Extending $\mathcal{H}_1$-Clauses with Path Disequalities

Helmut Seidl and Andreas Reuß*

Technische Universität München, Institut für Informatik I2, Boltzmannstraße 3,
D-85748 Garching, Germany

Abstract. We extend $\mathcal{H}_1$-clauses with disequalities between paths. This extension allows conveniently to reason about freshness of keys or nonces, as well as about more intricate properties such as that a voter may deliver at most one vote. We show that the extended clauses can be normalized into an equivalent tree automaton with path disequalities and therefore conclude that satisfiability of conjunctive queries to predicates defined by such clauses is decidable.

1 Introduction

In general, satisfiability of a finite conjunction of Horn clauses is undecidable. In [12,8], a class $\mathcal{H}_1$ of Horn clauses has been identified for which satisfiability is decidable in exponential time. The class $\mathcal{H}_1$ differs from general Horn clauses in that it requires the heads of clauses to contain at most one constructor and to be linear, i.e., no variable may occur twice in the head. Since any finite conjunction of Horn clauses can be approximated by means of $\mathcal{H}_1$-clauses, the decision procedure for $\mathcal{H}_1$ can be used for automatic program analysis based on abstract interpretation, given that the analysis problem can conveniently be described by means of Horn clauses. This approach has been successfully applied to the automatic analysis of secrecy in cryptographic protocols and their implementation [7].

In [11], the class $\mathcal{H}_1$ of Horn clauses has been extended with *disequality* constraints on terms. Such kinds of constraints allow to express that a key is *fresh*, i.e., different from all keys in a given list.

Example 1. The $\mathcal{H}_1$-clauses

$$\begin{array}{ll} fresh\_key(X) & \Leftarrow q(X,[\,]) \\ q(X,Z) & \Leftarrow q(X,::(Y,Z)), X \neq Y \\ q(X,Y) & \Leftarrow key(X), old\_keys(Y) \end{array}$$

define a predicate $fresh\_key$ which expresses that a key is not contained in the list old_keys. Here, the upper case letters represent variables, $[\,]$ denotes an empty list, and $::$ is the list constructor. The definition of the predicate q ensures that (within the least model) $q(t,[\,])$ only holds if $q(t,l)$ holds for a list l where t is not contained in l. □

In [11], we extended the normalization procedure for $\mathcal{H}_1$ from [8,6] to clauses with term disequality constraints. This procedure transforms every finite set of $\mathcal{H}_1$-clauses

* The author was supported by the DFG Graduiertenkolleg 1480 (PUMA).

L. Birkedal (Ed.): FOSSACS 2012, LNCS 7213, pp. 165–179, 2012.

with term disequality constraints into an equivalent finite set of *automata clauses* with disequality constraints [9] and thus allows to decide whether or not a given query is satisfiable.

Disequalities between terms, however, are not expressive enough for expressing more involved properties of cryptographic protocols than freshness of keys or nonces. Consider, e.g., a voting protocol where for a submitted vote not freshness of the overall expression must be checked but that the voter has not submitted any vote so far [4,1].

Example 2. Assume that we are given a list l of votes where each element of l is of the form $vote(p, v)$ for a constructor symbol $vote$, a person p who has voted, and his or her vote v. The next vote $vote(p', v')$ then should not only be not contained in the list l but should differ from all elements of l in the first argument. This can be expressed by the predicate $valid$ as defined by:

$$\begin{array}{ll} valid(X) & \Leftarrow q(X, [\,]) \\ q(X, Z) & \Leftarrow q(X, :: (Y, Z)),\ X.1 \neq Y.1 \\ q(X, Y) & \Leftarrow is\_vote(X), votes(Y) \end{array}$$

The second clause of this definition involves a comparison between the person trying to vote, identified by $X.1$, and the leftmost subterm of the first list entry, identified by $Y.1$, which denotes one of the persons who have already voted. □

Disequalities between subterms identified by paths, though, turn out to be provably more expressive than disequalities between terms. Finite tree automata with disequality constraints between paths have extensively been studied by Comon and Jacquemard [2] who showed that emptiness for these automata is decidable. Moreover, they applied this emptiness test to provide a DEXPTIME algorithm for deciding inductive reducibility [3]. While for tree automata with term disequalities, universality is decidable [9] — this problem is undecidable for tree automata with disequalities between paths [5].

In this paper, we consider $\mathcal{H}_1$-clauses with disequalities between paths and try to provide a normalization procedure in the spirit of [8,11] to construct for every finite set of clauses an equivalent automaton with disequalities between paths. The construction, turns out to be more intricate than the construction for $\mathcal{H}_1$-clauses with disequalities between terms. The reason is that now extra precautions must be taken that only finitely many different clauses are encountered during normalization. The key issue is to avoid that the lengths of paths occurring in constraints grow. In order to deal with that, we extend the language of constraints by additionally allowing disequalities between arbitrary terms which, instead of plain variables, contain *path queries* to variables. Another problem arises when a clause is to be split in order to remove variables which do not occur in the head. In this particular case of quantifier elimination, pigeon-hole like arguments as applied in [11] do no longer suffice. Instead, we develop a novel technique which is based on *blind exploration*.

The rest of the paper is organized as follows. Section 2 contains basic notions of paths and constraints, while Section 3 introduces subclasses of Horn clauses extended with disequality constraints. Section 4 then compares classes of automata extended with disequalities with respect to their expressiveness. In Section 5 we provide the normalization procedure for $\mathcal{H}_1$-clauses with path disequalities. Finally Section 6 concludes.

2 Basics

In the following, we consider a fixed finite ranked alphabet Σ of atoms (with arity 0) and constructor symbols (with arities ≥ 1). In order to avoid trivial cases, we assume that there is at least one atom and at least one constructor. Let $\mathcal{T}_\Sigma$ denote the set of all *ground* (i.e., variable-free) terms over Σ. Note that, according to our assumption on Σ, $\mathcal{T}_\Sigma$ is infinite. A *labeled path* π is a finite sequence $(f_1, i_1).(f_2, i_2).\ldots.(f_n, i_n)$ where for every j, $f_j \in \Sigma$ and $1 \leq i_j \leq m_j$ if m_j is the arity of f_j. As usual, the empty sequence is denoted ϵ. An expression $X.\pi$ for a variable X and a path π is called *path expression*. In case, $\pi = \epsilon$, we also write X for $X.\epsilon$.

A *general term* is built up from path expressions and nullary constructors by means of constructor application. For a general term t, the sub-term at path π, denoted by $t.\pi$, is recursively defined by:

- $t.\epsilon = t$;
- $(Y.\pi_1).\pi_2 = Y.(\pi_1.\pi_2)$;
- $t.(f, i).\pi' = t_i.\pi'$ if $t = f(t_1, \ldots, t_m)$ and $1 \leq i \leq m$. In particular, for $g \neq f$, $t.(g, i).\pi'$ is *undefined*.

