OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author’s version published in:
http://oatao.univ-toulouse.fr/22649

Official URL
DOI : https://doi.org/10.12694/scpe.v19i2.1351


Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr
Satisfiability of general intruder constraints with and without a set constructor

Tigran Avanesov\textsuperscript{a}, Yannick Chevalier\textsuperscript{b},
Michael Rusinowitch\textsuperscript{c}, Mathieu Turuani\textsuperscript{c}

\textsuperscript{a} SnT, Université du Luxembourg, Luxembourg
\textsuperscript{b} Université Paul Sabatier & IRIT Toulouse, France
\textsuperscript{c} INRIA Nancy–Grand Est & LORIA, 54600 Villers-lès-Nancy, France

Keywords: ACI Deducibility constraints Dolev–Yao deduction system Multiple intruders Security

1. Introduction

Detecting flaws in security protocol specifications under the perfect cryptography assumption in the Dolev–Yao intruder model is an approach that has been extensively investigated in recent years.

\url{http://dx.doi.org/10.1016/j.jsc.2016.07.009}
years (Armando et al., 2014; Groza and Minea, 2013; Viganò, 2012; Meadows, 2011; Blanchet, 2009; Guttman, 2007; Arapinis and Duflot, 2007; Turuani, 2006). In particular, symbolic constraint solving has proved to be a very successful approach in the area. It amounts to expressing the possibility of mounting an attack, e.g. the derivation of a secret, as a list of steps where for each step an instance of the message pattern awaited according to the protocol has to be derived from the current intruder knowledge. These steps correspond in general to the progression of the protocol execution, up to the last one which is the secret derivation.

Enriching the standard Dolev–Yao intruder model with different equational theories (Comon-Lundh and Shmatikov, 2003; Comon-Lundh, 2004; Basin et al., 2005; Chevalier and Rusinowitch, 2010; Baskar et al., 2010; Escobar et al., 2011) like exclusive OR, modular exponentiation, Abelian groups, etc. (Liu and Lynch, 2011; Erbatur et al., 2011; Malladi, 2012; Chevalier et al., 2005; Küsters and Truderung, 2008) helps to find flaws that could not be detected considering free symbols only. A particularly useful theory is the theory of an ACI operator (that is associative, commutative and idempotent) since it allows one to express sets in cryptographic protocols.

Up to the exception of Mazaré (2005), all proposed algorithms rely on two strong assumptions about the constraints to be processed:

- knowledge monotonicity, reflecting the fact that the intruder could see everything that occurred before and forgot nothing;
- variable origination, reflecting that each variation in the protocol is introduced by the intruder.

Constraints satisfying these hypotheses are called well-formed constraints in the literature. Well-formed constraints are sufficient to solve security problems in the standard case where a single Dolev–Yao intruder is assumed. However, we will see that in some situations it can be quite useful to relax these hypotheses and consider general constraints, that is constraints without the restrictions above. General constraints naturally occur when considering security problems involving several non-communicating Dolev–Yao intruders (see § 2.1). Note that if intruders can communicate during protocol execution, the model becomes attack-equivalent to one with a unique Dolev–Yao intruder (Syverson et al., 2000).

A discussion on a multiple non-collaborating attackers model as well as interesting examples can be found in Fiazza et al. (2012).

1.1. Contributions of the paper

First, we will show that as for the standard case, in this more general framework it is still possible to derive an NP decision procedure for detecting attacks on a bounded number of protocol sessions (Sections 5, 4). Second, our result extends previous ones by allowing non-atomic keys (which is an important feature for protocol design as it is common to build symmetric keys from shared secrets) and the usage of an associative commutative idempotent operator (Sections 3, 4) that can be used for instance to model sets of nodes in XML document (see § 2.2). This extension of well-formed constraint systems may seem trivial but a third contribution of this paper is to demonstrate this is not the case by considering subterm deduction systems which are akin to the Dolev–Yao deduction system, but in which the equational theory can be any subterm convergent one. Whereas the decidability of well-formed constraint systems for subterm deduction systems is well known, see e.g. Baudet (2005), we prove in Appendix A that the satisfiability of general constraint systems is not decidable for subterm deduction systems. Finally we will sketch several applications of our results to security analysis in Section 2.

1.2. Related work

The decision procedure for satisfiability of well-formed constraint systems can be used to decide the insecurity of cryptographic protocols with a bounded number of sessions (Rusinowitch and Turuani, 2003). In this domain, several works deviated from the perfect cryptography assumption and started to consider algebraic properties of functional symbols. For example properties of XOR operator and exponentiation were considered in Lynch and Meadows (2004), Chevalier et al. (2005, 2008),
Dougherty and Guttman (2013) and together with homomorphic symbol in Delaune (2006). A class of monoidal equational theories was studied in Delaune et al. (2008) including the ACUI theory which is quite similar to the ACI theory discussed in the present work, but the absence of a unit element in ACI seems to put the ACI theory out of scope of their method. Moreover Delaune et al. (2008) do not show the NP complexity of their procedure. We note that in Chevalier et al. (2007) the authors showed the NP-completeness for the AC theory, while Dolev–Yao deduction system was extended with a rule $x \cdot y \rightarrow x$, where $\cdot$ is an AC symbol. We will examine the ACI theory within the similar deduction system. Note also that some algebraic properties (like associative and commutative symbol) make the insecurity problem undecidable (Bursuc et al., 2007).

All the decidability results mentioned above consider systems of constraints with two restrictions namely knowledge monotonicity (the left-hand side of a constraint representing the current knowledge of the intruder is included into the left-hand side of the next one) and variable origination (variable appears first in the right-hand side of some constraint): this limitation is not impeding the solution of usual protocol insecurity problems since the constraints generated with an active Dolev–Yao intruder are of the required type. An attempt to swerve from well-formeded constraints was made in Mazaré (2005) where the knowledge monotonicity was relaxed to define “quasi well-formed” whose satisfiability is proven to be in NP. This result was later extended in Mazaré (2006) to general constraint systems but under the provision that all keys are atomic, an hypothesis that permits to guess at the start of the procedure which keys are available at which step to the intruder. However to our knowledge no extension of Dolev–Yao deduction system to non-atomic symmetric key or to algebraic properties has been shown decidable for general constraint systems. Moreover, satisfiability of well-formed constraints with the ACI theory was not considered before. We have presented our results without proofs in a conference paper (Avanesov et al., 2010). In this long version we give detailed proofs. We believe that they can be useful for readers that want to study general constraints for other intruder theories.

2. Motivating examples

2.1. Protocol analysis with several intruders

Security analysis of a protocol is usually conducted in the worst possible case: a single intruder (or a coalition of several intruders, but able to share information faster than normal communications) controls all the network. However some protocols, especially routing protocols, are designed to work through different, non-communicating sub-networks. A relevant analysis of such protocols must take into account that the security of the protocol depends on the (correct) assumption that some communications are not possible. In these cases one needs to consider a less powerful intruder, or rather a collection thereof, that still controls some parts of the network but remains completely ignorant of what happens in other parts.

Suppose several agents ($A, B \ldots$, see Fig. 1) execute a protocol i.e. a specified finite sequence of message emissions or receptions. Due to their (long distance) layout they have to transmit data through routers ($1, 2, 3 \ldots$). The routing tables of all honest routers/agents are static (messages follow always the same path). Some routers ($2, 5, 7$) may be compromised: an intruder managed to install a device controlling input and output of the router or implanted there his malicious code. A message circulated via such an untrusted channel (e.g. $DB$) is consumed by the corresponding compromised device (local intruder) ($7$) thereby increasing his knowledge. Moreover, a local intruder can forge and emit to an endpoint ($C, B, D$) of any channel he controls ($BD, DB, DC$) any message he can build using the content of his memory and some available transformations specified by a deduction system. Because of the network topology malicious routers have no means to communicate (there are no links between them, neither direct nor via other routers), but at some point the intruder can gather the knowledge of all the compromised routers (by physically collecting devices or reading their memory).

In this framework the Coordinated attack problem is to decide whether it is possible to initially give instructions to compromised routers (e.g. by reprogramming malicious devices) to force an execution such that honest agents following the protocol will reveal some secret data to the intruder (i.e. the intruder can deduce it using information collected at the end from all local intruders).
2.1.1. The coordinated attack problem
We give here a semi-formal description of the problem and some directions to solve it. A formal model can be found in Avanesov et al. (2010) or Avanesov (2011).

First we introduce some notations and definitions that will be detailed in the technical part of the paper (§ 3).

