Towards Verification of Security-Aware E-services

Silvio Ranise

Dipartimento di Informatica
Università di Verona, Italy

Abstract. We study an extension of the relational transducers introduced by Abiteboul, Vianu, Fordham, and Yesha, which are capable of specifying transaction protocols and their interplay with security constraints. We investigate the decidability of relevant verification problems such as goal reachability and log validation.

1 Introduction

Web-services providing e-commerce capabilities to support business transactions between several parties over the Internet are more and more widespread. The development of such services involves several issues involving security, authentication, and the design of business models to name a few.

Relational transducers have been proposed [17] to model the transaction behavior of e-services for e-commerce and to allow for the analysis of the underlying transaction protocols. Roughly, a relational transducer is a machine capable of translating an input sequence of relations into an output sequence of relations; its state being a relational database. A particular class of relational transducers, called semi-positive cumulative (spocus) transducers, has been identified and studied because its specification is declarative and simple to understand, and some interesting verification problems turn out to be decidable (see again [17]). The input-output mapping in spocus transducers is implicitly defined by a set of non-recursive (variant of) Datalog rules. This is based on the observation that any set of Datalog rules defines a query which is a (partial) function from relational databases (the input relations of the transducers) to answer relations (the output relations of the transducer), see, e.g., [16].

Although spocus transducers support declarative specifications and some interesting verification problems are decidable, one of their major shortcomings is the lack of support for the specification and verification of security requirements. In particular, the class of security requirements pertaining to authorization play a crucial role in several business transactions. For example, failure to meet authorization constraints may cause economic losses and may even have legal implications. In this paper, we propose an extension of the spocus transducers that overcome this problem. In particular, following [12], we extend the rules for input-output in spocus transducers with constraint Datalog rules to formalize access policy statements. This allows us to develop our ideas in the framework of

first-order logic and to adapt and reuse specification techniques belonging to the line of research which uses extensions of Datalog to express policy statements (see, e.g., [7, 2, 11]). Technically, the situation is more complex than with spocus transducers as certain patterns in authorization policies (typically, delegation) require recursion (see, e.g., [3]). The (variant of) Datalog rules used in [17] for spocus transducers were assumed to be non-recursive.

Plan of the paper. Section 2 introduces some background notions on first-order logic, relational databases, and Datalog rules. Section 3 extends spocus transducers with (constraint) Datalog rules for access policy specifications and introduces two verification problems. In Section 3.1, a simple (yet representative) business process specified by means of a policy-aware transducer illustrates the need to have recursive Datalog rules and a first-order theory to be able to correctly specify delegation and trust relationships, respectively, which are relevant to capture important business rules. As the verification problems can be reduced to satisfiability problems for a certain class of formulae, Section 4 defines such a class and proves the decidability of the satisfiability problem under suitable assumptions (for lack of space, proofs are in the extended version [13] of this paper) and Section 5 shows how to reduce verification to satisfiability. In particular, Section 5.1 discusses the hypotheses under which the fix-point computation of the query induced by a set of constraint Datalog rules terminates. Section 6 concludes.

2 Preliminaries

We assume familiarity with the basic syntactic and semantic notions of first-order logic (see, e.g., [8]). We work in first-order logic with equality and consider the equality symbol $=$ as a logical constant. A *constraint* is a (finite) conjunction of literals. An *expression* is a term, an atom, a literal, or a formula. A $\Sigma(\underline{x})$-*expression* is an expression built out of the symbols in a signature Σ where at most the variables in the sequence $\underline{x}$ may occur free, and we write $E(\underline{x})$ to emphasize that E is a $\Sigma(\underline{x})$-expression. (Below, by abuse of notation, we consider sequences also as finite sets and use the standard set-theoretic operations to combine them.)

Let Σ be a signature. A Σ-*theory* T is a set of Σ-sentences. In this paper, we assume that all theories are consistent (i.e. they admit at least one model). A Σ-formula $\varphi(\underline{x})$ is T-satisfiable iff there exists a model $\mathcal{M}$ of T (i.e. $\mathcal{M} \models T$) such that $\mathcal{M} \models \exists \underline{x}.\varphi(\underline{x})$. The *satisfiability modulo theory* T (in symbols, SMT(T)) *problem* consists of establishing the T-satisfiability of any quantifier-free Σ-formula. A formula $\varphi(\underline{x})$ is T-*valid* if for every model $\mathcal{M}$ of T, we have $\mathcal{M} \models \forall \underline{x}.\varphi(\underline{x})$. Two formulae φ and ψ are T-*equivalent* iff the formula $\varphi \leftrightarrow \psi$ is T-valid. A theory T admits *quantifier elimination* if for an arbitrary formula $\varphi(\underline{x})$ (possibly containing quantifiers), it is possible to compute a T-equivalent quantifier-free formula $\varphi'(\underline{x})$. A Σ-theory T is *locally finite* if Σ is finite and, for every set of constants $\underline{a}$, there are finitely many ground terms $t_1, ..., t_{k_{\underline{a}}}$, called *representatives*, such that for every ground $(\Sigma \cup \underline{a})$-term u, we have $T \models u = t_i$

for some i. If the representatives are effectively computable from $\underline{a}$ and t_i is computable from u, then T is *effectively* locally finite. For simplicity, we will often say "locally finite" to mean "effectively locally finite."

We consider the *Bernays-Schönfinkel-Ramsey* (BSR) class, also called Effectively Propositional Logic, whose satisfiability problem is well-known to be decidable. A formula of the BSR class is of the form $\exists \underline{x}.\forall \underline{y}.\varphi(\underline{x},\underline{y})$, where $\underline{x},\underline{y}$ are (disjoint) tuples of variables and φ is a quantifier-free formula built out of a signature containing only relation and constant symbols (i.e. no function symbol occurs in φ).

A *finite instance of (a n-ary relation)* r is a formula of the form

$$\forall \underline{x}.r(\underline{x}) \leftrightarrow \bigvee_{j=1}^{m} \underline{x} = \underline{c}^j,$$

where $\underline{x} = x_1, ..., x_n$ is a sequence of variables, $\underline{c}^j = c_1^j, ..., c_n^j$ is a sequence of constants (for $j = 1, ..., m$), and $\underline{x} = \underline{c}^j$ abbreviates $(x_1 = c_1^j \wedge \cdots \wedge x_n = c_n^j)$. Let $\underline{R}$ be a finite set of predicate symbols, a *database over* $\underline{R}$ is a conjunction of finite instances of $r \in \underline{R}$; we sometimes call $\underline{R}$ a *database schema*.

A *Datalog* formula is a BSR formula of the form

$$\forall \underline{x},\underline{y}. \bigwedge_{i=1}^{n} A_i(\underline{x},\underline{y}) \rightarrow A_0(\underline{x}) \text{ also written as a } \textit{rule } A_0(\underline{x}) \leftarrow \bigwedge_{i=1}^{n} A_i(\underline{x},\underline{y}),$$

where A_i is an atom (for $i = 0, 1, ..., n$) and $\underline{x},\underline{y}$ are disjoint tuples of variables. Furthermore, A_0 is said the *head*, $\bigwedge_{i=1}^{n} A_i(\underline{x},\underline{y})$ the *body* of the rule, and when $n = 0$, the Datalog rule is also called a *fact*. A set of Datalog rules is *semi-positive* if whenever a negative literal appears in the body of a rule, then the relation symbol of this literal is not the relation symbol of any literal which is the head of a non-trivial rule in the set. Another generalization of Datalog rules is that of *constraint Datalog rule* where, for some Σ-theory T, one quantifier-free $\Sigma(\underline{x},\underline{y})$-formula may appear in its body. A set of Datalog (semi-positive or constraint Datalog) rules is *recursive* if some predicate symbol occur both in the heads and the bodies of the rules; otherwise, we say it to be *non-recursive*.