Example 3. For an atom a and a binary constructor b, the following expressions t_i with

$$t_1 = a \qquad t_2 = b(b(a, a), X) \qquad t_3 = b(a, X.(b, 1).(b, 2))$$

all are general terms. We have, e.g.:

$$\begin{array}{llll} t_2.(b,1).(b,1) = a & & t_3.(b,2) & = X.(b,1).(b,2) \\ t_2.(b,2).(b,2) = X.(b,2) & & t_3.(b,2).(b,1) & = X.(b,1).(b,2).(b,1) \end{array}$$

□

A *general disequality constraint* is a finite conjunction of general disequalities, each of which is of the form $t_1 \neq t_2$ where t_1, t_2 are general terms. This notion subsumes *term* disequality constraints as considered in [9,11] which allow variables X only, i.e., rule out path expressions $X.\pi$ with $\pi \neq \epsilon$. This notion also subsumes *labeled-path* disequality constraints where each disequality is of the form $X.\pi \neq t$ where $X.\pi$ is a path expression and t is either another path expression $Y.\pi'$ or a *ground* term, i.e., a term not containing any variables or path expressions. In the following, we refer to general disequality constraints when we mention disequality constraints without further qualification.

Consider a general term t and a ground substitution θ which provides ground terms for all variables occurring in t. If $\theta(X).\pi$ is defined for each path expression $X.\pi$ occurring in t, then $t\theta$ is obtained from t by replacing each occurrence of $X.\pi$ with the ground term $\theta(X).\pi$. Otherwise, $t\theta$ is undefined. The ground substitution θ satisfies a disequality $t_1 \neq t_2$ between general terms t_1, t_2, if either $t_1\theta$ or $t_2\theta$ is undefined, or both terms are defined but different. In this case, we write $\theta \models (t_1 \neq t_2)$. Likewise, we extend satisfiability to arbitrary monotone Boolean combinations of disequalities by:

$$\begin{array}{l} \theta \models (\phi_1 \wedge \phi_2) \text{ iff } (\theta \models \phi_1) \wedge (\theta \models \phi_2) \\ \theta \models (\phi_1 \vee \phi_2) \text{ iff } (\theta \models \phi_1) \vee (\theta \models \phi_2) \end{array}$$

For convenience, we also consider *equality* constraints $t_1 = t_2$ between general terms. A ground substitution θ satisfies the equality $t_1 = t_2$ if and only if it does not satisfy the corresponding disequality $t_1 \neq t_2$, i.e., if both $t_1\theta$ and $t_2\theta$ are defined *and* equal.

Lemma 1. *For every disequality $t_1 \neq t_2$ between general terms t_1, t_2, a finite disjunction ϕ of path disequalities can be constructed such that $t_1 \neq t_2$ is equivalent to ϕ, i.e., for every substitution θ, it holds that $\theta \models (t_1 \neq t_2)$ iff $\theta \models \phi$.*

Proof. In the first step, we observe that the disequality $t_1 \neq t_2$ is equivalent to a finite disjunction of disequalities $X.\pi \neq t$ for suitable path expressions $X.\pi$ and subterms t occurring in t_1 or t_2. Now let Π denote the set of all labeled paths π' such that $t.\pi'$ either is a path expression or a ground term. Let Π_0 denote the minimal elements in Π, i.e., the set of all paths $\pi' \in \Pi$ where Π does not contain a proper prefix of π'. The elements in Π_0 denote labeled paths in t reaching maximal ground subterms or path expressions contained in t. Therefore, the subset Π_0 is finite. Then the disequality $X.\pi \neq t$ is equivalent to the disjunction

$$\bigvee_{\pi' \in \Pi_0} X.\pi.\pi' \neq t.\pi' \qquad \square$$

3 Horn Clauses

Let us briefly introduce the notions of Horn clauses and subclasses of Horn clauses which we consider here. Essentially, these classes are obtained from the classes considered in [11] by replacing constraints consisting of disequalities between terms with constraints consisting of disequalities between terms which now may also refer to path expressions. Thus, a Horn clause with (now general) disequality constraints is of the form:

$$q(t) \Leftarrow p_1(t_1), \ldots, p_k(t_k), \phi$$

where ϕ is a finite conjunction of disequalities between general terms, $q, p_1, \ldots, p_k$ are unary predicates, $t, t_1, \ldots, t_k$ are (ordinary) terms. Non-unary predicates can be integrated in our framework by equipping their arguments with an *implicit* constructor. As usual, $q(t)$ is called the *head*, while $p_1(t_1), \ldots, p_k(t_k), \phi$ is the *body* or *precondition* of the clause. The Horn clause is from the class $\mathcal{H}_1$, if the term t in the head contains at most one constructor and is linear, i.e., no variable occurs twice in t. For convenience, we adopt the convention that the (distinct) variables in the head are enumerated $X_1, \ldots, X_k$. This means that t either equals X_1 or is of the form $f(X_1, \ldots, X_k)$ for some constructor of arity k where the case of atoms is subsumed by choosing $k = 0$ (writing f instead of $f()$ in this case). Moreover for a distinction, variables *not* occurring in the head will be denoted by $Y, Y_1, \ldots$.

The Horn clause is a *normal clause* if it is of one of the forms:

$$q(X_1) \Leftarrow \phi \qquad \text{or}$$
$$q(f(X_1, \ldots, X_k)) \Leftarrow p_1(X_{i_1}), \ldots, p_r(X_{i_r}), \phi$$

where all variables occurring in the body of the clause also occur in the head. Moreover, the Horn clause is an *automata clause* if additionally each variable X_i occurring in the head occurs exactly once in the literals occurring in the body and the head contains exactly one constructor, i.e., the clause is of the form:

$$q(f(X_1, \ldots, X_k)) \Leftarrow p_1(X_1), \ldots, p_k(X_k), \phi \ .$$

In particular, each normal clause as well as each automata clause is $\mathcal{H}_1$.