2.1.1.1. Messages  We consider messages as first-order terms built from a set of function symbols (such as encryption, pairing, etc.), a set of constants $\mathcal{A}$ (representing elementary pieces of data: texts, public keys, names of agents, etc.) and a set of variables $\mathcal{X}$. Let $\mathcal{T}$ be the set of all possible terms. For a term $t$ we denote by $\text{Vars}(t)$ the set of all variables in $t$. A term $t$ is a ground term, if $\text{Vars}(t) = \emptyset$. The set of ground terms is denoted by $\mathcal{T}_g$.

We define a substitution $\sigma = \{ x_i \mapsto t_i, \ldots, x_k \mapsto t_k \}$ (where $x_i \in \mathcal{X}$ and $t_i \in \mathcal{T}$) to be the mapping $\sigma: \mathcal{T} \rightarrow \mathcal{T}$, such that $\sigma t$ is a term obtained by replacing, for all $i$, each occurrence of a variable $x_i$ by the corresponding term $t_i$. If $T \subseteq \mathcal{T}_g$, then by definition $T_{\sigma} = \{ t_{\sigma} : t \in T \}$. A substitution $\sigma$ is ground if for any $i \in \{1, \ldots, k\}$, $t_i$ is ground.

2.1.1.2. Agents  We will call honest communicating parties agents. Every agent is identified by its name. We denote the set of agent names by $A$.

2.1.1.3. Channels  Any two agents $a$ and $b$ communicate through a directed channel $a \rightarrow b$ implemented as a queue. The sent messages are stored in a queue to be processed in order of arrival.

2.1.1.4. Agents’ behavior  Each agent has a specific sequence of actions to execute. Every action is either of reception type $?_f r$ (expecting a message $r$ from agent $f$) or emission type $!_s s$ (sending a message $s$ to an agent with name $t$). Here $r$ and $s$ are terms representing message patterns. For example, if the sequence of actions of agent $a$ is $?_b \text{aenc}(X, pk_a),!_c \text{aenc}(X, pk_c)$ (where $X$ is a variable and $pk_a, pk_c$ are constants denoting a public key of $a$ and a public key of $c$ correspondingly and $\text{aenc}(u, v)$ is an asymmetric encryption of $u$ with key $v$) then $a$ waits until he/she receives some message $r$ matching a pattern $\text{aenc}(X, pk_a)$ on a channel $b \rightarrow a$ and sends a message $\text{aenc}(X \sigma, pk_b)$ on channel $a \rightarrow c$, where $\sigma$ is a substitution satisfying $t = \text{aenc}(X, pk_c)\sigma$. For instance, if $a$ receives $\text{aenc}(s, pk_a)$, i.e. some atomic value $s$ asymmetrically encrypted with a public key of $a$ $pk_a$, then the corresponding substitution $\sigma$ is $\{ X \mapsto s \}$ and $a$ sends to $c$ the value of $s$ reencrypted with public key $pk_c$ of $c$. On the other hand, it could be $\sigma = \{ X \mapsto \text{pair}(b, s) \}$ if $b$ sent $\text{aenc}(\text{pair}(b, s), pk_a)$. Note that if the message does not match the expected pattern, the agent stops this execution. Once a variable is instantiated on a reception, its value can be used in the following agent actions.

Fig. 1. Untrusted routers.
2.1.1.5. Intruder model. We assume that some communication channels are controlled by \(N\) local intruders from a set \(I = \{ I_i \}_{i=1}^{\ldots,N}\) and no channel is controlled by more than one intruder. Given some channel \(c\), we denote by \(i(c)\) the local intruder that controls \(c\) when there is one.

Every intruder \(I\) is given some initial knowledge \(K_i^0\) that is a set of ground terms. Once an agent sends a message via a channel controlled by an intruder, the intruder reads it and blocks it. Reading the message means extending intruder's current knowledge with this message. An intruder controlling a channel can generate a message from his knowledge using deduction rules and send it to its endpoint.

We now specify the intruder capabilities.

**Definition 2.1.** A rule is a tuple of terms written as \(s_1, \ldots, s_k \rightarrow s\), where \(s_1, \ldots, s_k, s\) are terms. A deduction system \(D\) is a set of rules.

As an example of deduction system, we refer to a version of the classical Dolev–Yao deduction system (DY) presented in Table 1.

From now to the end of this section rules are assumed to belong to a fixed deduction system \(D\) and that terms are evaluated modulo a congruence on terms \(\equiv_E\) by an equational theory (a set of equations between terms together with the properties of equality) \(E\).

Given two sets of ground terms \(E, F\) and a rule \(l \rightarrow r\), we write \(E \rightarrow_{l \rightarrow r} F\) iff \(F = E \cup \{s\}, s \equiv_E r\) and \(l \subseteq E\), where \(l\) is a set of terms. We write \(E \rightarrow F\) iff there exists rule \(l \rightarrow r\) such that \(E \rightarrow_{l \rightarrow r} F\).

**Definition 2.2.** A derivation \(D\) is a finite sequence of finite sets of ground terms \(E_0, E_1, \ldots, E_n\) such that \(E_0 \rightarrow E_1 \rightarrow \cdots \rightarrow E_n\), where \(E_i = E_{i-1} \cup \{t_i\}\), for all \(i \in \{1, \ldots, n\}\). A term \(t\) is derivable modulo an equational theory \(E\) from a set of terms \(E\) iff there exists a derivation \(D = E_0, \ldots, E_n\) such that \(E_0 = E\) and there exists \(s \in E_n\) such that \(s \equiv_E t\). We denote by \(\text{der}_E(E)\) the set of terms derivable from \(E\) modulo \(E\). A set of terms \(T\) is derivable from \(E\), iff \(T \subseteq \text{der}_E(E)\).

When the equational theory \(E\) is clear from context we write \(\text{Der}(E)\) instead of \(\text{der}_E(E)\), and similarly omit any reference to it. Informally, a local intruder may build new messages from the messages he knows by applying the deduction rules (e.g. concatenate two messages he knows). At each step a local intruder having recorded messages \(t_1, \ldots, t_n\) can send a message \(m\) if \(m \in \text{Der}(\{t_1, \ldots, t_n\})\).

2.1.1.6. Protocol execution. Given a set of protocol participants (agents) with their sequences of actions to execute, communication channels and local intruders each one with some initial knowledge, the course of events develops as follows. Each agent access channels connecting her to other agents. To send a message to Bob Alice inserts the message in the queue representing the channel, and to receive it Bob takes the first message in the queue and matches it against the pattern of the awaited message. An intruder Charlie controlling the channel may add, remove, read or reorder as he wishes messages in the queue.

2.1.1.7. Offline communication. At some execution point initially specified, the current knowledge of all local intruders is shared in order to derive a secret which probably they cannot deduce separately. We consider our intruder model is reasonable for covering situations where offline interactions between intruders are time-consuming and may be detected and therefore they take place when the protocol is over, when the honest agents have finished to execute their roles.
2.1.1.8. Coordinated attack problem  We are now able to define what is an attack by several local intruders on a protocol.

Coordinated attack problem (CAP)

**Input:** A finite set of agents $A$, list of actions per agent $\{(a, l_a)\}_{a \in A}$, a set of local intruders $I = \{(l_i)_{i=1,\ldots,N}\}$ each with initial knowledge $K^0_{l_i}$, and a partial function $\tau$ that assigns at most one intruder $\tau(c)$ to each channel $c$ and some sensitive data given as a ground term $s$.

**Output:** Yes iff there is a sequence of actions performed by agents and local intruders such that, at some point, from the union of intruders knowledge it is possible to derive $s$.

2.1.2. Reduction of the CAP to the satisfiability of a constraint system

We sketch in this section a non-deterministic reduction of the coordinated attack problem to the satisfiability of a constraint system. Briefly put, this reduction guesses an ordering of messages sent and received by the agents. Each intruder diverts all the messages put in a queue he has access to, and all messages received by agents are actually sent by the intruder. A guessed execution may be infeasible if the algorithm guesses an ordering in which an intruder who must send a message is unable to construct a message matching the awaited message pattern. We thus introduce constraints imposing that given a set $E$ of messages known by an intruder, he has to construct a message matching a pattern $t$.

**Definition 2.3.** Let $\mathcal{E}$ be an equational theory, $E$ be a set of terms and $t$ be a term. We define the couple $(E, t)$ denoted $E \triangleright t$ to be a constraint modulo $\mathcal{E}$. A constraint system modulo $\mathcal{E}$ is a set

$$S = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$$

where $n$ is a non-negative integer and $E_i \triangleright t_i$ is a constraint modulo $\mathcal{E}$ for $i \in \{1,\ldots,n\}$.