3 Security-Aware Transducers

For the rest of the paper, we fix a Σ-theory T and a relational signature $\underline{R} := \mathit{In} \cup \mathit{Out} \cup \mathit{DB} \cup \mathit{Policy}$ such that $X \cap Y = \emptyset$ and $X \cap \Sigma = \emptyset$ for $X, Y \in \{\mathit{In}, \mathit{Out}, \mathit{DB}, \mathit{Policy}\}$.

Definition 1. *A* (cumulative and policy-aware) transducer (over $(\Sigma, \underline{R})$) *is a tuple* $(\underline{past}, \tau, \omega, \pi)$ *where*

- $\underline{past} = past_{i_1}, ..., past_{i_n}$ *for* $\mathit{In} = \{r_{i_1}, ..., r_{i_n}\}$ *is a sequence of fresh predicate symbols, i.e.* $\underline{past} \cap (\Sigma \cup \underline{R}) = \emptyset$,

– *given a finite database over the database schema* $I \subseteq In$ *of the form*

$$\bigwedge_{r_i \in I} (\forall \underline{x}.r_i(\underline{x}) \leftrightarrow \bigvee_{j=1}^{n_i} \underline{x} = \underline{c}_i^j),$$

where c_i^j *are constant symbols,* $\tau(I)$ *is a finite set of* (cumulative) transitions *of the form*

$$\forall \underline{x}.past'_{r_i}(\underline{x}) \leftrightarrow \bigvee_{j=1}^{n_i} \underline{x} = \underline{c}_i^j \vee past_{r_i}(\underline{x}),$$

for each $r_i \in I$ *and*

$$\forall \underline{x}.past'_{r_i}(\underline{x}) \leftrightarrow past_{r_i}(\underline{x}),$$

for each $r_i \in In \setminus I$*. Let* τ *be the set of cumulative transitions for every finite instance over every possible sub-set of* In*;*

– ω *is a a finite set of semi-positive non-recursive rules of the form* $A_0 \leftarrow \bigwedge_{i=1}^n A_i$ *such that (i)* A_0 *is an Out-atom, (ii)* A_i *is a* $(In \cup \underline{past} \cup DB)$*-literal (for* $i = 1, ..., n$*), and (iii) every variable appearing in a rule must occur in at least one positive literal;*

– π *is a finite set of (possibly recursive) constraint Datalog rules of the form* $A_0 \leftarrow \bigwedge_{i=1}^n A_i \wedge \psi$ *such that (i)* A_0 *is a Policy-atom, (ii)* A_i *is a* $(DB \cup Policy \cup \underline{past})$*-atom, for* $i = 1, ..., n$*, (iii)* ψ *is a quantifier-free* Σ*-formula, and (iv) the set of variables occurring in* A_0 *are a sub-set of the variables occurring in* $\bigwedge_{i=1}^n A_i$*.*

To understand why what we have just defined is a transducer, recall that (constraint) Datalog rules define queries on databases; which, in turn, can be seen as mappings associating a so-called answer relation to each database (see, e.g., [6] for details). In this perspective, we can consider the union of (non-recursive) semi-positive rules in ω with the constraint (Datalog) rules in π as defining a mapping between databases over $In \cup \underline{past} \cup DB$ to databases over $Out \cup \underline{past}$. By taking $Policy = \emptyset$ and T to be the empty theory, it is easy to see that our notion of transducer reduces to that of relational Spocus transducer in [17]. Notice that the set π may contain recursive (constraint) Datalog rules. Recursion is crucial to express important mechanisms in policy management such as delegation. So, although, recursion complicates the reduction of verification problems for cumulative and policy-aware transducers to logical satisfiability problems (see Section 5 below), it is of paramount importance for naturally expressing several patterns of policy management.

Below, given a cumulative and policy-aware transducer $(\underline{past}, \tau, \omega, \pi)$ over $(\Sigma, \underline{R})$ and a $(\Sigma \cup \underline{R} \cup \underline{past})$-formula φ, the formula obtained by replacing each symbol $s \in In \cup Out \cup \underline{past}$ occurring in φ with a fresh symbol s^j and each symbol $p' \in \underline{past'}$ with fresh symbols p^{j+1} will be denoted by $ren(\varphi, j)$ for $j \geq 0$.

Definition 2. *Let* $(\underline{past}, \tau, \omega, \pi)$ *be a cumulative and policy-aware transducer over* $(\Sigma, \underline{R})$ *and db be a finite instance over DB. The sequence* $\mathcal{I}_1, \mathcal{O}_1; ...; \mathcal{I}_n, O_n$ *is a* (finite) run (with length n) *of* $(\underline{past}, \tau, \omega, \pi)$ *iff the formula*

$$\begin{array}{l} db \wedge \bigwedge_{p \in \underline{past}} \forall \underline{x}.p^0(\underline{x}) \leftrightarrow false \wedge \\ \bigwedge_{j=1}^{n} ren(\overline{\mathcal{I}_j}, j) \wedge ren(\tau(\mathcal{I}_j), j) \wedge ren(\omega, j) \wedge ren(\pi, j) \wedge ren(\mathcal{O}_j, j) \end{array} \quad (1)$$

is T*-satisfiable, where* $\mathcal{I}_1, ..., \mathcal{I}_n$ *is a finite sequence where each* $\mathcal{I}_j$ *is an instance over In, for* $j = 1, ..., n$*, called the* input sequence*, and* $\mathcal{O}_1, ..., \mathcal{O}_n$ *is a finite sequence where each* $\mathcal{O}_j$ *is an instance over Out, for* $j = 1, ..., n$*, called the* output sequence*.*

Given a cumulative and policy-aware transducer $(\underline{past}, \tau, \omega, \pi)$ over $(\Sigma, \underline{R})$ and a $(\Sigma \cup \underline{R} \cup \underline{past})$-formula φ, $\mathcal{R}_1; ...; \mathcal{R}_n$ be a run of the transducer, and $Log \subseteq (In \cup Out)$; then, $prj(Log, \mathcal{R}_1; ...; \mathcal{R}_n)$ is a finite sequence of formulae obtained from $\mathcal{R}_1; ...; \mathcal{R}_n$ by forgetting all those finite instances over $(In \cup Out) \setminus Log$. A *goal* is a BSR formula of the form

$$\exists \underline{x}. \bigwedge_{j=1}^{n} A_j(\underline{x})$$

where A_j is an Out-literal, for $j = 1, ..., n$.