Let $\mathcal{C}$ denote a (possibly infinite) set of Horn clauses. Then the *least model* of $\mathcal{C}$ is the least set M such that $q(t\theta) \in M$ for a ground substitution θ whenever there is a Horn clause $q(t) \Leftarrow p_1(t_1), \ldots, p_k(t_k), \phi$ such that $p_i(t_i\theta) \in M$ for all $i = 1, \ldots, k$, and $\theta \models \phi$. The set of all all terms s with $q(s) \in M$, for a given predicate q, then is denoted by $[\![q]\!]_{\mathcal{C}}$. Similar to the case of ordinary Horn clauses or Horn clauses with term constraints, we have:

Lemma 2 ([11]). *For every finite set $\mathcal{N}$ of normal clauses, a finite set $\mathcal{A}$ of automata clauses can be effectively constructed such that for every predicate p occurring in $\mathcal{N}$, $[\![p]\!]_{\mathcal{N}} = [\![p]\!]_{\mathcal{A}}$.* □

Example 4. For terms t, let $s^0(t) = t$, $s^{i+1}(t) = s(s^i(t))$. Consider the set $\mathcal{N}$ of normal clauses:

$$\begin{array}{ll} adult(pers(X_1, X_2)) & \Leftarrow age(X_1), name(X_2), old(X_1) \\ old(X_1) & \Leftarrow \bigwedge_{i \leq 18} X_1 \neq s^i(0) \\ age(0) & \Leftarrow \\ age(s(X_1)) & \Leftarrow age(X_1) \end{array}$$

In order to construct a corresponding finite set of automata clauses, the variables X_1 occurring in the heads of normal clauses $p(X_1) \Leftarrow \phi$ are instantiated with all terms $c(X_1, \ldots, X_k)$, c a constructor of arity $k \geq 0$. Additionally, we introduce auxiliary predicates for all conjunctions of predicates, possibly occurring in preconditions. For the set $\mathcal{N}$ of normal clauses, we obtain the set $\mathcal{A}$ of clauses:

$$\begin{array}{lll} adult(pers(X_1, X_2)) \Leftarrow age\_old(X_1), name(X_2) & & \top(0) \Leftarrow \\ old(s(X_1)) \Leftarrow \top(X_1), \bigwedge_{i \leq 17} X_1 \neq s^i(0) & & \top(s(X_1)) \Leftarrow \top(X_1) \\ age\_old(s(X_1)) \Leftarrow age(X_1), \bigwedge_{i \leq 17} X_1 \neq s^i(0) & & age(0) \Leftarrow \\ \top(pers(X_1, X_2)) \Leftarrow \top(X_1), \top(X_2) & & age(s(X_1)) \Leftarrow age(X_1) \\ old(pers(X_1, X_2)) \Leftarrow \top(X_1), \top(X_2) & & \end{array}$$

From the three possible new clauses defining the predicate old, we only kept the clauses for the constructors s and $pers$, since the precondition of the clause

$$old(0) \Leftarrow \bigwedge_{i \leq 18} 0 \neq s^i(0)$$

contains the disequality $0 \neq 0$ which is unsatisfiable. Concerning the required conjunctions, the new predicate $\top$ denotes the *empty* conjunction of predicates, i.e., accepts all terms in $\mathcal{T}_\Sigma$ where $\Sigma = \{0, s, pers\}$ is given by the atoms and constructors occurring in $\mathcal{N}$. And the new predicate age_old, on the other hand represents the conjunction of the predicates age and old. □

4 Automata Classes and Expressiveness

A finite set of automata clauses together with a finite set of accepting predicates can be considered as a non-deterministic tree automaton with disequality constraints. Different classes of disequality constraints thus correspond to different classes of tree automata. Automata clauses with term disequality constraints have been considered in [11], while tree automata with disequalities of path expressions have been investigated by Comon and Jacquemard [2]. In fact, they have only considered disequalities between *unlabeled* paths. Unlabeled paths are obtained from labeled paths $(f_1, i_1). \ldots . (f_n, i_n)$ by omitting the labels $f_1, \ldots, f_n$. The resulting automata, though, are equally expressive since the constructors occurring in paths can be recorded by means of specialized predicates. E.g., a clause

$$p(f(X_1, X_2)) \Leftarrow q(X_1), r(X_2),\ X_1.(f, 1) \neq X_2$$

is replaced by the clauses

$$p_{f(\_,\_)}(f(X_1, X_2)) \Leftarrow q_{f(\_,\_)}(X_1), r_t(X_2), \quad X_1.1 \neq X_2$$
$$p_{f(\_,\_)}(f(X_1, X_2)) \Leftarrow q_{g(\_,\ldots,\_)}(X_1), r_t(X_2)$$

for arbitrary patterns t and all $g \neq f$. In this construction, only those patterns are enumerated which are made up from suffixes of paths occurring in the given set of automata clauses. For the reverse direction, we observe that every unlabeled-path disequality is equivalent to a finite conjunction of labeled-path disequalities (since Σ is finite). Recalling Lemma 1, the following simulations therefore can be proven.

Theorem 1. *Assume that $\mathcal{C}$ is a finite subset of automata clauses with general term disequality constraints. Then the following holds:*

1. *A finite subset $\mathcal{C}'$ of automata clauses with labeled-path disequality constraints can be effectively constructed such that for every predicate p, $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\mathcal{C}'}$.*
2. *A finite subset $\mathcal{C}''$ of automata clauses with unlabeled-path disequality constraints can be effectively constructed such that for all predicates p, $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\mathcal{C}''}$.*

□

In [2], it has also been proven that emptiness for finite tree automata with unlabeled-path disequality constraints is decidable. From that, we deduce that emptiness is also decidable for automata clauses with general term disequalities. We have:

Corollary 1. *Given a finite set $\mathcal{C}$ of automata clauses with general term disequality constraints and a predicate p, it is decidable whether or not $[\![p]\!]_{\mathcal{C}} = \emptyset$. Moreover in case that $[\![p]\!]_{\mathcal{C}} \neq \emptyset$, a witness $t \in [\![p]\!]_{\mathcal{C}}$ can effectively be computed.* □

Since unlabeled-path disequality constraints can be expressed by means of labeled-path disequality constraints, and labeled-path disequality constraints are a special case of general term disequality constraints, all three classes of automata clauses compared in Theorem 1, are equally expressive.

Term disequality constraints are special cases of general term disequality constraints. Therefore by Theorem 1, automata clauses with term disequality constraints can be simulated by means of automata clauses with labeled-path or unlabeled-path disequality

constraints. The reverse simulation, though, is not possible. One indication is that universality is decidable [9] for tree automata with term disequality constraints — while it is undecidable for tree automata with path disequality constraints, since emptiness for automata with path *equalities* only [5] is undecidable.