A ground substitution $\sigma$ is a model of constraint modulo $\mathcal{E}$ $E \triangleright t$ (or $\sigma$ satisfies this constraint) if $t\sigma \in \text{Der}(E\sigma)$. A ground substitution $\sigma$ is a model of a constraint system $S$ modulo $\mathcal{E}$ if it satisfies all the constraints of $S$. In order to simplify notations, when the equational theory $\mathcal{E}$ is clear from context we will omit any reference to it.

To express the problem with constraints we first guess an order in which the agents execute their actions since several interleavings are possible. Once the order is fixed, we build the constraint system $S$ (initially empty) incrementally by progressing in the action list:

- **Agent $a$ executes $\{b\}t$ on channel $a \rightarrow b$:**
  - if $a \rightarrow b$ is controlled by $I$ we increase the knowledge of Intruder $l$ with $t$ by setting $K_l := K_l \cup \{t\}$;
  - otherwise we put $t$ directly in the channel queue.
- **Agent $a$ executes $\{b\}t$ on channel $b \rightarrow a$:**
  - if $b \rightarrow a$ is controlled by $I$ we add a constraint $K_l \triangleright t$ into $S$ expressing the fact that Intruder $l$ knowing $K_l$ must send a message compatible with the expected message pattern $t$;
  - otherwise we must ensure that the first message $m$ in the queue of $b \rightarrow a$ is compatible with the expected pattern $t$, i.e. $m\sigma = t\sigma$, where $\sigma$ is a ground substitution satisfying all other constraints in $S$. For the case of DY deduction system we can encode this equation by a constraint $\{\text{enc}(m, k)\} \triangleright \text{enc}(t, k)$ for some atom $k^1$ since we can prove they admit the same solutions. We add the constraint to $S$ and remove $m$ from the queue.

Finally, we add $(\bigcup_{l \in I} K_l) \triangleright s$ to $S$ which expresses the offline communication phase: Secret $s$ can be deduced from the union of local intruders knowledge.

It is clear that if there exists a substitution $\sigma$ that satisfies $S$ then CAP has a solution. Conversely an attack for the CAP orders and instantiates the message patterns in the description of the protocol with a substitution $\sigma$, which satisfies the constraint system constructed as above.

---

1 For different deduction systems and equational theories the encoding may vary.
We give an example below and will show in Sections 3 and 4 that the satisfiability of constraint systems is in NP in the case of the Dolev–Yao deduction theory and its extension with an associative-commutative idempotent operator. This kind of operator can be useful to model messages in XML format as is shown in next subsection.

2.1.3. Example

Suppose three agents $a$, $b$, $c$ execute a protocol whose normal execution is shown in Fig. 2(a). Each agent follows his sequence of actions shown in Fig. 2(b). Agent $a$ wants to send a secret $s$ to $b$. He asks $b$ to send her public key $k_b$ to $c$, that will forward it to $a$. Nonce $n$ permits $a$ to correlate the message from $c$ with his request.

The agents are connected by three directed channels: channel $a \rightarrow b$ is controlled by local intruder $I_1$, channel $b \rightarrow c$ is controlled by local intruder $I_2$, while channel $c \rightarrow a$ is secure (free from intruders).

We assume that the two local intruders have as initial knowledge a pair of fresh public/private keys and public keys of all participants: $K_{I_1}^0 = K_{I_2}^0 = \{k_1, \text{priv}(k_i), k_a, k_b, k_c\}$.

The question we ask is whether the local intruders $I_1$ and $I_2$ can cooperate in such a way that joining their final knowledge they can derive $s$, although they have no means to communicate during the protocol execution.

Following the sequence of actions shown in Fig. 2(a), we obtain the constraint system shown in Fig. 2(c). First $a$ sends $n$ which is intercepted by $I_1$. The knowledge of $I_1$ becomes $K_{I_1} = K_{I_1}^0 \cup \{n\}$. Then $I_1$ must build from his knowledge a message that can be accepted by $b$. This is expressed in (1).

Once $b$ accepts this message, he executes his next action: emission of a message on channel $b \rightarrow c$. This message is intercepted by $I_2$ and his knowledge becomes $K_{I_2} = K_{I_2}^0 \cup \{\text{aenc}(\text{pair}(Y, k_b), k_c)\}$. Now, $I_2$ must send a message to $c$ on behalf of $b$ (2). Then $c$ accepts a message sent by $I_2$, it sends his reply to $a$ which should be compatible with the expected pattern: $\text{aenc}(Z, k_{ac}) = \text{aenc}(\text{pair}(n, X), k_{ac})$. This fact is encoded into (3). Then $a$ sends a message to $b$ intercepted by $I_1$ ($K_{I_1} = K_{I_1}^0 \cup \{n, \text{aenc}(s, X)\}$) and $I_1$ must generate a message acceptable by $b$ (4). Finally, in (5) we specify that from the common knowledge of $I_1$ and $I_2$ one may deduce a secret $s$.

The obtained constraint system has no solution as well as any other constraint systems generated from different interleaving of actions executed by the agents. We conclude that there is no coordinated attack under the given hypothesis. On the other hand, if $I_1$ and $I_2$ can communicate, or the same intruder controls both channels $a \rightarrow b$ and $b \rightarrow c$, a simple attack can be mounted: once the intruder receives $n$ on channel $a \rightarrow b$, he generates the message $\text{aenc}(\text{pair}(n, k_i), k_c)$ and sends it via $b \rightarrow c$; $c$ accepts it and sends message $\text{aenc}(\text{pair}(n, k_i), k_{ac})$ to $a$ which matches the expected pattern. Thus, $a$ will send via $a \rightarrow b$ a message $\text{aenc}(s, k_i)$ which is intercepted by the intruder and can be decomposed using $\text{priv}(k_i)$. The intruder can also finalize the actions of $b$ by sending him, e.g., $n$ and then $\text{aenc}(s, k_b)$.

Fig. 2. Example illustrations.

\[ l_a : \{n, \text{aenc}(\text{pair}(n, X), k_{ac})\} \quad l_b : \{\text{aenc}(s, X)\} \quad l_c : \{\text{aenc}(Z, k_{ac})\} \]

(a) Normal execution + Intruders

(b) Sequence of actions per agent

\begin{align}
K_{I_1}^0 \cup \{n\} & \vdash Y \\
K_{I_2}^0 \cup \{\text{aenc}(\text{pair}(Y, k_b), k_c)\} & \vdash \text{aenc}(Z, k_c) \\
\{\text{aenc}(Z, k_{ac})\} & \vdash \text{aenc}(\text{pair}(n, X), k_{ac}) \\
K_{I_1}^0 \cup \{n, \text{aenc}(s, X)\} & \vdash \text{aenc}(s, k_b) \\
K_{I_1}^0 \cup \{n, \text{aenc}(s, X)\} \cup K_{I_2}^0 & \cup \{\text{aenc}(\text{pair}(Y, k_b), k_c)\} \vdash s
\end{align}
2.2. Attacks exploiting XML format of messages

We propose a way to model some attacks based on an XML-representation of messages. A different technique to handle this kind of attacks was presented in Chevalier et al. (2007) which considers multisets of XML nodes, while in this work the notion of set is taken as a basis.

We consider an e-shop that accepts e-cheques, and we suppose that it is presented by a Web Service using the SOAP protocol for exchanging messages.

It consists of two services:

1. the first exposes the list of goods for sale with their prices and processes the orders by accepting payment;
2. the second is a delivery service; it receives information from the first one about successfully paid orders, and sends the ordered goods to the buyer.

A simple scenario for ordering items is shown in Fig. 3. First, a client sends an order using the e-shop interface that consists of an item identifier, e-cheque, delivery address and some comments. Then, the first service of the e-shop checks whether the price of the ordered item corresponds to the received cheque. If it does, the service consumes the cheque and forwards the order to the stock/delivery service (without the used e-cheque). Stock and Delivery service prepares a parcel for the ordered item and sends it to given address. The comment is automatically printed on the parcel to give some information to the postman about, for example, delivery time or access instructions.

Suppose Alice has an e-cheque for 5€. She can buy a simple pen (with ItemID simple) but she likes very much a more expensive gilded one (with ItemID gilded). Can we help Alice to get what she wants for what she has?

Let us formalize the behavior of scenario players. We will write \((t_1 \cdots t_n)\) (or equivalently \((t_1, \ldots, t_n)\)) an XML message where \(t_1, \ldots, t_n\) are XML nodes. We model the fact that the nodes are not duplicated and their order is meaningless by assuming that \(\cdot\) is associative commutative and idempotent. Identifiers starting with a capital letter are considered as variables; numbers and identifier starting from lower-case letter are considered as constants. We model a delivery of item with some ItemID to address Address with comments Comments by the following message: \(\text{sig}((\text{ItemID} \cdot \text{Address} \cdot \text{Comments}), \text{priv}(k_s))\) — a message signature produced by e-shop, where \(k_s\) its public key and \(\text{priv}(k_s)\) is the corresponding private key, such that no one can produce this message except the shop. We abstract away from the procedure of checking the item price and will suppose that Shop Interface expects a 5€ e-cheque for Item “simple”. For simplicity we assume only two items.