Definition 3. *Let* $(\underline{past}, \tau, \omega, \pi)$ *be a cumulative and policy-aware transducer over* $(\Sigma, \underline{R})$ *and* $Log \subseteq In \cup Out$*. We define the following two verification problems for* $(\underline{past}, \tau, \omega, \pi)$*:*

Goal reachability: *does a given goal* γ *hold in the last output of some run of* $(\underline{past}, \tau, \omega, \pi)$*?*

Log validity: *given a finite sequence* $\mathcal{L}_1, ..., \mathcal{L}_n$ *where each* $\mathcal{L}_j$ *is a finite instance over Log for* $j = 1, ..., n$*, does there exist a finite run* $\mathcal{S}_1; ...; \mathcal{S}_n$ *of* $(\underline{past}, \tau, \omega, \pi)$ *such that* $\mathcal{L}_1, ..., \mathcal{L}_n = prj(Log, \mathcal{S}_1; ...; \mathcal{S}_n)$*?*

Goal reachability consists of checking whether a set of goals can be reached by some run of the transducer. This verification problem is a first sanity check on the design of the business model underlying the transducer as the latter is usually conceived to reach a certain goal, e.g., delivering a product provided that certain conditions are met. Log validation consists of checking whether a given log sequence can be generated by some input sequence. This problem arises, for example, when the transducer of a supplier is allowed to run on a customer's site for efficiency or convenience. The trace provided by the log allows the supplier to validate the transaction carried out by the customer.

3.1 Example

We consider a business model where a customer wants to buy a book, is billed for it, pays, and then takes delivery and a discount voucher for his/her next

purchase if he/she is a preferred customer and is an employee of an accredited company. One kind of preferred customers are those affiliated to a organization EOrg, which issues credentials to certify that a person is one of its members. The business process has a database for accredited companies which are trusted by the business model to pass to other companies the fact of being accredited as well as the possibility of declaring other companies being accredited. The identification of an employee of a company is done by issuing a certificate signed by the company. Companies are organized according to a certain hierarchy which, for the purpose of this paper, can be assumed to be a partial order. The customer, willing to get a discount voucher should provide suitable valid credentials that he/she is a preferred customer (e.g., a member of EOrg) and an employee of an accredited company.

We need the theory of partial orders T_{po} to specify the trust relationship between companies. The signature Σ_{po} of T_{po} consists of the binary relation $\mathsf{is\_trusted\_by}$ (written infix) and its axioms are the following three sentences:

$\forall x, y, z.(x \ \mathsf{is\_trusted\_by}\ y \wedge y \ \mathsf{is\_trusted\_by}\ z \rightarrow x \ \mathsf{is\_trusted\_by}\ z)$,
$\forall x. x \ \mathsf{is\_trusted\_by}\ x$, and $\forall x, y.(x \ \mathsf{is\_trusted\_by}\ y \wedge y \ \mathsf{is\_trusted\_by}\ x \rightarrow x = y)$.

Intuitively, x $\mathsf{is\_trusted\_by}$ y means that company y trusts company x with respect to the fact of being accredited and the possibility of delegating this capability.

The business model can be formalized by a policy-aware transducer (whose policies use the theory T_{po}) as follows. We fix the following relational signature $\underline{R} := \mathit{In} \cup \mathit{Out} \cup \mathit{DB} \cup \mathit{Policy}$, where $\mathit{In} := \{\mathit{order}, \mathit{pay}, \mathit{eorg}, \mathit{emplcert}\}$, $\mathit{Out} := \{\mathit{sendbill}, \mathit{deliver}, \mathit{sendvoucher}\}$, $\mathit{DB} := \{\mathit{price}, \mathit{available}, \mathit{accredited}\}$, $\mathit{Policy} := \{\mathit{preferred}, \mathit{employee}, \mathit{employeeof}\}$.

The business process has three databases *price*, *available*, and *accredited* storing the prices of books, their availability, and the set of (root) accredited companies, respectively. A customer interacts with the business system by inserting tuples in four input relations, *order*, *pay*, *eorg*, and *emplcert*. The system responds by producing output relations *sendbill*, *deliver*, and *sendvoucher* and it keeps track of the history of the business transaction using the state relations: $\mathit{past}_{\mathit{order}}$, $\mathit{past}_{\mathit{pay}}$, $\mathit{past}_{\mathit{eorg}}$, and $\mathit{past}_{\mathit{emplcert}}$. The state of the transducer cumulatively adds the tuples inserted into the input relations. The output rules ω of the transducer are the following:

$$\begin{aligned}
\mathit{sendbill}(X,Y,Z) &\leftarrow \mathit{order}(X,Z) \wedge \mathit{price}(X,Y) \wedge \neg \mathit{past}_{\mathit{pay}}(X,Y,Z) \\
\mathit{deliver}(X,Z) &\leftarrow \mathit{past}_{\mathit{order}}(X,Z) \wedge \mathit{price}(X,Y) \wedge \\
&\qquad \mathit{pay}(X,Y,Z) \wedge \neg \mathit{past}_{\mathit{pay}}(X,Y,Z) \\
\mathit{sendvoucher}(Z) &\leftarrow \mathit{past}_{\mathit{deliver}}(X,Z) \wedge \mathit{preferred}(Z) \wedge \mathit{employee}(Z)\,.
\end{aligned}$$

The first rule governs the sending of a bill whose amount is Y about a book X to a customer Z when an order is placed on X by Z, the price of X is Y, and Z has not already been paid for X. The second rule enables the delivery of a book X to Z if X has been ordered by Z and it is being paid the correct price by Z. Finally, the last rule is about sending the discount voucher to Z if a book has

input sequence	$order(\mathsf{Book}_1, \mathsf{Alice})$ $order(\mathsf{Book}_2, \mathsf{Bob})$ $eorg(\mathsf{Alice})$	$pay(\mathsf{Book}_1, 8, \mathsf{Alice})$ $pay(\mathsf{Book}_2, 15, \mathsf{Bob})$ $emplcert(\mathsf{Alice}, \mathsf{Comp}_3)$	$eorg(\mathsf{Bob})$
output sequence	$sendbill(\mathsf{Book}_1, 8, \mathsf{Alice})$ $sendbill(\mathsf{Book}_2, 15, \mathsf{Bob})$	$deliver(\mathsf{Book}_1, \mathsf{Alice})$ $deliver(\mathsf{Book}_2, \mathsf{Bob})$	$sendvoucher(\mathsf{Alice})$

Table 1. A run of a policy-aware transducer

been delivered to Z and this last is both a preferred customer and an employee of an accredited company. Finally, the policy rules π of the transducer are the following:

$$\begin{aligned} preferred(Z) &\leftarrow past_{eorg}(Z) \\ employee(Z) &\leftarrow past_{emplcert}(Z,U) \wedge employeeof(Z,U) \\ employeeof(Z,F) &\leftarrow accredited(F) \\ employeeof(Z,U) &\leftarrow employeeof(Z,F) \wedge U \text{ is\_trusted\_by } F \end{aligned}$$

The first rule simply establishes whether Z is a preferred customer by checking if the credential saying that Z is an EOrg member has been presented. (There can be other rules of this kind once the business model establishes that customers having a certain affiliation become preferred customers.) The last three rules check if Z has presented a certificate saying that he is an employee of a company which is in the suitable trust relationship with some accredited company. Notice the use of recursion to express delegation of the capability of certifying the fact of being an employee of an accredited company to some of its units. Since *employeeof* is a recursive predicate, $\omega \cup \pi$ is not a set of non-recursive semi-positive rules and, hence, the associated transducer is not spocus; all the techniques developed in [17] cannot be used to investigate verification problems for this business process. Also the results in [14] cannot be used here as the rules considered in that work (which allow for non-cumulative transducers) are assumed to be non-recursive. An interesting line of future work is to allow for non-cumulative rules also for the policy-aware transducers considered in this paper (i.e. in presence of recursive rules for policy specification).