Here, we additionally present a specific language T which can be expressed as $[\![p]\!]_{\mathcal{C}}$ for a finite set $\mathcal{C}$ of automata clauses with labeled-path disequality constraints, but which cannot be characterized by means of finite sets of automata clauses with term disequality constraints only. Let $T = [\![p]\!]_{\mathcal{C}}$ for the following set $\mathcal{C}$ of automata clauses:

$$\begin{array}{ll} p(f(X_1, X_2)) & \Leftarrow \top(X_1), \top(X_2), X_1 \neq X_2.(f, 1) \\ \top(f(X_1, X_2)) & \Leftarrow \top(X_1), \top(X_2) \\ \top(a) & \Leftarrow \end{array}$$

Lemma 3. *There is no finite set $\mathcal{C}'$ of automata clauses with term disequality constraints which define a predicate p such that $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\mathcal{C}'}$.*

Proof. Assume for a contradiction that there is such a finite set $\mathcal{C}'$ which defines such a predicate p. Then we construct a finite set $\mathcal{N}$ of normal clauses for $\mathcal{C}'$ using equality term constraints only such that for every predicate q of $\mathcal{C}'$, $\mathcal{N}$ has a predicate $\bar{q}$ with $[\![\bar{q}]\!]_{\mathcal{N}}$ is the complement of $[\![q]\!]_{\mathcal{C}'}$. This set is constructed as follows. Assume that $q(f(X_1, \ldots, X_k)) \Leftarrow l_{i1}, \ldots, l_{ir_i}$ for $i = 1, \ldots, m$ are the clauses for q and constructor f where each l_{ij} either is a literal of the form $p'(X_s)$ or a single disequality. Then the predicate $\bar{q}$ for constructor f is defined by the set of all clauses

$$\bar{q}(f(X_1, \ldots, X_k)) \Leftarrow \bigwedge_{1 \leq i \leq m} \bar{l}_{ij_i}$$

where for each i, $1 \leq j_i \leq r_i$. For a literal l_{ij_i} of the form $p'(X_s)$, $\bar{l}_{ij_i}$ is given by $\bar{p}'(X_s)$, and if l_{ij_i} equals a disequality $t_1 \neq t_2$, $\bar{l}_{ij_i}$ is given by $t_1 = t_2$. By this construction the resulting clauses contain equality constraints only. As in Lemma 2, a finite set $\mathcal{A}$ of automata clauses with term equality constraints can be computed such that for all predicates $\bar{q}$ of $\mathcal{N}$, $[\![\bar{q}]\!]_{\mathcal{N}} = [\![\bar{q}]\!]_{\mathcal{A}}$.

A similar construction for automata without constraints has been described in [10]. Another variant recently has been presented in [5].

Let $\overline{T}$ denote the complement of T, i.e., the set $\mathcal{T}_\Sigma \setminus T$. $\overline{T}$ consists of all elements of the form

$$f(t, f(t, s))$$

for arbitrary terms s, t. By construction, $[\![\bar{p}]\!]_{\mathcal{A}} = \bar{T}$. Then $\mathcal{A}$ must contain a clause

$$\overline{p}(f(X_1, X_2)) \Leftarrow q_1(X_1), q_2(X_2), \phi$$

where ϕ is a finite conjunction of term equalities, such that there are distinct ground terms $t_1, t_2 \in [\![q_1]\!]_{\mathcal{A}}$ and for $i = 1, 2$ there are two distinct terms t_{i1}, t_{i2} such that $f(t_i, t_{ij}) \in [\![q_2]\!]_{\mathcal{A}}$ and $f(t_i, f(t_i, t_{ij})) \in [\![\overline{p}]\!]_{\mathcal{A}}$ by application of this clause. This means for $\theta_{ij}(X_1) = t_i$ and $\theta_{ij}(X_2) = f(t_i, t_{ij})$, that $\theta_{ij} \models \phi$. In order to see this, we first convince ourselves that for every term t there must be some clause c_t by which for

two distinct terms s_1, s_2, facts $\overline{p}(f(t, f(t, s_1)))$ and $\overline{p}(f(t, f(t, s_2)))$ can be derived. Assume for a contradiction, this were not the case. Then for some t and every clause c for predicate $\overline{p}$, there is at most one term t_c such that a fact $\overline{p}(f(t, f(t, t_c)))$ is derived by means of c. Accordingly, for this t, the set $\{s \mid f(t, f(t, s)) \in \bar{T}\}$ were finite — which is not the case. Consequently, for every t we can find a clause c_t by which for two distinct terms s_1, s_2, facts $\overline{p}(f(t, f(t, s_1)))$ and $\overline{p}(f(t, f(t, s_2)))$ can be derived. Since the number of terms is infinite while the number of clauses is finite, we conclude that there must be two distinct terms t_1, t_2 for which the clauses c_{t_1} and c_{t_2} coincide.

Now recall that each finite conjunction of term equalities $s_i = s_j$ between arbitrary terms s_i, s_j can be expressed as a finite conjunction of term equalities of the form $Z = s$ for variables Z. W.l.o.g. let ϕ be of this form. Furthermore, recall that a term equality $Z = s$ where s contains Z either is trivially true or trivially false. In addition to equalities $X_1 = X_1$ and $X_2 = X_2$, the satisfiable constraint ϕ therefore can only contain equalities of one of the following forms:

(1) $X_1 = g$ or $X_2 = g$, where g denotes a ground term;
(2) $X_1 = s$, where s contains variable X_2;
(3) $X_2 = s$, where s contains variable X_1

In the following, we show that an equality of any of these forms leads to a contradiction.

Case 1. Assume that there is an equality $X_r = g$ for some ground term g. If $r = 1$, then either $t_1 \neq g$ or $t_2 \neq g$ implying that $\theta_{ij} \not\models (X_r = g)$ for some i, j. If on the other hand, $r = 2$, then either $f(t_1, t_{11}) \neq g$ or $f(t_1, t_{12}) \neq g$, and hence also $\theta_{ij} \not\models (X_r = g)$ for some i, j.

Case 2. Assume that there is an equality $X_1 = s$ where s contains X_2. If $\theta_{11} \models (X_1 = s)$, then t_1 would contain the term $f(t_1, t_{11})$ — which is impossible.

Case 3. Finally, assume that there is an equality $X_2 = s$ where s contains X_1. Then consider the two substitutions θ_{11} and θ_{12}. If $\theta_{11} \models (X_2 = s)$, we conclude that $f(t_1, t_{11}) = \theta_{11}(X_2) = s[t_1/X_1]$. Then $\theta_{12} \models (X_2 = s)$ implies that $s[t_1/X_1]$ also equals $f(t_1, t_{12})$, and therefore, $f(t_1, t_{11}) = f(t_1, t_{12})$. This however, implies that $t_{11} = t_{12}$ — in contradiction to our assumption.