We will use the same notation as in § 2.1 for emission and reception actions.
For Shop Interface we have:

\[ ?_{\text{Client}}(\text{simple} \cdot \text{cheque5} \cdot \text{IAddr} \cdot \text{IComm}); \]
\[ !_{\text{Delivery}}(\text{simple} \cdot \text{IAddr} \cdot \text{IComm}). \]

For Shop Stock/Delivery we have:

\[ ?_{\text{Interface}}(\text{DItemID} \cdot \text{DAddr} \cdot \text{DComm}); \]
\[ !_{\text{Client}} \text{sig} ((\text{DItemID} \cdot \text{DAddr} \cdot \text{DComm}), \text{priv} (k_s)). \]

Alice initially has:

- simple, gilded: identifiers of items;
- cheque5: an e-cheque for 5€;
- addr: her address;
- cmnts: residence digital code;
- \( k_s \): a public key of the shop.

Now we build a mixed constraint system (derivation constraints and equations) to know whether Alice can do what she wants:

\[
\begin{align*}
\{\text{gilded, simple, cheque5, addr, cmnts, } k_s\} & \triangleright \\
(\text{simple} \cdot \text{cheque5} \cdot \text{IAddr} \cdot \text{IComm}) & \quad (6) \\
(\text{simple} \cdot \text{IAddr} \cdot \text{IComm}) & \equiv_{\text{ACI}} (\text{DItemID} \cdot \text{DAddr} \cdot \text{DComm}) \\
\{\text{gilded, simple, cheque5, addr, cmnts, } k_s, \text{sig} ((\text{DItemID} \cdot \text{DAddr} \cdot \text{DComm}), \text{priv} (k_s))\} & \triangleright \\
\text{sig} ((\text{gilded} \cdot \text{addr} \cdot \text{DComm}), \text{priv} (k_s)) & \quad (8)
\end{align*}
\]

The constraint (6) shows that Alice can construct a message expected by the shop from a client. Constraint (7) represents a request from the first to the second service of the shop: the left-hand side is a message sent by the interface service and the right-hand side is a message expected by the stock/delivery subservice; \( \equiv_{\text{ACI}} \) means that these messages must be compatible (modulo ACI). The last constraint shows that from the received values Alice can build a message that models a delivery of item with ItemID gilded.

To solve it, we first get rid of syntactic equations by applying the most general unifier; and then of equations modulo ACI (\( t_1 =_{\text{ACI}} t_2 \)) by encoding them into a deducibility constraint (as it was done in § 2.1.2).

Then, one of the solutions is:

- \( \text{IAddr} \mapsto \text{addr} \)
- \( \text{IComm} \mapsto (\text{gilded} \cdot \text{cmnts}) \)
- \( \text{DItemID} \mapsto \text{gilded} \)
- \( \text{DAddr} \mapsto \text{addr} \)
- \( \text{DComm} \mapsto (\text{simple} \cdot \text{cmnts}) \)

From this solution we see that Alice can send an ill-formed comment (that contains two XML-nodes), and the Delivery service parser can choose an entry with ID gilded. An attack-request can look like this:

- \(<\text{ItemID}>\text{simple</ItemID}>\)
- \(<\text{Cheque}>\text{cheque5</Cheque}>\)
- \(<\text{Address}>\text{addr</Address}>\)
- \(<\text{Comments}>\text{cmnts</Comments}>\)
- \(<\text{ItemID}>\text{gilded</ItemID}>\)
The parser of the first service can return the value of the first occurrence of ItemID: <ItemID> simple</ItemID>, while the parser of the second one, being a different software, can return <ItemID> gilded</ItemID>.

This attack is possible if Alice constructs a request “by hand”, but a similar attack is probably feasible using an XML-injection: Alice when filling a request form enters instead of her comments the following string:

```
<Comments>
  <ItemID>gilded</ItemID>
</Comments>
```

and in the resulting request we get:

```
<ItemID>simple</ItemID>
<Cheque>cheque5</Cheque>
<Address>addr</Address>
<Comments>cmnts</Comments>
<ItemID>gilded</ItemID>
</Comments>
```

This kind of XML-injection attack was described in OWASP Foundation (2008).

3. Satisfiability of general DY + ACI constraint systems

In Section 2 we have shown how to reduce the problem of protocol insecurity in presence of several intruders to solving a system of general deducibility constraints. In this section we present a decision procedure for the satisfiability of general constraint systems where the Dolev—Yao deduction system is extended with some deduction rules for an associative-commutative-idempotent symbol (DY + ACI). We consider operators for pairing, symmetric and asymmetric encryptions, signature, hashing and an ACI operator that will be used as a set constructor.

As for the proof structure, after introducing the formal notations and some basic properties, the main steps to show the decidability are as follows:

1. We present an algorithm for solving the derivability problem in the DY + ACI model;
2. We prove that for checking intruder constraints satisfiability it is sufficient to consider normalized constraints and normalized substitutions;
3. We show that a satisfiable normalized constraint system admits at least one conservative solution, that is a substitution $\sigma$ that maps each variable of the constraint system to a set of subterms from the constraint system (instantiated with $\sigma$) and private keys;
4. We give a bound on the size of a conservative solution, and, as a consequence, we obtain decidability.

3.1. Formal introduction to the problem

First we would like to note that we omit the proofs of some statements that we find intuitive. In any case the detailed proofs may be found in Avanesov (2011) (particularly in Lemma 4, p. 33). Though the notions of terms and subterms are standard in the litterature, they are not necessarily the most convenient when working with equational theories. Accordingly, in this section and in addition to the standard notions, we introduce the sets of Subterms and Quasi-subterms of a term. The notion of subterm of a term $t$ is the standard one, whereas quasi-subterms are subterms of the flattening of $t$ by successive applications of associativity rules. The notions of subterms and quasi-subterms are equivalent on normalized terms, but one of the difficulty of handling an equational theory is to analyze their interplay when instantiating and normalizing a term modulo this equational theory. We also provide a few technical lemmas to address this question in this subsection.

**Definition 3.1.** Let $\mathcal{A}$ be a set of atoms, and $\mathcal{X}$ be a set of variables. Let $\mathcal{F}$ be the minimal set of functional symbols containing: binary symbols pair, enc, aenc, sig, apply, $\cdot_2$, unary symbols priv, $\cdot_1$, 
and for all \( i \in \mathbb{N}^+ \) an \( i \)-ary symbol \( \cdot_i \). The set of terms \( \mathcal{T}(\mathcal{A}, \mathcal{X}) \) is a minimal set satisfying: \( \mathcal{A} \subseteq \mathcal{T} \), \( \mathcal{X} \subseteq \mathcal{T} \), if \( t_1, \ldots, t_k \) are terms and \( f \) is a \( k \)-ary symbol from \( \mathcal{F} \) then \( f(t_1, \ldots, t_k) \in \mathcal{T} \). To be short we write \( \mathcal{T} \) instead of \( \mathcal{T}(\mathcal{A}, \mathcal{X}) \).

By \( \text{sig}(p, \text{priv}(a)) \) we mean a signature of message \( p \) with private key \( \text{priv}(a) \) that does not contain a message itself (the message \( p \) together with its signature with private key \( \text{priv}(a) \) can be modeled as pair \( (p, \text{sig}(p, \text{priv}(a))) \)).

We consider the following family of function symbols (indexed by their fixed arities): \( \bigcup_{i \in \mathbb{N}^+} \{ \cdot_i \} \). Symbol \( \cdot_i \) has a list of \( i \) terms as its arguments. Since the arity can be inferred from the number of arguments, we write \( \cdot(t_1, t_2) \) as a shortcut for \( \cdot_2(t_1, t_2) \), \( \cdot(t_1, t_2, t_3) \) as a shortcut for \( \cdot_3(t_1, t_2, t_3) \) and so on.

We denote \( \text{dom}(\sigma) \) the domain \( \{x_1, \ldots, x_n\} \) of a substitution \( \sigma = \{ x_1 \mapsto t_1, \ldots, x_n \mapsto t_n \} \) and \( \text{img}(\sigma) = \bigcup_{x \in \text{dom}(\sigma)} \{ x \sigma \} \) its image. The cardinality of a set \( P \) is denoted by \( |P| \).