Table 1 shows an example of a run for the transducer specified above. We have assumed that $price(\mathsf{Book}_1, 8)$, $price(\mathsf{Book}_2, 15)$, $accredited(\mathsf{Comp}_1)$, Comp_3 is_trusted_by Comp_2, and Comp_2 is_trusted_by Comp_1 . Now, if we take $Log = Out$, then the instance of the log validity problem for the run in Figure 1 reduces to the checking T_{po}-satisfiability of the following formula (only an excerpt is shown for the sake of conciseness):

$$\begin{pmatrix} \forall x,y.price(x,y) \leftrightarrow ((x = \mathsf{Book}_1 \wedge y = 8) \vee (x = \mathsf{Book}_2 \wedge y = 15)) \wedge \\ \forall x.accredited(x) \leftrightarrow x = \mathsf{Comp}_1 \qquad\qquad\qquad\qquad\qquad\qquad \wedge \\ (\mathsf{Comp}_3 \text{ is\_trusted\_by } \mathsf{Comp}_2) \wedge (\mathsf{Comp}_2 \text{ is\_trusted\_by } \mathsf{Comp}_1) \end{pmatrix} \wedge$$

$$\begin{pmatrix} \forall x,y.past^0_{order}(x,y) \leftrightarrow false \wedge \forall x,y,z.past^0_{pay}(x,y,z) \leftrightarrow false \wedge \\ \forall x.past^0_{eorg}(x) \leftrightarrow false \wedge \forall x,y.past^0_{emplcert}(x,y) \leftrightarrow false \end{pmatrix} \wedge$$

$$\forall x, y.(order^0(x, y) \leftrightarrow (x = \mathsf{Book_1} \wedge y = \mathsf{Alice}) \vee (x = \mathsf{Book_2} \wedge y = \mathsf{Bob})) \wedge$$
$$\forall x, y.(eorg^0(x) \leftrightarrow x = \mathsf{Alice} \wedge$$
$$\forall x, y, z.(order^0(x, z) \wedge price^0(x, y) \wedge \neg past^0_{pay}(x, y, z) \rightarrow sendbill^0(x, y, z)) \wedge ...$$

From the piece formula above, it is not difficult to see that the output sequence (at time instant 0) of the transducer is exactly the one depicted in the first column, second line of Table 1.

4 An extension of the BSR class

Since the goal reachability and the log validity problems will be reduced to the satisfiability of a certain class of first-order formulae (see Section 5), we define such a class of formulae and study their decidability.

Let T be Σ-theory, we are interested in studying the T-satisfiability of formulae of the form

$$\exists \underline{x}.\forall \underline{y}.\varphi(\underline{x}, \underline{y})$$

where φ is a quantifier-free $\Sigma^{\underline{R}}$-formula and $\Sigma^{\underline{R}} := \Sigma \cup \underline{R}$ such that $\Sigma \cup \underline{R} = \emptyset$. Sentences of this form are called $BSR(\Sigma^{\underline{R}})$-sentences. If T is the empty theory, then $BSR(\Sigma^{\underline{R}})$-sentences are BSR formulae.

We show the decidability of $BSR(\Sigma^{\underline{R}})$-sentences under suitable hypotheses on the theory T in two steps. First, we eliminate universal quantifiers by identifying finitely many instances which are sufficient for (un-)satisfiability checking (under suitable hypotheses). Second, we show the decidability of the resulting quantifier-free formulae.

Lemma 1 (Instantiation). *Let T be a locally finite Σ-theory and the class of models of T be closed under sub-structures. Furthermore, let $\underline{R}$ be a finite set of predicate symbols such that $\Sigma \cap \underline{R} = \emptyset$. The $BSR(\Sigma^{\underline{R}})$-sentence*

$$\exists \underline{x}.\forall \underline{y}.\varphi(\underline{x}, \underline{y})$$

is satisfiable iff it is satisfiable in a finite model iff the quantifier-free formula

$$\exists \underline{x}. \bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma)$$

is satisfiable, where σ ranges over the substitutions mapping the variables $\underline{y}$ into the set of representative $\Sigma(\underline{x})$-terms and $\underline{y}\sigma$ denotes the simultaneous application of σ to the variables of the tuple $\underline{y}$.

We are left with the problem of showing the decidability of the satisfiability of quantifier-free formulae over $\Sigma^{\underline{R}}$.

Theorem 1 (Decidability). *Let T be a locally finite Σ-theory whose class of models is closed under sub-structures and the SMT(T) problem be decidable; $\underline{R}$ be a finite set of predicate symbols such that $\Sigma \cap \underline{R} = \emptyset$. Then, the satisfiability of $BSR(\Sigma^{\underline{R}})$-sentences is decidable.*

The assumption of the background theory T to be locally finite, at first, seems to be quite restrictive. However, the constraint domains considered as relevant for trust management in [12] have quite simple algebraic structures so as to allow for efficient algorithm to answer policy access queries. We believe that locally finite theories can be put to productive use for the verification of policy-aware transducers as suitable abstractions of this class of constraint domains.

5 Decidability of goal reachability and log validity

Before describing the reduction of the goal reachability and the log validity problems to the satisfiability of the extension of the BSR class considered in the previous section, we re-consider how semi-positive (non-recursive) rules and (constraint) Datalog rules define queries on databases, or, equivalently, mapping of databases to answer relations. As already observed in Section 3, this is crucial to characterize the input-output behavior of the class of relational transducers considered in this paper.

5.1 Constraint Datalog queries

A *database query* is a mapping associating to each database an (answer) relation. A set of (constraint) Datalog rules can be seen as a way to implicitly define queries as follows. Let T be a Σ-theory admitting quantifier-elimination and $\underline{R}$ be a database schema such that $\Sigma \cap \underline{R} = \emptyset$. Let $\Pi = \omega \cup \pi$ be a finite set of rules such that $\underline{R} = \underline{B} \cup \underline{E}$ for some cumulative and policy-aware transducer $(\underline{past}, \tau, \omega, \pi)$, where $\underline{B}$ is the set of predicate symbols occurring only in the body of the rules in Π and not in their heads, and $\underline{E}$ is the set of predicate symbols occurring both in the body and head of the rules in Π. Consider a rule $\rho \in \Pi$ of the form $e(\underline{x}) \leftarrow \bigwedge_{i=1}^{n} b_i(\underline{x}, \underline{y}) \wedge \psi(\underline{x}, \underline{y})$ where $e \in \underline{E}$, the predicate symbol of b_i is in $\underline{R}$ (for $i = 1, ..., n$), and ψ is a quantifier-free $\Sigma(\underline{x}, \underline{y})$-formula, the *constraint query associated to* ρ is the formula

$$\exists \underline{y}. \bigwedge_{i=1}^{n} b_i(\underline{x}, \underline{y}) \wedge \psi(\underline{x}, \underline{y}).$$