We conclude that the conjunction ϕ is equivalent to true. But then $[\![\overline{p}]\!]_{\mathcal{A}}$ must also contain the term $f(t_1, f(t_2, t_{21}))$ — which is not contained in $\bar{T}$. This completes the proof. □

5 $\mathcal{H}_1$-Normalization

This section describes a normalization procedure which constructs for each finite set of $\mathcal{H}_1$-clauses with general term disequalities a finite set of normal clauses with general term disequalities which is equivalent. The normalization procedure repeatedly applies three rules. Each rule adds finitely many simpler clauses which are implied by the current set of clauses. The three rules are *resolution*, *splitting* and *propagation*. Thus, this general procedure is quite in-line with the normalization procedures for unconstrained $\mathcal{H}_1$-clauses [8,6] or $\mathcal{H}_1$-clauses with (ordinary) term disequalities [11]. In order to make this idea also work in presence of disequalities involving path expressions, a completely new construction for splitting is required (see subsection 5.2). Also the argument for

termination must be appropriately generalized. The following subsections provide the normalization rules. We refer to the current set of all implied clauses (whether originally present or added during normalization) as $\mathcal{C}$, while $\mathcal{N} \subseteq \mathcal{C}$ denotes the subset of normal clauses in $\mathcal{C}$.

5.1 Resolution

The first type of rules simplifies a complicated clause from $\mathcal{C}$ by applying a resolution step with a normal clause. Assume that $\mathcal{C}$ contains a clause $h \Leftarrow \alpha_1, p(t), \alpha_2, \psi$.

If $\mathcal{N}$ contains a clause $p(X_1) \Leftarrow \phi$, then we add the clause $h \Leftarrow \alpha_1, \alpha_2, \psi, \psi'$ where $\psi' = \phi[t/X_1]$.

If $\mathcal{N}$ has a clause $p(f(X_1, \ldots, X_k)) \Leftarrow \beta, \phi$, and $t = f(t_1, \ldots, t_k)$, then we add the clause:

$$h \Leftarrow \alpha_1, \alpha', \alpha_2, \psi, \psi'$$

where $\alpha' = \beta[t_1/X_1, \ldots, t_k/X_k]$ and likewise, $\psi' = \phi[t_1/X_1, \ldots, t_k/X_k]$. These resolution steps may introduce new disequalities. The new constraints are obtained from already available constraints by substitution of terms for variables. We remark, though, that after simplification of queries $t.\pi$ with t not a variable, the new constraints only contain path expressions for paths which are *suffixes* of paths already occurring in constraints of $\mathcal{C}$.

Example 5. Consider again the voting protocol Example 2:

$$\begin{aligned} &valid(X_1) \Leftarrow q(X_1, [\,]) \\ &q(X_1, X_2) \Leftarrow q(X_1, :: (Y, X_2)),\ X_1.(vote, 1) \neq Y.(vote, 1) \\ &q(X_1, X_2) \Leftarrow is\_vote(X_1), votes(X_2) \end{aligned}$$

enhanced with the normal clauses:

$$\begin{aligned} empty([\,]) \Leftarrow \qquad & h_b(pers(X_1, X_2)) \Leftarrow n_{bob}(X_1), age_{25}(X_2) \\ age_{15}(15) \Leftarrow \qquad & h_a(pers(X_1, X_2)) \Leftarrow n_{alice}(X_1), age_{15}(X_2) \\ age_{25}(25) \Leftarrow \qquad & p_{bob}(vote(X_1, X_2)) \Leftarrow h_b(X_1) \\ n_{alice}(alice) \Leftarrow \qquad & p_{alice}(vote(X_1, X_2)) \Leftarrow h_a(X_1) \\ n_{bob}(bob) \Leftarrow \qquad & votes(:: (X_1, X_2)) \Leftarrow p_{bob}(X_1), v'(X_2) \\ & v'(:: (X_1, X_2)) \Leftarrow p_{alice}(X_1), empty(X_2) \end{aligned}$$

to fill the list of already submitted votes with the two entries $vote(pers(alice, 15), \_)$ and $vote(pers(bob, 25), \_)$ where _ is intended to represent one of the atoms yes or no ($yes, no \in \Sigma$). Resolving the first two clauses of this example with the third one for the substitution $X_2 \mapsto [\,]$ and $X_2 \mapsto :: (Y, X_2)$, respectively, yields the clauses:

$$\begin{aligned} &valid(X_1) \Leftarrow is\_vote(X_1), votes([\,]) \\ &q(X_1, X_2) \Leftarrow is\_vote(X_1), votes(:: (Y, X_2)), X_1.(vote, 1) \neq Y.(vote, 1) \end{aligned}$$

With the clause $votes(:: (X_1, X_2)) \Leftarrow p_{bob}(X_1), v'(X_2)$, the second new clause can further be resolved to obtain

$$q(X_1, X_2) \Leftarrow is\_vote(X_1), p_{bob}(Y), v'(X_2), X_1.(vote, 1) \neq Y.(vote, 1)$$

5.2 Splitting

The next type of saturation rules is concerned with the removal of variables not contained in the head of a clause. Assume that $\mathcal{C}$ contains a clause $h \Leftarrow \alpha, \psi$ and Y is a variable occurring in α, ψ but not in h. We rearrange the precondition α into a sequence $\alpha', q_1(Y), \ldots, q_r(Y)$ where α' does not contain the variable Y. Then we construct a finite sequence $t_1, \ldots, t_l$ of ground terms such that, w.r.t. $\mathcal{N}$,

$$\psi[t_1/Y] \vee \ldots \vee \psi[t_l/Y]$$

is equivalent to $\exists\, Y.q_1(Y), \ldots, q_r(Y), \psi$, and add the clauses

$$h \Leftarrow \alpha', \psi[t_j/Y], \quad j = 1, \ldots, l$$

to the set $\mathcal{C}$. This operation is referred to as *splitting*.

According to this construction, splitting may introduce new disequalities. As in the case of resolution, new constraints are obtained from already available constraints by substitution of (ground) terms. This means that, after simplification of queries $t.\pi$ with t not a variable, the new constraints only contain path expressions for paths which are *suffixes* of paths already occurring in constraints of $\mathcal{C}$. The remainder of this section provides a proof that the finite sequence $t_1, \ldots, t_l$ of ground terms exists together with an effective construction of the t_j. For notational convenience, let us assume that we are not given a finite sequence $q_1, \ldots, q_r$ of predicates but just a single predicate p which is defined by means of a finite set of automata clauses $\mathcal{A}$.

Theorem 2. *Let $\mathcal{A}$ be a finite set of automata clauses, p a predicate, and Y a variable. For every conjunction of labeled-path disequalities ψ, a finite sequence of ground terms $t_1, \ldots, t_l$ can be constructed such that with respect to $\mathcal{A}$, the disjunction $\phi = \psi[t_1/Y] \vee \ldots \vee \psi[t_l/Y]$ is equivalent to the expression $\exists\, Y.\, p(Y), \psi$.*

Proof. W.l.o.g. we assume that the variable Y does not occur on both sides within the same disequality in ψ. Otherwise, we modify the set $\mathcal{A}$ of clauses in such a way that only those terms of the (original) set $[\![p]\!]_{\mathcal{A}}$ are accepted by the predicate p which satisfy those disequalities.