We denote any element of \( \{\text{enc}, \text{aenc}, \text{pair}, \text{sig}, \text{apply}\} \). The root of a term \( t \in \mathcal{T} \) is denoted \( \text{root}(t) \) and is defined as follows:

\[
\text{root}(t) = \begin{cases} 
\text{bin}, & \text{if } t = \text{bin}(p, q) \\
\text{...}, & \text{if } t = \cdot(L) \\
\text{priv}, & \text{if } t = \text{priv}(p) \\
t, & \text{if } t \in \mathcal{X} \cup \mathcal{A}.
\end{cases}
\]

We denote the \( i \)-th term of a list of terms \( L \) as \( L[i] \) and we write \( t \in L \) if \( t \) occurs as an element of \( L \). We also define two binary relations \( \subseteq \) and \( \approx \) on lists as follows: \( L_1 \subseteq L_2 \) iff every element of \( L_1 \) is an element of \( L_2 \); \( L_1 \approx L_2 \) iff \( L_1 \subseteq L_2 \) and \( L_2 \subseteq L_1 \) (we use the same definition if \( L_1 \) or \( L_2 \) is a set).

In Definition 3.2 we present algebraic properties of the \( \cdot \) symbol (denoted as \( ACI \)), that are considered together with the Dolev–Yao deduction system in this section. In the rest of this paper deduction constraints are always modulo this \( ACI \) equational theory.

**Definition 3.2.** We define \( \equiv_{ACI} \) to be the congruence relation on \( \mathcal{T} \) generated by the equational theory \( ACI \):

\[
\{ \cdot(y) = y, \quad \cdot(y_1, y_1) = \cdot(y_1), \quad \cdot(y_1, y_2) = \cdot(y_2, y_1) \} \cup \\
\bigcup_{n=1}^{\infty} \{ \cdot(t_1, \ldots, t_k) = \cdot(t_{k+1}, \ldots, t_m), \cdot(t_{m+1}, \ldots, t_n) \} \}_{0 \leq k < m \leq n}.
\]

Since \( \cdot \) is variadic, the associativity property is expressed above as an infinite set of flattening rules.

**Definition 3.3.** The set of elements of a term \( t \in \mathcal{T} \) is denoted \( \text{elems}(t) \) and defined by:

\[
\text{elems}(t) = \begin{cases} 
\bigcup_{p \in L} \text{elems}(p) & \text{if } t = \cdot(L) \\
\{t\} & \text{otherwise}.
\end{cases}
\]

We extend \( \text{elems}(\cdot) \) to sets of terms or lists of terms \( T \) by \( \text{elems}(T) = \bigcup_{t \in T} \text{elems}(t) \).

**Example 1.** The set of elements of \( t = \cdot(a, \cdot(b, a, \text{pair}(a, b)), \text{pair}(\cdot(b, b), a)) \) is \( \text{elems}(t) = \{a, b, \text{pair}(\cdot(b, b), a), \text{pair}(a, b)\} \).

**Definition 3.4.** Let \( < \) be a strict total order on \( \mathcal{T} \) such that the question whether \( p < q \) can be answered in polynomial time.

**Definition 3.5.** The normal form of a term \( t \) (denoted by \( \text{nf}(t) \)) and a set of terms \( T \) (denoted by \( \text{nf}(T) \)) is defined in a mutually recursive way (on their size) by:

1. \( \text{nf}(t) = t \), if \( t \in \mathcal{X} \cup \mathcal{A} \);
2. \( \text{nf}(\text{bin}(t_1, t_2)) = \text{bin}(\text{nf}(t_1), \text{nf}(t_2)) \);
(3) \( \text{priv}(t)^\gamma = \text{priv}("t")^\gamma; \)

(4) \( \text{r}(L)^\gamma = \begin{cases} 
\cdot (L'), & \text{if } |\text{elems}(L)^\gamma| > 1 \text{ and } L' \approx \text{r}\text{elems}(L)^\gamma \\
\cdot (\text{r}(t_1), \text{r}(t_2)) & \text{and for all } i < j, L'[i] < L'[j]; \\
t, & \text{if } \text{r}\text{elems}(L)^\gamma = \{t'\} 
\end{cases} \)

(5) \( \text{r}T = \{ \text{r}t : t \in T \} \)

We say a term \( t \) is normalized iff \( t = \text{r}t^\gamma \).

We note right away that \( |T\sigma| \leq |T| \) and \( |\text{r}t^\gamma| \leq |T| \) and two terms are congruent modulo the ACI properties of \( \text{r}, " \) if they have the same normal form. In particular the following equalities hold:
\[
\text{r}(t, t')^\gamma = \text{r}t^\gamma, \text{r}(t_1, t_2) = \text{r}(t_2, t_1)^\gamma, \text{r}(\cdot (t_1, t_2), t_3) = \text{r}(\cdot (t_2, t_3), t_1) = \text{r}(\cdot (t_1, t_2), t_3) \in \text{r}(\cdot (t_1, t_2), t_3) .
\]

Note also that normalization is idempotent: \( \text{r}t^\gamma = \text{r}t^\gamma \). Moreover the equivalence relation \( \equiv_{\text{ACI}} \) is closed under substitution (see Baader and Nipkow, 1998), i.e. \( t_1 \equiv_{\text{ACI}} t_2 \) implies \( t_1\sigma \equiv_{\text{ACI}} t_2\sigma \). Therefore, we obtain \( t \equiv_{\text{ACI}} \text{r}t^\gamma \) and \( \sigma \equiv_{\text{ACI}} \text{r}\text{t}^\gamma \), and then \( \text{r}t\sigma = \text{r}\text{t}^\gamma \sigma = \text{r}\text{t} \equiv_{\text{ACI}} \text{r}\text{t}^\gamma \sigma = \text{r}\text{t}^\gamma \).

Given its importance we turn this last property into a lemma.

**Lemma 1.** For any term \( t \) and substitution \( \sigma \) we have \( \text{r}t^\gamma = \text{r}\text{t}^\gamma \).

**Example 2.** In Example 1 we have \( \text{r}t^\gamma = \cdot ([a, b, \text{pair}(a, b), \text{pair}(b, a)]) \).

We give some useful properties of \( \text{elems}(\cdot) \) function with respect to normalization.

**Lemma 2.** For any term \( t \), \( \text{r}\text{ elems}(t)^\gamma = \text{elems}("t")^\gamma \).

**Proof.** This statement is trivial, if \( t \neq \cdot (L) \). Otherwise, let \( t = \cdot (t_1, . . . , t_n) \).

(1) If \( \text{r}\text{elems}(t)^\gamma = \{p\} \) where \( p \neq \cdot (L_1) \).

Then \( \text{r}t^\gamma = p \) and then \( \text{elems}(\text{r}t^\gamma) = \text{elems}(p) = \{p\} = \text{r}\text{elems}(t)^\gamma \);

(2) If \( \text{r}\text{elems}(t)^\gamma = \{p_1, . . . , p_k\}, k > 1 \) where \( p_i \neq \cdot (L_1) \) for all \( i \).

Then \( \text{r}t^\gamma = \cdot (L) \) where \( L \approx \{p_1, . . . , p_k\} \).

That means that \( \text{elems}(\text{r}t^\gamma) = \bigcup_{p \in \{p_1, . . . , p_k\}} \text{elems}(p) = \{p_1, . . . , p_k\}. \)

**Lemma 3.** For any terms \( t_1, . . . , t_m \), we have \( \text{r}(\cdot (t_1^\gamma), . . . , t_m^\gamma)^\gamma = \text{r}(\cdot (t_1, . . . , t_m)^\gamma) \)

**Proof.** Follows from the definition of normal form and Lemma 2. \( \square \)

**Lemma 4.** For terms \( t_1, . . . , t_m \), we have:
\[
\text{elems}(\cdot (t_1^\gamma, . . . , t_m^\gamma)^\gamma) = \text{elems}(\cdot (t_1^\gamma, . . . , t_m^\gamma)) = \bigcup_{i=1, . . . , m} \text{elems}(t_i^\gamma)
\]

**Proof.** We get the first equality by applying Lemma 2: \( \text{elems}(\cdot (t_1^\gamma, . . . , t_m^\gamma)^\gamma) = \text{elems}(\cdot (t_1^\gamma, . . . , t_m^\gamma)) \)

and then from Definition 3.3 and Lemma 2 we have that \( \text{elems}(\cdot (t_1^\gamma, . . . , t_m^\gamma)) \) is a set of normalized terms. The second equality directly follows from Definition 3.3. \( \square \)

We introduce now quasi-subterms as subterms of terms flattened by applications of associativity rules.