If b_i contains only symbols in $\underline{B}$ (as it is the case of the semi-positive non-recursive rules in ω), then it is possible to replace each occurrence of such symbols with disjunctions of conjunctions of equalities (recall the definition of finite instance in Section 2). In this way, the resulting (existentially quantified) formula turns out to be a $\Sigma(\underline{x}, \underline{y})$-formula and the decidability result above (i.e. Theorem 1) can be used. This is the key insight used in reducing both log validity and goal reachability to the satisfiability of BSR formulae in [17] which in our framework corresponds to the case there $\pi = \emptyset$. However, this is not enough for the class of transducers considered in this paper because of the presence of the constraint Datalog rules in Π which may be recursive (we have already argued that this complication is necessary to be able to express certain patterns

```
function constraintFixPoint(F : constraint facts,  R : constraint Datalog rules)
1   results ⟵ F; Changed ⟵ true;
2   while Changed do
3       Changed ⟵ false
4       foreach rule ∈ R do
5           foreach tuple of constraint facts constructed from results do
6               newres ⟵ constraint facts obtained by
                                constraint rule application between rule and tuple
7               foreach fact ∈ newres do
8               if (results ⊭ fact) then results ⟵ results ∪ {fact};
9               Changed ⟵ true;
10              end
11          end
12      end
13  end
14  return results
```

Fig. 1. Least fix-point computation of constraint Datalog rules

of policy management such as delegation). As a consequence, we need a fix-point characterization for the queries associated to a set Π of possible recursive rules. To this end, we proceed as follows. First, we regard each finite instance of $r \in In \cup Out \cup DB$ as a constraint fact, i.e. $\forall r(\underline{x}) \leftrightarrow \bigvee_{j=1}^{m} \underline{x} = \underline{c}^j$ is transformed into the Datalog rule $r(\underline{x}) \leftarrow \bigvee_{j=1}^{m} \underline{x} = \underline{c}^j$, where $\underline{x}$ is a tuple of variables and $\underline{c}^j$ is a tuple of constants, for $j = 1, ..., m$. Let $r_0(\underline{x}) \leftarrow \bigwedge_{i=1}^{n} r_i(\underline{x}) \wedge \psi_0(\underline{x})$ be a constraint Datalog rule in Π for $r_i \in \underline{R}$ and $i = 0, 1, ..., n$ and $r_i(\underline{x}_{k_i}) \leftarrow \psi_i(\underline{x}_{k_i})$ be a constraint fact for ψ_i be a $\Sigma(\underline{x}_i)$-quantifier-free formula, k_i the arity of r_i, and $i = 1, ..., n$. A *constraint rule application* produces $m \geq 0$ facts of the form $r_0(\underline{x}) \leftarrow \psi'_j(\underline{x})$ where ψ'_j is a quantifier-free $\Sigma(\underline{x})$-formula for $j = 1, ..., m$ $(m \geq 0)$ and $\bigvee_{j=1}^{m} \psi'_j$ is equivalent (by the elimination of quantifiers in T) to the formula

$$\exists \underline{y}.(\bigwedge_{i=1}^{n} \psi_i(\underline{x}_{k_i}) \wedge \psi_0(\underline{x})),$$

where $\underline{y}$ is the tuple of variables occurring in the body of the rule but not in the head. The least fix-point of a set of constraint Datalog rules can be computed by the algorithm in Figure 1. The function constraintFixPoint terminates when all derivable new facts are implied by previously derived facts.[1] The requirement

[1] The test at line 8 can be reduced (by eliminating quantifiers) to a T-satisfiability test on a conjunction of two quantifier-free Σ-formulae. To understand how, consider two facts $r_1(\underline{x}) \leftarrow \psi_1(\underline{x})$ and $r_2(\underline{y}) \leftarrow \psi_2(\underline{y})$. To check that latter is a logical consequence of the former (modulo T), it is sufficient to check the T-validity of $\forall \underline{x}.\psi_1(\underline{x}) \rightarrow \forall \underline{y}.\psi_2(\underline{y})$. This, reasoning by refutation, is equivalent to check the T-unsatisfiability of the negation of the previous formula, which, by eliminating quantifiers can be reduced to a quantifier-free formula.

that T admits elimination of quantifiers is not sufficient to guarantee termination of the algorithm in Figure 1. We need the following notion which is adapted from [16].

Definition 4. *Let T be a Σ-theory and $\underline{x}$ be a finite set of variables. If for every (possibly infinite) T-satisfiable set S of $\Sigma(\underline{x})$-constraint, there exists a finite subset $S_{fin} \subseteq S$ such that for every ψ in S, there exists a ψ' in S_{fin} for which $\forall \underline{x}.\psi(\underline{x}) \rightarrow \psi'(\underline{x})$ is T-valid, then T is said to be* constraint compact. *If S_{fin} is effectively computable from S, then T is said to be* effectively constraint compact *(in the following, when we say 'constraint compact', we in fact always mean 'effectively constraint compact').*

Constraint compactness is a sufficient condition for the termination of the algorithm in Figure 1.

Theorem 2 ([16]). *Let T be a constraint compact theory. Then, the algorithm in Figure 1 terminates for every query.*

The sketch of the proof is as follows. By contradiction, assume that the algorithm does not terminate. Thus, every iteration of the loop produces at least one fact which is not a consequence of those which were already derived. Since there are only finitely many different predicate symbols, there must be at least one such symbol for which the algorithm derives an infinite set S of facts. But, T is assumed to be constraint compact and so we can replace S with a finite subset S_{fin} such that the facts in $S \setminus S_{fin}$ are consequence of those in S_{fin}. As the algorithm checks for the logical consequence of newly derived facts before adding it to the resulting set of derived facts, all the facts in $S \setminus S_{fin}$ should not be considered: a contradiction.

Interestingly, locally finite theories are also constraint compact.

Proposition 1. *If T is an (effectively) locally finite theory, then it is also (effectively) constraint compact.*

Proof. Let S be a set of constraints over a finite set $\underline{x}$ of variables. Then, since the theory T is locally finite, there exists a finite set *RTerms* of $\Sigma(\underline{x})$-terms such that for every term u in S, there exists $t \in$ *RTerms*, $T \models u = t$. By using the terms in *RTerms* and the finite signature Σ, it is possible to build only finitely many distinct $\Sigma(\underline{x})$-atoms $\psi_1(\underline{x}), ..., \psi_n(\underline{x})$ (also called representative atoms) such that for any further $\Sigma(\underline{x})$-atom $\psi(\underline{x})$, there exists $i \in \{1, ..., n\}$, $T \models \forall \underline{x}.\psi_i(\underline{x}) \leftrightarrow \psi(\underline{x})$. If we consider a $\Sigma(\underline{x})$-constraint $\phi(\underline{x})$, it is possible to collect all the atoms occurring in it, say $\{\alpha_1, ..., \alpha_m\}$; for each α_j (for $j = 1, ..., m$), find the equivalent representative atom ψ_{k_j}, and then substitute each α_j with the corresponding ψ_{k_j} in ϕ and then eliminating duplicates. The result will be a constraint taken from the finitely many distinct conjunctions of literals built out of the finitely many representative atoms $\psi_1(\underline{x}), ..., \psi_n(\underline{x})$. This can be generalized to sets of constraints in the obvious way. Clearly, this implies the constraint compactness of T. □

As an immediate consequence of the last two facts we obtain the following result.