Now let Π denote the set of path expressions $Y.\pi$ occurring in ψ, and m the total number of occurrences of such expressions in ψ. We construct a finite sequence of terms $t_1, \ldots, t_l$ of $[\![p]\!]_{\mathcal{A}}$ such that for each ground substitution θ not mentioning Y, the following holds: if $\theta \oplus \{Y \mapsto t\} \models \psi$ for some $t \in [\![p]\!]_{\mathcal{A}}$, then also $\theta \oplus \{Y \mapsto t_i\} \models \psi$ for some i. Each of the terms t_i is generated during one possible run of the following nondeterministic algorithm. The algorithm starts with one term $s_0 \in [\![p]\!]_{\mathcal{A}}$. If no such term exists, the empty sequence is returned. Otherwise, the algorithm adds s_0 to the output sequence and proceeds according to one permutation of the occurrences of path expressions $Y.\pi$ occurring in ψ. Then it iterates of the path expressions in the permutation. In the round i for the path expression $Y.\pi$, the current set $\mathcal{A}$ of automata clauses is modified in such a way that all terms t with $t.\pi = s_{i-1}.\pi$ are excluded from $[\![p]\!]_{\mathcal{A}}$. Let $\mathcal{A}'$ be the resulting set of automata clauses. If $[\![p]\!]_{\mathcal{A}'}$ is empty the algorithm terminates. Otherwise, a term $s_i \in [\![p]\!]_{\mathcal{A}'}$ is selected and added to the output sequence.

For the correctness of the approach, consider an arbitrary ground substitution θ defined for all variables occurring in ψ with the exception of Y. First, assume that $\exists\, Y.\, p(Y), \psi\theta$ is not satisfiable. Then for no $s \in [\![p]\!]_{\mathcal{A}}$, $\psi\theta[s/Y]$ is true. Hence, also the finite disjunction provided by our construction cannot be satisfiable for such a θ, and therefore is equivalent to $\exists\, Y.\, p(Y), \psi\theta$. Now assume that $\theta \models \exists\, Y.\, p(Y), \psi$, i.e., some $s \in [\![p]\!]_{\mathcal{A}}$ exists with $\theta \models \psi[s/Y]$. Then we claim that there exists some s' occurring during one run of the nondeterministic algorithm with $\theta \models \psi[s'/Y]$. We construct this run as follows. Let $s_0 \in [\![p]\!]_{\mathcal{A}}$ denote the start term of the algorithm. If s_0 satisfies all disequalities, we are done. Otherwise, we choose one disequality $Y.\pi \neq t$ in $\psi\theta$ which is not satisfied. This means that $s_0.\pi = t$. Accordingly, we choose $Y.\pi$ as the first occurrence of a path expression selected by the algorithm. In particular, this means that all further terms s_i output by the algorithm will satisfy the disequality $Y.\pi \neq t$. After each round, one further disequality is guaranteed to be satisfied – while still s is guaranteed to be accepted at p by the resulting set $\mathcal{A}$ of automata clauses. □

Corollary 2. *Let $\mathcal{N}$ be a finite set of normal clauses, $q_1, \ldots, q_r$ a finite sequence of predicates, and Y a variable. For every conjunction of arbitrary constraints ψ, a finite sequence of ground terms $s_1, \ldots, s_l$ can be constructed such that with respect to $\mathcal{N}$, the disjunction $\phi = \psi[s_1/Y] \vee \ldots \vee \psi[s_l/Y]$ is equivalent to the expression $\exists\, Y.\, q_1(Y), \ldots, q_r(Y), \psi$.*

Proof. First, recall that we can construct a finite set $\mathcal{A}$ of automata clauses together with a predicate p such that $[\![p]\!]_{\mathcal{A}} = [\![q_1]\!]_{\mathcal{N}} \cap \ldots \cap [\![q_r]\!]_{\mathcal{N}}$. Clearly, $\exists\, Y.\, p(Y), \psi$ is implied by every constraint $\psi[s/Y]$ with $s \in [\![p]\!]_{\mathcal{A}}$.

By Lemma 1, every disequality $t_1 \neq t_2$ is equivalent to a disjunction of disequalities of the form $X.\pi \neq Z.\pi'$ or $X.\pi \neq t$ for variables X, Z and ground terms t. Accordingly, ψ is equivalent to a disjunction $\psi_1 \vee \ldots \vee \psi_k$ for suitable labeled-path constraints ψ_i. By Theorem 2, each conjunction $p(Y), \psi_i$ is equivalent to a disjunction $\psi_i[s_{i1}/Y] \vee \ldots \vee \psi_i[s_{il_i}/Y]$. Since $p(Y), \psi$ is equivalent to the disjunction

$$p(Y), \psi_1 \vee \ldots \vee p(Y), \psi_k$$

we conclude that it is equivalent to the disjunction:

$$\psi_1[s_{11}/Y] \vee \ldots \vee \psi_k[s_{kl_k}/Y]$$

The latter, on the other hand implies the disjunction

$$\psi[s_{11}/Y] \vee \ldots \vee \psi[s_{kl_k}/Y]$$

Therefore, the sequence $s_{11}, \ldots, s_{kl_k}$ satisfies the requirements of the corollary. □

Example 6. The resolution steps of Example 5 produced the clause

$$q(X_1, X_2) \Leftarrow is\_vote(X_1), p_{bob}(Y), v'(X_2), X_1.(vote, 1) \neq Y.(vote, 1)$$