**Definition 3.6.** The set of quasi-subterms of term \( t \) is defined as follows:
\[
\text{QSub}(t) = \begin{cases} 
\{t\}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\
\{t\} \cup \text{QSub}(t_1), & \text{if } t = \text{priv}(t_1); \\
\{t\} \cup \text{QSub}(t_1) \cup \text{QSub}(t_2), & \text{if } t = \text{bin}(t_1, t_2); \\
\{t\} \cup \bigcup_{p \in \text{elems}(L)} \text{QSub}(p), & \text{if } t = \cdot (L)
\end{cases}
\]
If $T$ is a set of terms then $QSub(T) = \bigcup_{t \in T} QSub(t)$. If $S = \{ E_i \triangleright t_i \}_{i=1, \ldots , n}$ is a constraint system then $QSub(S) = \bigcup_{t \in \bigcup_{i=1}^n E_i \cup \{ t_i \}} QSub(t)$.

**Example 3.** Referring to **Example 1**, we have

$$QSub(t) = \{ (b, a, \text{pair}(a, b)) \}.$$  

Note also that $QSub(QSub(t)) = QSub(t)$ and $s = \uparrow s^{-1}$ for all $s \in QSub(\uparrow t^{-1})$.

Let us now define the set $Sub(t)$ of subterms of a term $t$.

**Definition 3.7.** Let $t$ be a term. We define $Sub(t)$ as follows:

$$Sub(t) = \begin{cases} 
\{ t \}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\
\{ t \} \cup Sub(t_1), & \text{if } t = \text{priv}(t_1); \\
\{ t \} \cup Sub(t_1) \cup Sub(t_2), & \text{if } t = \text{bin}(t_1, t_2); \\
\{ t \} \cup \bigcup_{p \in \mathcal{L}} Sub(p), & \text{if } t = \cdot (L). 
\end{cases}$$

If $T$ is a set of terms, then $Sub(T) = \bigcup_{t \in T} Sub(t)$. If $S = \{ E_i \triangleright t_i \}_{i=1, \ldots , n}$ is a constraint system, we define $Sub(S) = \bigcup_{t \in \bigcup_{i=1}^n E_i \cup \{ t_i \}} Sub(t)$.

**Example 4.** Referring to **Example 1**, we have

$$Sub(t) = QSub(t) \cup \{ (b, a, \text{pair}(a, b)) \}.$$  

**Lemma 5.** For any term $t$ and substitution $\sigma$, $Sub(t\sigma) = Sub(t) \cup Sub(Vars(t)\sigma)$.

**Proof.** By induction on $|Sub(t)|$:

- $|Sub(t)| = 1$.
  - $t \in \mathcal{A}$. As $t\sigma = t$ and $\text{Vars}(t) = \emptyset$, the statement becomes trivial;
  - $t \in \mathcal{X}$. Then $Sub(t)\sigma = \sigma$, $\text{Vars}(t) = \{ t \}$, and as for any term $p$, $p \in Sub(p)$, we have $Sub(t\sigma) = \{ \sigma \} \cup Sub(t\sigma)$.
- Suppose that for any $t : |Sub(t)| < k (k \geq 1)$, the statement is true;
- Given a term $t : |Sub(t)| = k$, $k > 1$. Let us consider all possible cases:
  - $t = \text{bin}(t_1, t_2)$. Then $t\sigma = \text{bin}(t_1\sigma, t_2\sigma)$ and $\text{Vars}(t) = \text{Vars}(t_1) \cup \text{Vars}(t_2)$. Sub($t\sigma$) = $t\sigma$ \cup Sub($t_1\sigma$) \cup Sub($t_2\sigma$) = (as $|Sub(t_i)| < k$) = $\{ \sigma \} \cup Sub(t_1\sigma) \cup Sub(Vars(t_1)\sigma) \cup Sub(t_2\sigma) \cup Sub(Vars(t_2)\sigma)$ = $Sub(t)\sigma$ \cup Sub(Vars(t)\sigma);
  - $t = \text{priv}(t_1)$. The proof is the same as for the previous case;
  - $t = \cdot (t_1, \ldots , t_m)$. Then $t\sigma = \cdot (t_1\sigma, \ldots , t_m\sigma)$ and $\text{Vars}(t) = \bigcup_{i=1}^m \text{Vars}(t_i)$. Then we have Sub($t\sigma$) = $\{ \sigma \} \cup \bigcup_{i=1}^m \text{Sub}(t_i\sigma)$ = (as $|Sub(t_j)| < k$) = $\{ \sigma \} \cup \bigcup_{i=1}^m (\text{Sub}(t_i)\sigma \cup \text{Sub}(\text{Vars}(t_i)\sigma))$ = $Sub(t)\sigma$ \cup Sub(Vars(t)\sigma).  

The inclusion relation between elements (·), $QSub(·)$ and $Sub(·)$ is stated in **Lemma 6**.

**Lemma 6.** Let $t$ be a term. Then $\text{elems}(t) \subseteq QSub(t) \subseteq Sub(t)$

For normalized terms, the set of its subterms coincides with the set of its quasi-subterms:

**Lemma 7.** For any normalized term $t$, $QSub(t) = Sub(t)$.

**Proof.** By induction on $|Sub(t)|$. 

Lemma 8. For any term \( t \), \( \text{QSub}(⌜t\⌝) = \text{QSub}(⌜t\⌝) \).

Proof. By induction on \(|\text{Sub}(t)|\).

1. \(|\text{Sub}(t)| = 1\). Then \( t \in A \cup \mathcal{X} \), and thus, \( \text{QSub}(t) = \text{Sub}(t) = [t] \).
2. Suppose that for any \( t \) : \(|\text{Sub}(t)| < k (k > 1) \), the statement holds;
3. Given a term \( t \) : \(|\text{Sub}(t)| = k, k > 1 \). We need to show that \( \text{QSub}(t) = \text{Sub}(t) \).

\( t \in \{ \text{bin}(t_1, t_2), \text{priv}(t), \text{apply}(t, t_1, t_2) \} \), and thus \( \text{QSub}(t) \).

\( t = \text{bin}(t_1, t_2) \). Then \( \text{QSub}(\text{bin}(t_1, t_2)) = \{ t \} \cup \text{QSub}(t_1) \cup \text{QSub}(t_2) \) and \( \text{QSub}(\text{bin}(t_1, t_2)) = \{ t \} \cup \text{QSub}(t_1) \cup \text{QSub}(t_2) \).

\( t = \text{priv}(t_1) \). Then \( \text{QSub}(\text{priv}(t_1)) = \{ t \} \cup \text{QSub}(t) \).

\( t = \text{apply}(t, t_1, t_2) \). Then \( \text{QSub}(\text{apply}(t, t_1, t_2)) = \{ t \} \cup \text{QSub}(t_1) \cup \text{QSub}(t_2) \).

Lemma 9. For any term \( t \), \( \text{Sub}(⌜t\⌝) \subseteq \text{QSub}(t) \).

Proof. From Lemma 7 we have \( \text{Sub}(⌜t\⌝) \subseteq \text{QSub}(t) \). By Lemma 8 we obtain \( \text{QSub}(⌜t\⌝) = \text{QSub}(t) \) and by Lemma 6 we have \( \text{QSub}(t) \subseteq \text{Sub}(t) \). Thus, \( \text{Sub}(⌜t\⌝) \subseteq \text{QSub}(t) \subseteq \text{Sub}(t) \). □

3.2. Dolev–Yao deduction system modulo ACI

We define a Dolev–Yao deduction system modulo ACI (denoted \( \text{DY} + \text{ACI} \)). It consists of composition rules and decomposition rules, depicted in Table 2 where \( t_1, t_2, \ldots, t_m \in T \) (i.e., we consider an infinite system having a finite representation given in the table). Note that rule \( \text{enc}(t_1, t_2), t_3 \rightarrow t_1 \) and \( t_3 \equiv_{\text{ACI}} t_3 \) is equivalent to \( \text{enc}(t_1, t_2), t_3 \rightarrow t_1 \), since it is always possible to normalize term \( t_3 \) (and obtain \( t_3 \equiv_{\text{ACI}} t_3 \)) using, e.g., \( t_3, t_3 \rightarrow \text{pair}(t_3, t_3) \) and since \( \text{pair}(t_3, t_3) = \text{pair}(t_3, t_3) \) we can decompose it by obtaining \( \text{pair}(t_3, t_3) = t_3 \).

We introduce two restrictions to define a subset of valid ground terms. The first one follows from the semantics of the functional symbol \( \text{sig} \): its second argument must always be a private key. The second one is imposed in order to simplify our reasoning for solving constraint systems: we only consider atomic asymmetric private keys (this is not a real limitation for many applications).