Corollary 1. *Let T be an (effectively) locally finite theory. Then, the algorithm in Figure 1 terminates for every query.*

This will be a crucial ingredient in the reduction of goal reachability and log validity to the satisfiability of the BSR($\Sigma^{\underline{R}}$) class.

5.2 Reduction to satisfiability

We first consider the log validity problem as goal reachability can be reduced to this problem.

Lemma 2. *Let T be a locally finite Σ-theory admitting elimination of quantifiers. Furthermore, let $(\underline{past}, \tau, \omega, \pi)$ be a cumulative and policy-aware transducer over $(\Sigma, \underline{R})$, db be a database over DB, and Log $\subseteq$ In $\cup$ Out. The log validity problem for $(\underline{past}, \tau, \omega, \pi)$ reduces to the T-satisfiability problem of an effectively computable BSR($\Sigma^{\underline{R}}$)-formula.*

Proof. The proof is along the lines of Theorem 3.1 in [17] for Spocus transducers. A log $L_1, ..., L_n$ is valid if there exists an input sequence $I_1, ..., I_n$ generating it. We view $I_1, ..., I_n$ as database over the set of predicate symbols obtained by making n copies of each relation $r \in In$ yielding $r_1, ..., r_n$. To state that the input sequence $I_1, ..., I_n$ yields the log $L_1, ..., L_n$, we require that the relations in $I_1, ..., I_n$ recorded by the log have the values specified by $L_1, ..., L_n$ together with the relations in *Out* determined by $I_1, ..., I_n$. For input relations, the situation is easy (we analyze concrete cases as the generalization are straightforward). For example, suppose that L_j requires that a tuple $\underline{c}$ belongs to the relation r_j in the input sequence; this can be stated by the following BSR($\underline{R}$) formula:

$$\exists \underline{x}.(r_j(\underline{x}) \wedge \underline{x} = \underline{c}). \tag{2}$$

For example, if the log specifies that r_j contains the tuples $\underline{c}$ and $\underline{d}$, the inclusion of r_j in the log is state by the following BSR($\underline{R}$) formula:

$$\forall \underline{x}.(r_j(\underline{x}) \rightarrow (\underline{x} = \underline{c} \vee \underline{x} = \underline{d})). \tag{3}$$

For relation in the output sequence, we need to resort to Corollary 1 above: the fix-point algorithm in Figure 1 terminates on all queries and generates a (finite) set of constraint facts. Then, for each fact of the form $r_j(\underline{x}) \leftarrow \psi_j^i(\underline{x})$ in the output set, we take the disjunction of each ψ_j^i for $i \geq 0$. Let $\varphi_j(\underline{x})$ be the resulting quantifier-free $\Sigma(\underline{x})$-formula. The remainder is similar to the case of input relations. Requiring that a tuple belongs to the relation r_j in the output sequence can be expressed by a formula similar to (2) except that r_j is replaced by φ_j obtained by the fix-point computation. Similarly, if the log specifies that r_j contains the two tuples $\underline{c}$ and $\underline{d}$, the inclusion of r_j in the log is stated by a formula which is similar to (3) except that r_j is replaced by φ_j obtained by the fix-point computation. It is easy to see that the resulting formula is a BSR($\underline{R}$) formula and its satisfiability is equivalent to the existence of an input sequence yielding the desired log. □

Before stating our main decidability results for log validity and goal reachability of cumulative and policy-aware transducers, we observe the following. An immediate consequence of Lemma 2 and Theorem 1 is the decidability of both verification problems for cumulative and policy-aware transducers whose Σ-theory is locally finite and admits elimination of quantifiers. Unfortunately, local finiteness and elimination of quantifiers are difficult to reconcile as the former is obtained for theories with simple "algebraic structure" while the latter is satisfied for those with rich "algebraic structure". To illustrate, we consider three theories. The theory whose signature contains only constants $c_1, ..., c_n$ and axiomatized by the following sentences:

$$\begin{aligned} c_i \neq c_j \text{ for } i, j = 1, ..., n \text{ and } i \neq j \\ \forall x. x = c_1 \vee \cdots x = c_n, \end{aligned}$$

is both locally finite and admits elimination of quantifier. The theory T_{lo} of linear orders whose signature contains the binary relation $\preceq$ and axiomatized by the following sentences:

$$\begin{aligned} \forall x, y, z. (x \preceq y \wedge y \preceq z \rightarrow x \preceq z) \quad & \forall x, y. (x \preceq y \wedge y \preceq x \rightarrow x = y) \\ \forall x. x \preceq x \quad & \forall x, y. (x \preceq y \vee y \preceq x) \end{aligned}$$

is locally finite, the SMT(T_{lo}) problem is decidable, but it does not admit elimination of quantifiers. The theory T_{dlo} of dense linear orders is the theory over the same signature of T_{lo} and obtained by adding the following two sentences to the set of axioms of T_{lo}:

$$\forall x, y. (x \prec y \rightarrow \exists z. (x \prec z \wedge z \preceq y)) \text{ and } \exists x, y. x \neq y\,,$$

where $t_1 \prec t_2$ abbreviates $t_1 \preceq t_2 \wedge t_1 \neq t_2$ for t_1, t_2 variables. The SMT(T_{dlo}) problem is decidable, T_{dlo} admits elimination of quantifiers, but it is not locally finite.

Fortunately, it is sometimes possible to reconcile local finiteness and quantifier elimination. For the theories T_{lo} and T_{dlo}, it is possible to show that a quantifier-free formula is T_{lo}-satisfiable iff it is T_{dlo}-satisfiable. So, to check the T_{lo}-satisfiability of BSR($\Sigma^{\underline{R}}$) sentences, we can use its being locally finite to apply the instantiation procedure underlying the proof of Lemma 2 while we can use the fact that T_{dlo} admits elimination of quantifiers to obtain the termination of the fix-point algorithm in Figure 1 (as we know that the T_{lo}-satisfiability of quantifier-free formulae is preserved). This phenomenon is not an accident but an application of the notion of model completeness of a theory.