In order to decide satisfiability of $\exists\, Y. p_{bob}(Y), X_1.(vote, 1) \neq Y.(vote, 1)$, the algorithm of Theorem 2 starts with the term $vote(pers(bob, 25), yes) \in [\![p_{bob}]\!]_{\mathcal{N}}$ for the subset $\mathcal{N}$ of normal clauses. Then it enforces the disequality $s.(vote, 1) \neq pers(bob, 25)$

for terms $s \in [\![p_{bob}]\!]_{\mathcal{N}}$ and finds out by a successful emptiness-test that no such term exists. Therefore only the *normal* clause:

$$q(X_1, X_2) \Leftarrow is\_vote(X_1), v'(X_2), X_1.(vote, 1) \neq pers(bob, 25)$$

is added. This new clause in turn enables two more resolution steps for the first two clauses of this voting protocol example, yielding

$$\begin{array}{ll} valid(X_1) \Leftarrow is\_vote(X_1), v'([\,]), & X_1.(vote, 1) \neq pers(bob, 25) \\ q(X_1, X_2) \Leftarrow is\_vote(X_1), v'(:: (Y, X_2)), & X_1.(vote, 1) \neq pers(bob, 25), \\ & X_1.(vote, 1) \neq Y.(vote, 1) \end{array}$$

for the substitution $X_2 \mapsto [\,]$ and $X_2 \mapsto :: (Y, X_2)$, respectively. Another resolution step with the clause $v'(:: (X_1, X_2)) \Leftarrow p_{alice}(X_1), empty(X_2)$ now yields:

$$\begin{array}{ll} q(X_1, X_2) \Leftarrow is\_vote(X_1), p_{alice}(Y), empty(X_2), & X_1.(vote, 1) \neq pers(bob, 25), \\ & X_1.(vote, 1) \neq Y.(vote, 1) \end{array}$$

with substitution $X_1 \mapsto Y$. To the last clause, again splitting can be applied in order to replace the precondition $p_{alice}(Y), X_1.(vote, 1) \neq Y.(vote, 1)$ with a disjunction of constraints not containing Y. As is the case with p_{bob}, the algorithm of Theorem 2 only finds one term for predicate p_{alice}, say $vote(pers(alice, 15), no)$. Then for the path $(vote, 1)$ the term $pers(alice, 15)$ is excluded and p_{alice} becomes empty. Thus, the new normal clause

$$\begin{array}{ll} q(X_1, X_2) \Leftarrow is\_vote(X_1), empty(X_2), & X_1.(vote, 1) \neq pers(bob, 25), \\ & X_1.(vote, 1) \neq pers(alice, 15) \end{array}$$

is added. Again the obtained normal clause enables two resolution steps for the first two clauses of the voting protocol, yielding

$$\begin{array}{l} valid(X_1) \Leftarrow is\_vote(X_1), empty([\,]), \phi \\ q(X_1, X_2) \Leftarrow is\_vote(X_1), empty(:: (Y, X_2)), X_1.(vote, 1) \neq Y.(vote, 1), \phi \end{array}$$

where ϕ abbreviates the conjunction $X_1.(vote, 1) \neq pers(bob, 25), X_1.(vote, 1) \neq pers(alice, 15)$. Finally a last resolution step with the clause $empty([\,]) \Leftarrow$ for the first of these two clauses achieves the result

$$valid(X_1) \Leftarrow is\_vote(X_1), \phi$$

where ϕ again equals $X_1.(vote, 1) \neq pers(bob, 25), X_1.(vote, 1) \neq pers(alice, 15)$, stating that a vote is valid if it is submitted by a person who is different from the two persons as stored in the given list. □

5.3 Propagation

The last type of rules considers clauses $p(X_1) \Leftarrow q_1(X_1), \ldots, q_r(X_1), \psi$ where ψ only contains the variable X_1 (or none). Assume that $r > 0$, and $\mathcal{N}$ contains normal clauses $q_j(f(X_1, \ldots, X_k)) \Leftarrow \alpha_j, \psi_j$ for $j = 1, \ldots, r$. Then we add the normal clause:

$$p(f(X_1, \ldots, X_k)) \Leftarrow \alpha_1, \ldots, \alpha_r, \psi_1, \ldots, \psi_r, \psi'$$

where $\psi' = \psi[f(X_1, \ldots, X_k)/X_1]$.

Also this rule may create new disequalities. Again, however, after simplification of queries $t.\pi$ where t is not a variable, the new constraints only contain path expressions for paths which are suffixes of paths already occurring in disequalities of the original set $\mathcal{C}$.

Example 7. Consider the following variant of the voting protocol example

$$\begin{array}{ll} person(pers(X_1, X_2)) & \Leftarrow name(X_1), age(X_2) \\ adult(X_1) & \Leftarrow person(X_1),\ \bigwedge_{0 \le i \le 17} X_1.(pers, 2) \neq i \\ valid(vote(X_1, Y)) & \Leftarrow q(X_1, [\,]) \\ q(X_1, X_2) & \Leftarrow q(X_1, :: (Y, X_2)), X_1 \neq Y.(vote, 1) \\ q(X_1, X_2) & \Leftarrow adult(X_1), votes(X_2) \end{array}$$

intending to only allow adults to submit a vote. Here, the first argument of q is a person instead of a vote. Then the second clause is instantiated for constructor $pers$ with substitution $X_1 \mapsto pers(X_1, X_2)$. Together with the first clause, we obtain:

$$adult(pers(X_1, X_2)) \Leftarrow name(X_1), age(X_2), \bigwedge_{0 \le i \le 17} pers(X_1, X_2).(pers, 2) \neq i$$

or simplified: $adult(pers(X_1, X_2)) \Leftarrow name(X_1), age(X_2), \bigwedge_{0 \le i \le 17} X_2 \neq i$ □

Theorem 3. *Let $\mathcal{C}$ denote a finite set of $\mathcal{H}_1$-clauses. Let $\overline{\mathcal{C}}$ denote the set of all clauses obtained from $\mathcal{C}$ by adding all clauses according to the resolution, splitting and propagation rules. Then the subset $\mathcal{N}$ of all normal clauses in $\overline{\mathcal{C}}$ is equivalent to $\mathcal{C}$, i.e., $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\mathcal{N}}$ for every predicate p occurring in $\mathcal{C}$.*

Proof. The proof follows the same lines as the proof of the corresponding statement in [11]: every clause added to $\mathcal{C}$ by means of resolution, splitting or propagation is implied by the set of $\mathcal{H}_1$-clauses in $\mathcal{C}$. Therefore for every predicate p, $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\overline{\mathcal{C}}}$. In the second step one verifies that every fact $p(t)$ which can be deduced by means of the clauses in $\overline{\mathcal{C}}$ can also be deduced in at most as many steps with clauses from $\mathcal{N}$ alone. This completes the proof. □

For the proof of termination of $\mathcal{H}_1$-normalization we consider *families* of clauses which (semantically) only differ in their constraints, and conceptually replace them by single clauses whose constraints are disjunctions of (conjunctions of) disequalities. Two clauses $h \Leftarrow \alpha_1, \phi_1$ and $h \Leftarrow \alpha_2, \phi_2$ belong to the same family if α_1 and α_2 contain the same set of literals.

To enforce termination of $\mathcal{H}_1$-normalization it suffices not to add clauses that are already *subsumed* by the current set of clauses. A clause $h \Leftarrow \alpha_0, \phi_0$ is subsumed by a set of clauses $h \Leftarrow \alpha_i, \phi_i, i \ge 1$, if all clauses belong to the same family and ϕ_0 implies the disjunction $\bigvee_{i \ge 1} \phi_i$.