### Table 2

<table>
<thead>
<tr>
<th>Composition rules</th>
<th>Decomposition rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1, t_2 \rightarrow \text{enc}(t_1, t_2) )</td>
<td>( \text{enc}(t_1, t_2), t_3 \rightarrow t_1 ) if ( t_3 \equiv_{\text{ACI}} t_3 )</td>
</tr>
<tr>
<td>( t_1, t_2 \rightarrow \text{aenc}(t_1, t_2) )</td>
<td>( \text{aenc}(t_1, t_2), \text{priv}(t_3) \rightarrow t_1 ) if ( t_3 \equiv_{\text{ACI}} t_3 )</td>
</tr>
<tr>
<td>( t_1, t_2 \rightarrow \text{pair}(t_1, t_2) )</td>
<td>( \text{pair}(t_1, t_2) \rightarrow t_1 )</td>
</tr>
<tr>
<td>( t_1, \text{priv}(t_2) \rightarrow \text{sig}(t_1, \text{priv}(t_2)) )</td>
<td>( \text{pair}(t_1, t_2) \rightarrow t_1 )</td>
</tr>
<tr>
<td>( t_1, t_2, \ldots, t_m \rightarrow \text{apply}(t_1, t_2, \ldots, t_m) )</td>
<td>( t_1, \ldots, t_m \rightarrow t_1 ) for all ( i )</td>
</tr>
</tbody>
</table>
**Definition 3.8.** A ground term \( t \) is valid iff for any subterm \( s \in \text{Sub}(t) \):

1. \( s = \text{sig}(p, q) \implies \neg q \vdash \text{priv}(q') \) for some \( q' \);
2. \( s = \text{priv}(p) \implies \neg p \vdash A \)

We suppose hereinafter that the considered constraint system \( S \) contains at least one atom, i.e. \( \text{QSub}(S) \cap \mathcal{A} \neq \emptyset \). Otherwise we add a dummy constraint \( \{a \} \vdash a \) to \( S \) which will be satisfied by any substitution. We define \( \text{priv}(T) = \{\text{priv}(t) : t \in T\} \) for a set of terms \( T \). We define \( \text{Vars}(S) = \bigcup_{i=1}^{\mathcal{m}} \text{Vars}(E_i) \cup \text{Vars}(t_i) \). We say that \( S \) is normalized, iff for all \( t \in \text{Sub}(S) \), \( t \) is normalized.

**Example 5.** We give a example of general constraint system and its solution within \( \text{DY} + \text{ACI} \) deduction system.

\[
S = \begin{cases}
\text{enc}(x, a) \cdot \text{pair}(c, a) \vdash b \\
\cdot (x, c) \vdash a
\end{cases},
\]

where \( a, b, c \in \mathcal{A} \) and \( x \in \mathcal{X} \). One of its solutions within \( \text{DY} + \text{ACI} \) is \( \sigma = \{x \mapsto \text{enc}(\text{pair}(a, b), c)\} \).

**Definition 3.9.** Let \( T = \{t_1, \ldots, t_k\} \) be a non-empty set of terms. Then we define \( \pi(T) \) as follows:

\( \pi(T) = \neg \cdot (t_1, \ldots, t_k) \).

We remark that \( \pi(\{t\}) = \neg t \) and \( \pi(T) = \neg \pi(T) \). Note also that \( \pi(t_1, \ldots, t_k) \) can be interpreted as a set of terms. The terms derivable from \( \pi(t_1, \ldots, t_k) \) are derivable from \( t_1, \ldots, t_k \) and vice-versa.

**Lemma 10.** Let \( T \) be a set of terms \( T = \{t_1, \ldots, t_k\} \). Then \( \pi(T) \in \text{Der}(\neg T) \) and \( \neg T \subseteq \text{Der}(\{\pi(T)\}) \).

From Lemma 3 we derive another property: \( \pi(T) = \pi(\neg T) \).

**Lemma 11.** \( \pi(T_1 \cup \cdots \cup T_m) = \pi(\{\pi(T_1), \ldots, \pi(T_m)\}) \)

**Proof.** From definition of \( \pi \) and Lemma 2, we have \( \text{elems}(\pi(T_i)) = \neg \text{elems}(T_i) \). Next \( \pi(\{\pi(T_1), \ldots, \pi(T_m)\}) = \neg \cdot (L) \) where

\[
L \approx \neg \text{elems}(\{\pi(T_1), \ldots, \pi(T_m)\}) = \neg \bigcup_{i=1, \ldots, m} \text{elems}(T_i) \approx \neg \bigcup_{i=1, \ldots, m} \text{elems}(T_i)
\]

On the other hand \( \pi(T_1 \cup \cdots \cup T_m) = \neg \cdot (L') \), where \( L' \approx \bigcup_{i=1, \ldots, m} \text{elems}(T_i) \). \( \square \)

3.3. Solving the derivability problem of \( \text{DY} + \text{ACI} \)

The decidability of the derivability problem for \( \text{DY} \) has been extensively studied (see e.g. Amadio and Lugiez, 2000 for the atomic keys case, and Rusinowitch and Turuani, 2003 for the complex keys case) and is known to be polynomial. Also, we note that the derivability problem when the set of symbols only contains the dot \( \cdot \) and constants is trivial, for in this case a constraint \( E \vdash t \) is satisfied if, and only if, the constants occurring in \( t \) also occur in \( E \) (proof left to the reader). Then the combination algorithm of Chevalier and Rusinowitch (2005) yields directly the decidability of derivability for the \( \text{DY} + \text{ACI} \) case. Moreover since the derivability in \( \text{DY} \) and ACI sub-theories can be decided in polynomial time the algorithm deciding derivability returns in time polynomial in the number of subterms (Cortier and Delaune, 2012).
3.4. Existence of conservative solutions for satisfiable systems

In this subsection we show that for any satisfiable constraint system there exist particular models called conservative solutions. Roughly speaking, such a model can be defined by mapping each variable to a set of subterms and atoms extracted from the constraint system. This will bound the search space for finding a model (see § 3.5).

First, in Proposition 1 we state that a constraint system and its normal form have the same models and we show the equivalence between the existence of a model and the existence of a normalized model. As a consequence we will need only to consider normalized constraints and models in the sequel.

**Proposition 1.** The substitution $\sigma$ is a model of the constraint system $S$ if and only if $\sigma$ is a model of $\forall S \forall$. Moreover, $\sigma$ is a model of $S$ if and only if $\forall \sigma \forall$ is a model of $S$.

**Proof.** By definition, $\sigma$ is a model of $S = \{ E_i \mid i = 1, \ldots, n \}$, iff $\forall i \in \{ 1, \ldots, n \}, E_i(\sigma) \in \text{Der}(\forall E_i(\sigma))$. But we know that $\forall t_i(\sigma) = \forall t_i(\sigma)$ and $\forall E_i(\sigma) = \forall E_i(\sigma)$. Thus, $\sigma$ is a model of $S$ if and only if $\sigma$ is a model of $\forall S \forall$. But again, since $\forall t_i(\sigma) = \forall t_i(\sigma)$ and $\forall E_i(\sigma) = \forall E_i(\sigma)$, we have $\sigma$ is a model of $S$ if and only if $\forall \sigma \forall$ is a model of $S$. $\square$

We introduce a transformation $\pi(H_{S,\sigma}(\cdot))$ on ground terms that replaces recursively every occurrence of any binary symbol $\bin$ by the ACI symbol $\cdot$ (and flattens nested ACI terms) when this occurrence roots a subterm that is not matched by any non-variable subterm of the constraint system with substitution $\sigma$.

Later, we will show that $\pi(H(\sigma))$ is also a model of $S$.

**Definition 3.10.** We denote a set of non-variable subterms of a constraint system $S$ as $\text{Sub}(S)$, i.e. $\text{Sub}(S) \setminus \mathcal{X} = \text{Sub}(S)$.

**Definition 3.11.** Let $S$ be a normalized constraint system which is satisfiable with the model $\sigma$. Let us fix some $\alpha \in (\mathcal{A} \cap \text{Sub}(S))$. We define a function $H_{S,\sigma}(\cdot) : \mathcal{T}_g \rightarrow 2^{\mathcal{T}_S}$ as follows:

\[ H_{S,\sigma}(t) = \begin{cases} \{ \alpha \}, & \text{if } t \in (\mathcal{A} \setminus \text{Sub}(S)); \\ \{ a \}, & \text{if } t = a \in (\mathcal{A} \cap \text{Sub}(S)); \\ \{ \text{priv}(\pi(H_{S,\sigma}(t_1))) \}, & \text{if } t = \text{priv}(t_1); \\ \{ \text{bin}(\pi(H_{S,\sigma}(t_1)), \pi(H_{S,\sigma}(t_2))) \}, & \text{if } t = \text{bin}(t_1, t_2) \\ & \text{and } \forall t \in \forall \text{Sub}(S) \forall \sigma; \\ H_{S,\sigma}(t_1) \cup H_{S,\sigma}(t_2), & \text{if } t = \text{bin}(t_1, t_2) \\ & \text{and } \forall t \notin \forall \text{Sub}(S) \forall \sigma; \\ \bigcup_{p \in L} H_{S,\sigma}(p), & \text{if } t = L. \end{cases} \]

Henceforward, we will omit parameters and write $H(\cdot)$ instead of $H_{S,\sigma}(\cdot)$ for shorter notation.