A Σ-theory T is *model complete* iff, for all models $\mathcal{M}, \mathcal{N}$ of T, if $\mathcal{M}$ is a sub-structure of $\mathcal{N}$, then $\mathcal{M}$ is also an *elementary* sub-structure of $\mathcal{N}$, i.e. for every Σ-formula $\varphi(\underline{x})$ and all elements $\underline{a}$, we have that $\mathcal{M} \models \varphi(\underline{a})$ iff $\mathcal{N} \models \varphi(\underline{a})$. Intuitively, the elementary sub-model $\mathcal{M}$ of a model $\mathcal{N}$ preserves the set of satisfiable first-order formulae (and hence of quantifier-free formulae in particular). It is possible to show that T_{dlo} is model complete (see, e.g., [4]). The key property

(see Remark 3.5.6 in [4]) is that $\mathcal{M}$ is a sub-model of the set of universal sentences which are logical consequences of T_{dlo} (if T is a theory, the set of universal sentences which are T-valid is denoted with $T^{\forall}$) iff $\mathcal{M}$ is a sub-model of some model of $T^{\forall}_{dlo}$. (The validity of universal sentences is the dual of the satisfiability of quantifier-free formulae.) Now, since $T^{\forall}_{dlo} = T^{\forall}_{lo}$ (see [10]), we have formally justified the use of T_{lo} in Lemma 2 and that of T_{dlo} to eliminate quantifiers in the algorithm of Figure 1. Notice that the class of models of T_{lo} is closed under sub-structures since its axioms are universal sentences (see, e.g., [4]). Furthermore, these observations constitute the sketch of the proof of our first (main) decidability result about the verification of cumulative and policy-aware transducers.

Theorem 3. *Let $(\underline{past}, \tau, \omega, \pi)$ be a cumulative and policy-aware transducer over $(\Sigma, \underline{R})$, db be a database over DB, and Log $\subseteq$ In $\cup$ Out. If (a) T is a locally finite theory such that the SMT(T) problem is decidable and its class of models is closed under sub-structures, (b) $T_* \supseteq T$ is a model complete theory admitting elimination of quantifiers, and (c) $T^{\forall} = T^{\forall}_*$, then the log validity problem for $(\underline{past}, \tau, \omega, \pi)$ is decidable.*

We are now ready to state and prove our second (main) decidability result.

Theorem 4. *Let $(\underline{past}, \tau, \omega, \pi)$ be a cumulative and policy-aware transducer over $(\Sigma, \underline{R})$, db be a database over DB, and Log $\subseteq$ In $\cup$ Out. If (a) T is a locally finite theory such that the SMT(T) problem is decidable and its class of models is closed under sub-structures, (b) $T_* \supseteq T$ is a model complete theory admitting elimination of quantifiers, and (c) $T^{\forall} = T^{\forall}_*$, then the goal reachability problem for $(\underline{past}, \tau, \omega, \pi)$ is decidable.*

Proof. Again the proof is along the lines of Theorem 3.2 in [17] for Spocus transducers. First of all, observe that only runs of length two should be considered. To understand why this is so, consider an input sequence $I_1, ..., I_n$ with $n > 2$. Since outputs depend only on the current input, the database, and the state relations (storing the union of all previous inputs), the last output in the run of the transducer on $I_1, ..., I_n$ is the same as the last output on the run of the same transducer on the following input sequence of length 2: $(I_1 \cup \cdots I_{n-1}), I_n$. At this point, the problem is reduced to that of the satisfiability of a BSR($\underline{R}$) with the same technique described in the proof of Lemma 2. □

6 Conclusion

We have introduced a class of policy-aware relational transducers. We have studied the hypotheses under which log validity and goal reachability are decidable for this class of transducers by a reduction to the satisfiability problem of an extension of the BSR class of formulae.

There are two main lines of future work. First, we intend to study the decidability of those verification problems involving two or more transducers such as containment and equivalence [17]. Second, it would be interesting to see how

to implement the algorithm for checking the satisfiability of the extension of the BSR class on top of state-of-the-art theorem provers or Satisfiability Modulo Theories solvers. The latter, in particular, with the recent interest in developing decision procedures for the BSR class (see, e.g., [5]) and techniques for quantifier instantiation (see, e.g., [9]) seem to be ideal candidates.

Acknowledgments. The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures." We thank Silvio Ghilardi and the members of the AVANTSSAR project for many stimulating discussions.

References

1. Franz Baader and Silvio Ghilardi. Connecting many-sorted theories. *Journal of Symbolic Logic*, 72:535–583, 2007.
2. M. Y. Becker, C. Fournet, and A. D. Gordon. Security Policy Assertion Language (SecPAL). http://research.microsoft.com/en-us/projects/SecPAL/.
3. M. Y. Becker and S. Nanz. The Role of Abduction in Declarative Authorization Policies. In *10th International Symposium on Practical Aspects of Declarative Languages (PADL)*, 2008.
4. C.-C. Chang and J. H. Keisler. *Model Theory*. North-Holland, Amsterdam-London, third edition, 1990.
5. L. de Moura and N. Bjørner. Deciding effectively propositional logic using dpll and substitution set. In *Int. Joint Conference on Automated Reasoning*, 2008.
6. J. Van den Bussche. *Constraint Databases*, chapter Constraint databases, queries, and query languages. Springer, 2000.
7. J. DeTreville. Binder, a logic-based security language. In *IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 2002. IEEE Computer Society.
8. H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York-London, 1972.
9. Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. *Annals of Mathematics and Artificial Intelligence*, 2009. To appear.
10. S. Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4), 2004.
11. Y. Gurevich and I. Neeman. DKAL: Distributed-knowledge authorization language. In *Proceedings of CSF 2008*, pages 149–162. IEEE Computer Society, 2008.
12. N. Li and J. C. Mitchell. Datalog with constraints: a foundation for trust management langauges. In *PADL'03*, pages 58–73, 2003.
13. S. Ranise. Towards Verification of Security-Aware E-Services. Extended version of the paper with the same title in FTP'09.
14. M. Spielmann. Verification of relational transducers for electronic commerce. In *19th ACM Symp. on Princ. of DB Systems (PODS)*. ACM Press, 2000.
15. C. Tinelli and C. G. Zarba. Combining non-stably infinite theories. *Journal of Automated Reasoning*, 34(3), 2005.
16. D. Toman. Computing the well-founded semantics for constraint extensions of datalog. In *2nd Int. WS on Constraint Database Systems*, volume 1191 of *LNCS*, pages 64–79, 1997.
17. S. Abiteboul and V. Vianu, B. Fordham, and Y. Yesha. Relational Transducers for Electronic Commerce. *J. of Comp. and Sys. Sciences*, 61:236–269, 2000.

A Proofs

Lemma 1 [Instantiation]

Let T be a locally finite Σ-theory and the class of models of T be closed under sub-structures. Furthermore, let $\underline{R}$ be a finite set of predicate symbols such that $\Sigma \cap \underline{R} = \emptyset$. The $BSR(\Sigma^{\underline{R}})$-sentence

$$\exists \underline{x}.\forall \underline{y}.\varphi(\underline{x}, \underline{y}) \tag{4}$$

is satisfiable iff it is satisfiable in a finite model iff the quantifier-free formula

$$\exists \underline{x}. \bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma) \tag{5}$$

is satisfiable, where σ ranges over the substitutions mapping the variables $\underline{y}$ into the set of representative $\Sigma(\underline{x})$-terms and $\underline{y}\sigma$ denotes the simultaneous application of σ to the variables of the tuple $\underline{y}$.

