogeneity} \\
& \Rightarrow \mathcal{A} \models \tau_\Sigma(\phi)_\ominus &
\end{array}$$

$\square$

Proof (of Proposition 5). We prove that for every $c_1 \in \mathcal{C}'_{s_1}, \ldots, c_n \in \mathcal{C}'_{s_n}$ we have:

$$P^{\mathcal{C}'}(c_1, \ldots, c_n) \Rightarrow P_{\ominus}^{\mathcal{C}'}(\alpha(c_1), \ldots, \alpha(c_n))$$

by induction on the derivation of $P^{\mathcal{C}'}(c_1, \ldots, c_n)$. So assume there is a rule:

$$R = \frac{Pt_1^1 \ldots t_n^1 \quad \ldots \quad Pt_1^m \ldots t_n^m \quad \phi}{Pt_1^0 \ldots t_n^0} \quad \in \mathcal{R}$$

and a valuation $\nu : X \to \mathcal{C}'$ such that:

$$\begin{array}{l} P^{\mathcal{C}'}(\nu(t_1^1), \ldots, \nu(t_n^1)) \\ \vdots \\ P^{\mathcal{C}'}(\nu(t_1^m), \ldots, \nu(t_n^m)) \\ \mathcal{C}, \nu \models \phi \end{array}$$

and by induction hypothesis:

$$\begin{array}{l} P_{\ominus}^{\mathcal{A}'}(\alpha\nu(t_1^1), \ldots, \alpha\nu(t_n^1)) \\ \vdots \\ P_{\ominus}^{\mathcal{A}'}(\alpha\nu(t_1^m), \ldots, \alpha\nu(t_n^m)) \end{array}$$

then by Theorem 1, $\mathcal{A}, \alpha\nu \models \phi_{\ominus}$, so we can apply the rule $R_{\ominus} \in \mathcal{R}_{\ominus}$ and obtain $P_{\ominus}^{\mathcal{A}'}(\alpha\nu(t_1^0), \ldots, \alpha\nu(t_n^0))$. $\square$

References

1. E. Astesiano, M. Broy, and G. Reggio. Algebraic specification of concurrent systems. chapter 13 in [2], pages 467–520, 1999.
2. E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors. *Algebraic Foundations of Systems Specification*. Springer, 1999.
3. B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci and P. Schnoebelen. *Systems and Software Verification*. Springer, July 2001.
4. D. Bolignano. Towards a mechanization of cryptographic protocol verification. In *Proc. 9th Int. Conf. Computer Aided Verification (CAV'97), Haifa, Israel, June 1997*, volume 1254 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 1997.
5. D. Bolignano. Using abstract interpretation for the safe verification of security protocols. In *MFPS XV Mathematical Foundations of Programming Semantics, Fifteenth Conference, Tulane University, New Orleans, LA, April 28-May 1, 1999*, volume 20 of *Electronic Notes in Theor. Comp. Sci.* Elsevier Science, 1999.
6. E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Programming Languages (POPL'92), Albuquerque, New Mexico, January 19–22, 1992*, pages 343–354. ACM Press, 1992.

7. Cofi: The common framework initiative for algebraic specification and development. http://www.brics.dk/Projects/CoFI/.
8. P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2–3):103–179, 1992.
9. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. In *ACM Transactions on Programming Languages ans Systems*, volume 19, 1997.
10. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
11. J. Goubault-Larrecq. A method for automatic cryptographic protocol verification (extended abstract). In *Proc. 15th IPDPS 2000 Workshops, Cancun, Mexico, May 2000*, volume 1800 of *Lecture Notes in Computer Science*, pages 977–984. Springer, 2000.
12. S. Merz. Rules for abstraction. In R. K. Shyamasundar and K. Ueda, editors, *Advances in Computing Science (ASIAN'97)*, volume 1345 of *LNCS*, pages 32–45, Berlin, Germany, 1997. Springer.
13. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. Computer Security*, 6:85–128, 1998.
14. A. Tarlecki. Towards heterogeneous specifications. In *Proc. Workshop on Frontiers of Combining Systems FroCoS'98, Amsterdam, October 1998*, Applied Logic Series. Kluwer Academic Publishers, 1998. (To appear).