Theorem 4. *Let $\mathcal{C}$ denote a finite set of $\mathcal{H}_1$-clauses. Let $\overline{\mathcal{C}}$ denote the set of clauses obtained from $\mathcal{C}$ by adding all clauses according to the resolution, splitting and propagation rules that are not subsumed by the current set of clauses. Then $\overline{\mathcal{C}}$ is finite.*

Proof. Since the number of predicates as well as the number of constructors is finite, there are only finitely many distinct heads of clauses. Also, the number of literals occurring in preconditions is bounded since new literals $p(t)$ are only added for *subterms* t of terms already present in the original set $\mathcal{C}$ of clauses. Therefore, the number of occurring families of clauses is finite. For each family f let $\psi_{\mathcal{C},f}$ denote the disjunction of constraints of clauses of $\mathcal{C}$ which belong to f. Each clause that is added to $\mathcal{C}$ extends one of the finitely many constraints $\psi_{\mathcal{C},f}$ to $\psi_{\mathcal{C},f} \vee \phi$ for a conjunction of disequalities ϕ. The number of variables in each constraint $\psi_{\mathcal{C},f}$ is bounded. Resolution with normal clauses does not introduce new variables, and propagation steps always produce normal clauses, which contain at most as many variables as the maximum arity in Σ.

In order to show that the resulting disjunctions eventually are implied, we recall that in every normalization step, the terms in constraints may grow — but the lengths of paths in path expressions remain bounded. Therefore, the number of all possibly occurring path expressions is finite.

In [11] we have shown that for each sequence ψ_i of conjunctions of (ordinary) *term* disequalities over finitely many plain variables, the disjunction $\bigvee_{i=1}^{m} \psi_i, m \geq 1$, eventually becomes *stable*, i.e., there exists some M such that $\bigvee_{i=1}^{m} \psi_i = \bigvee_{i=1}^{M} \psi_i$ for all $m \geq M$. This also holds true if we use finitely many path expressions instead of variables. Therefore a disjunction $\bigvee_{i=1}^{m} \psi_i, m \geq 1$, also eventually becomes stable if each ψ_i is a conjunction of general term disequalities if the set of occurring path expressions is finite.

We conclude that eventually, every newly added clause is subsumed — implying that the modified normalization procedure terminates with a finite set of clauses $\overline{\mathcal{C}}$. □

By Theorem 3 and Theorem 4, $\mathcal{H}_1$-normalization constitutes a sound and complete procedure which constructs for every finite set $\mathcal{C}$ of $\mathcal{H}_1$-clauses with path disequalities an equivalent finite set $\mathcal{N}$ of normal clauses within finitely many steps. By Lemma 2, $\mathcal{N}$ can in turn be transformed into an equivalent finite set $\mathcal{A}$ of automata clauses, for which by Corollary 1 emptiness can be decided for every predicate. Altogether this proves our main result:

Theorem 5. *Assume that $\mathcal{C}$ is a finite set of $\mathcal{H}_1$-clauses with general term disequality constraints. Then a finite set $\mathcal{A}$ of automata clauses can be effectively constructed such that for every predicate p of $\mathcal{C}$, $[\![p]\!]_{\mathcal{C}} = [\![p]\!]_{\mathcal{A}}$. In particular, it is decidable whether or not $[\![p]\!]_{\mathcal{C}}$ is empty.* □

6 Conclusion

We showed that finite sets of $\mathcal{H}_1$-clauses with path disequalities can be effectively transformed into finite tree automata with path disequalities. By that, we have provided a procedure to decide arbitrary queries to predicates defined by $\mathcal{H}_1$-clauses. From the proof of termination, however, little information could be extracted about the behavior of the method on realistic examples. It is a challenging open problem to provide explicit upper or nontrivial lower bounds to our algorithms.

$\mathcal{H}_1$-clauses can be used to approximate general Horn clauses and are therefore suitable for the analysis of term-manipulating programs such as cryptographic protocols.

Beyond unconstrained $\mathcal{H}_1$-clauses, $\mathcal{H}_1$-clauses with path disequalities additionally allow to approximate notions such as freshness of nonces or keys directly and also to some extent, *negative* information extracted from failing checks. It remains for future work to provide a practical implementation of our methods and to evaluate it on realistic protocols.

Acknowledgements. We thank Florent Jacquemard for a lively discussion, many helpful remarks and in particular, for pointing out reference [5].

References

1. Baskar, A., Ramanujam, R., Suresh, S.P.: Knowledge-based modelling of voting protocols. In: Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2007, pp. 62–71. ACM, New York (2007)
2. Comon, H., Jacquemard, F.: Ground Reducibility and Automata with Disequality Constraints. In: Enjalbert, P., Mayr, E.W., Wagner, K.W. (eds.) STACS 1994. LNCS, vol. 775, pp. 151–162. Springer, Heidelberg (1994)
3. Comon, H., Jacquemard, F.: Ground reducibility is exptime-complete. In: LICS 1997: Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, pp. 26–34. IEEE Computer Society, Washington, DC, USA (1997)
4. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
5. Godoy, G., Giménez, O., Ramos, L., Àlvarez, C.: The hom problem is decidable. In: STOC, pp. 485–494 (2010)
6. Goubault-Larrecq, J.: Deciding H1 by resolution. Information Processing Letters 95(3), 401–408 (2005)
7. Goubault-Larrecq, J., Parrennes, F.: Cryptographic Protocol Analysis on Real C Code. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 363–379. Springer, Heidelberg (2005)
8. Nielson, F., Nielson, H.R., Seidl, H.: Normalizable Horn Clauses, Strongly Recognizable Relations, and Spi. In: Hermenegildo, M.V., Puebla, G. (eds.) SAS 2002. LNCS, vol. 2477, pp. 20–35. Springer, Heidelberg (2002)
9. Reuß, A., Seidl, H.: Bottom-Up Tree Automata with Term Constraints. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 581–593. Springer, Heidelberg (2010)
10. Seidl, H., Neumann, A.: On Guarding Nested Fixpoints. In: Flum, J., Rodríguez-Artalejo, M. (eds.) CSL 1999. LNCS, vol. 1683, pp. 484–498. Springer, Heidelberg (1999)
11. Seidl, H., Reuß, A.: Extending H1-clauses with disequalities. Information Processing Letters 111(20), 1007–1013 (2011)
12. Weidenbach, C.: Towards an Automatic Analysis of Security Protocols in First-Order Logic. In: Ganzinger, H. (ed.) CADE 1999. LNCS (LNAI), vol. 1632, pp. 314–328. Springer, Heidelberg (1999)