Proof. W.l.o.g. assume that the length of $\underline{x}$ is strictly greater than 0; this avoids to consider structures with empty domains. First, we show that

$$(4) \text{ is satisfiable } \Leftrightarrow (5) \text{ is satisfiable.}$$

The $\Rightarrow$-side of the equivalence is trivial. So, we focus on the $\Leftarrow$-side: we assume the satisfiability of (5) and show the satisfiability of (4). Since (5) is satisfiable, we have that

$$\mathcal{M} \models \exists \underline{x}. \bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma),$$

for some $\mathcal{M} \in \mathcal{C} = \{\mathcal{M} \text{ is a } \Sigma\text{-structure} \mid \mathcal{M} \models T\}$. So, we can find a mapping v assigning elements of the domain D of $\mathcal{M}$ to the variables in $\underline{x}$ in such a way that

$$\mathcal{M}, v \models \bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma).$$

Now, consider the model $\mathcal{M}'$ obtained from $\mathcal{M}$ as follows. Let the domain D' of $\mathcal{M}'$ be the set of elements in D which are the images of the variables in $\underline{x}$ according to the assignment v above. Furthermore, let the interpretation of $\mathcal{M}'$ be obtained from that of $\mathcal{M}$ by restricting it to D'. By construction of $\mathcal{M}'$, we have that

$$\mathcal{M}', v \models \bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma);$$

furthermore, since φ is quantifier-free and since, varying σ, the elements assigned to the terms $\underline{y}\sigma$ cover all the $\underline{y}$-tuples of elements in the domain D' of $\mathcal{M}'$, we have that

$$\mathcal{M}' \models \forall \underline{y}.\varphi(\underline{x}, \underline{y}).$$

To conclude that (4) is satisfiable—thereby showing the equivalence of the satisfiability of (4) and (5)—observe that $\mathcal{M}'$ is a substructure of $\mathcal{M}$ and, by assumption, $\mathcal{C}$ is closed under sub-structures, so that also $\mathcal{M}' \in \mathcal{C}$. Hence, (4) is satisfiable. Since the model $\mathcal{M}'$ is finite, the above also proves the remaining equivalence, i.e. (4) is satisfiable iff it is satisfiable in a finite model. □

Theorem 1 [Decidability]

Let T be a locally finite Σ-theory whose class of models is closed under substructures and the SMT(T) problem be decidable; $\underline{R}$ be a finite set of predicate symbols such that $\Sigma \cap \underline{R} = \emptyset$. Then, the satisfiability of $BSR(\Sigma^{\underline{R}})$-sentences is decidable.

Proof. By Lemma 1, given a satisfiability problem for a $BSR(\Sigma^{\underline{R}})$-sentence of the form (4), we equivalently consider the satisfiability of a formula of the form (5). Since existentially quantified variables in $\underline{x}$ can be regarded as Skolem constants, we are left with the problem of checking the satisfiability of a quantifier-free formula of the form:

$$\bigwedge_{\sigma} \varphi(\underline{x}, \underline{y}\sigma),$$

where σ ranges over the mapping from $\underline{y}$ to the set $\underline{t}$ of representative $\Sigma(\underline{x})$-terms. By uniquely renaming the terms in $\underline{t}$ with fresh constant symbols (let $\underline{c}$ be the set of such constants), we can equivalently consider the satisfiability of the following quantifier-free formula:

$$\bigwedge_{i=1}^{m} c_i = t_i \wedge \bigwedge_{\sigma} \varphi'(\underline{x}, \underline{y}\sigma),$$

where φ' has been obtained from φ by replacing each term t_i in $\underline{t}$ with the corresponding fresh constant c_i in $\underline{c}$ (for $i = 1, ..., m$). Since, by assumption $\Sigma \cap \underline{R} = \emptyset$, we can consider the problem of checking the satisfiability of constraints (i.e. conjunctions of literals) of the following form:

$$\Gamma_T(\underline{x}, \underline{c}) \wedge \Gamma_{\underline{R}}(\underline{x}, \underline{c}), \tag{6}$$

where $\Gamma_{\underline{R}}$ is a conjunction of $\underline{R}$-literals and Γ_T is a a conjunction of Σ-literals. (Notice that an equality of the form $c_i = t_i$ can be considered as a Σ-literal, for $i = 1, ..., m$.) This amounts to checking the satisfiability in the disjoint union of T and the empty theory over the signature $\underline{R} \cup \underline{c} \cup \underline{x}$. Recall that, by assumption, the SMT(T) problem is decidable. Also, observe that checking the satisfiability of conjunctions of $(\underline{R} \cup \underline{c} \cup \underline{x})$-literals is a particular case of the problem of checking the satisfiability of formulae in the BSR class, which is well-known to be decidable. We can decide the satisfiability of (6) by using the following non-deterministic algorithm:

1. guess an equivalence relation Δ on $\underline{c} \cup \underline{x}$ and assume that $\underline{c} \cup \underline{x} = j_1, ..., j_n$;

2. check the T-satisfiability of

$$\Gamma_T(\underline{x}, \underline{c}) \wedge \bigwedge_{(j_k, j_l) \in \Delta} j_k = j_l \wedge \bigwedge_{(j_k, j_l) \notin \Delta} j_k \neq j_l \ ;$$

3. check the satisfiability of

$$\Gamma_R(\underline{x}, \underline{c}) \wedge \bigwedge_{(j_k, j_l) \in \Delta} j_k = j_l \wedge \bigwedge_{(j_k, j_l) \notin \Delta} j_k \neq j_l \ ;$$

4. if both satisfiability checks above are unsuccessful for all possible equivalence relations, then return the unsatisfiability of (6); otherwise (i.e. both checks are successful for at least one equivalence relation), return its satisfiability.

Notice that equalities are propagated only in one direction; from the theory T to the BSR theory. There is more than one way to show the correctness of this non-deterministic algorithm. However, notice that it is not possible to apply the Nelson-Oppen combination method as T is not assumed to be stably-infinite (while it is possible to show that any BSR theory is stably infinite, see, e.g., [15]). One possibility, clearly showing why it is necessary to only propagate equalities in one direction, is to apply the combination method in [1]. To do this, proceed as follows. First, introduce a sort *Universe* for constants in $\Sigma \cup \underline{c} \cup \underline{x}$, regard each predicate $p \in \underline{R}$ of arity n as a function of sort $\mathit{Universe}^n \rightarrow \mathit{Bool}$, and consider two constant symbols $\top$ and $\bot$ of sort *Bool*. Then, the conjunction (6) of literals can be transformed into the following equisatisfiable conjunction of (pure) literals:

$$\Gamma_T(\underline{x}, \underline{c}) \wedge \bigwedge_{p(\underline{i}_p) \in \Gamma_{\underline{R}}} p(\underline{i}_p) = \top \bigwedge_{\neg p(\underline{i}_p) \in \Gamma_{\underline{R}}} p(\underline{i}_p) = \bot \wedge \top \neq \bot \ , \tag{7}$$

where $\underline{i}_p \subseteq \underline{x} \cup \underline{c}$ is a tuple of variables of sort *Universe* whose length is equal to the arity of the predicate p. The satisfiability of (7) can be seen as the problem of finding a model for T (over the sort *Universe*), a model for the empty theory (over the sort *Bool*), and the functions in $\underline{R}$ connecting their domains (as their sort is $\mathit{Universe} \times \cdots \times \mathit{Universe} \rightarrow \mathit{Bool}$). This is a (generalization of) theory connection in the sense of [1] and the associated satisfiability problem can be solved by propagating equalities in just one direction. This is exactly what the non-deterministic algorithm above does and it is the key insight to proving its correctness. □