**Definition 3.12.** We define the superposition of $\pi(\cdot)$ and $H(\cdot)$ on a set of terms $T = \{ t_1, \ldots, t_k \}$ as follows: $\pi(H(T)) = \{ \pi(H(t)) \mid t \in T \}$.

**Definition 3.13.** Let $\theta = \{ x_1 \mapsto t_1, \ldots, x_k \mapsto t_k \}$ be a substitution. We define $\pi(H(\theta))$ to be the substitution $\{ x_1 \mapsto \pi(H(t_1)), \ldots, x_k \mapsto \pi(H(t_k)) \}$.

Note that since for every $x \in \text{dom}(\theta)$ we have $x \notin \text{Vars}(\theta)$, we also have $\text{dom}(\pi(H(\theta))) = \text{dom}(\theta)$. Moreover, by definition, $\pi(H(\theta))$ is normalized.
Example 6. Let us consider a (normalized) constraint system $S$ and its model $\sigma$ from Example 5 and show that $\pi(H(\sigma))$ is also a model of $S$. $\pi(H(\text{enc}(\text{pair}(a, b), c))) = \pi(H(\text{pair}(a, b))) = \pi([a] \cup [b] \cup [c]) = \pi(a) \cap \pi(b) \cap \pi(c)$ (we suppose that $a < b < c$). One can see that $\pi(H(\sigma)) = \{x \mapsto (a, b, c)\}$ is also a model of $S$ within $\text{DV} + \text{ACI}$.

We give several useful properties about the function $H (\cdot)$.

Lemma 12. For a ground term $t$, $H (t) = \bigcup_{p \in \text{elems}(t)} H (p)$.

Lemma 13. For a ground term $t$, $H (t) = H (\langle t \rangle)$.

Proof. By induction on $|\text{Sub}(t)|$:

- $|\text{Sub}(t)| = 1$ is possible in the only case: $t = a \in \mathcal{A}$ and as $a = \langle a \rangle$, the equality is trivial;
- Suppose that for any $t : |\text{Sub}(t)| < k \ (k > 1)$, $H (t) = H (\langle t \rangle)$ holds;
- Given a term $t : |\text{Sub}(t)| = k \ (k > 1)$. We need to prove that $H (t) = H (\langle t \rangle)$.
  - if $t = \text{priv}(t_1)$, then $H (t) = \{\text{priv}(\pi(H(t_1)))\} = \{\text{priv}(\pi(H(\langle t_1 \rangle)))\}$
    = $H (\text{priv}(\langle t_1 \rangle)) = H (\langle t \rangle)$;
  - if $t = \text{bin}(p, q)$ and $\langle t \rangle \not\in \text{Sub}(S) \sigma \triangledown$, then $H (\langle t \rangle) = (\text{bin}(\langle p \rangle, \langle q \rangle)) = \{\text{bin}(\pi(H(\langle p \rangle)), \pi(H(\langle q \rangle)))\} = \{\text{by Lemma 3}\}$.

As a consequence, $\pi(H(t)) = \pi(H(\langle t \rangle)) = \pi(H(t))$. We also note as another consequence that $\pi(H(\cdot))$ transforms valid terms to valid ones.

Now, we show that for the subterms of a normalized constraint system instantiated with its normalized model, the application of the transformation $\pi(H(\cdot))$ on such terms is equivalent modulo ACI to the result obtained if the transformation was applied only on the model.

Proposition 2. Let $S$ be a normalized constraint system and $\sigma$ its normalized model. For all $t \in \text{Sub}(S)$, $\langle t \rangle \pi(H(\sigma)) \triangledown = \pi(H(\langle t \rangle))$.

Proof. Note that $t$ is normalized. We will prove it by induction on $|\text{Sub}(t)|$.

- Let $|\text{Sub}(t)| = 1$. Then:
  - either $t \in \mathcal{A}$. In this case $t \in (\mathcal{A} \cap \text{Sub}(S))$, and as $t \mu = t$ for any substitution $\mu$, then $\pi(H(\langle t \rangle)) = \pi(H(t)) = \pi(t) = t$ and $\pi(H(\langle t \rangle)) = t$. Thus, $t \pi(H(\langle t \rangle)) = \pi(H(t))$.
  - or $t \in \mathcal{A}$. As $\sigma$ is a model and $t \in \text{Sub}(S)$, we have $H(t) \in \text{dom}(\sigma)$, and, by definition, $t \in \text{dom}(\pi(H(\sigma)))$. Then, by definition of $\pi(H(\sigma))$, $t \pi(H(\langle t \rangle)) = \pi(H(t))$.
- Assume that for some $k \geq 1$ if $|\text{Sub}(t)| \leq k$, then $\langle t \rangle \pi(H(\langle t \rangle)) \triangledown = \pi(H(\langle t \rangle))$.
- Show that for any $t$ such that $|\text{Sub}(t)| \geq k + 1$, where $t = \text{bin}(p, q)$ or $t = \text{priv}(q)$ or $t = \langle t_1, \ldots, t_m \rangle$, but where $|\text{Sub}(p)| \leq k$, $|\text{Sub}(q)| \leq k$ and $|\text{Sub}(t_i)| \leq k$, for all $i \in \{1, \ldots, m\}$, statement $\langle t \rangle \pi(H(\langle t \rangle)) \triangledown = \pi(H(\langle t \rangle))$ is still true. We have:
  - either $t = \text{bin}(p, q)$. As $t = \text{bin}(p, q) \in \text{Sub}(S) \Rightarrow p \in \text{Sub}(S)$ and $q \in \text{Sub}(S)$. As $|\text{Sub}(p)| < |\text{Sub}(t)|$ and from the induction assumption, we have $\langle p \rangle \pi(H(\langle p \sigma \rangle)) \triangledown = \pi(H(\langle p \sigma \rangle))$. The same holds for $q$.
  - Again, as $\text{bin}(p, q) \sigma \in \text{Sub}(S) \sigma$ (as $\text{bin}(p, q) \not\in \mathcal{A}$ and $t \in \text{Sub}(S)$ we have that
Thus, the proposition is proven. □

Now we show that the derivability of a term from a set of terms is preserved by the transformation \( \pi(\cdot) \). But before we recall two simple properties of the derivability relation:

**Lemma 14.** Let \( A, B, C, D \subseteq T_g \). Then if \( A \subseteq \text{Der}(B) \) and \( B \subseteq \text{Der}(C) \) then \( A \subseteq \text{Der}(C) \). Moreover, if \( A \subseteq \text{Der}(B) \) and \( C \subseteq \text{Der}(D) \) then \( A \cup C \subseteq \text{Der}(B \cup D) \).

**Proposition 3.** Let \( S \) be a normalized constraint system and \( \sigma \) its normalized model. For any \( \text{DY + ACI} \) rule \( l_1, \ldots, l_k \rightarrow r \), \( \pi(H(r)) \in \text{Der}(\{\pi(H(l_1)), \ldots, \pi(H(l_k))\}) \).

**Proof idea.** We proceed by considering all possible deduction rules. To give an idea, we show a proof for only one rule (see the full proof in Appendix B.4): \( \text{aenc}(t_1, t_2) \rightarrow \pi(H(\cdot)) \). We have to show that term \( \pi(H(\cdot)) \) is derivable from the set of terms \( \{\pi(H(\text{aenc}(t_1, t_2))), \pi(H(\text{priv}(t_2)))\} \). Consider two cases:

1. \( \exists u \in \text{Sub}(S) \) such that \( \text{aenc}(t_1, t_2) \rightarrow u \). Then \( \pi(H(\text{aenc}(t_1, t_2))) = \text{aenc}(\pi(H(t_1)), \pi(H(t_2))) \), and then \( \pi(H(\cdot)) = \pi(H(t_1)) \in \text{Der}(\{\text{aenc}(\pi(H(t_1)), \pi(H(t_2))), \pi(H(\cdot))\}) \). On the other hand, we have \( \pi(H(\text{priv}(t_2))) = \pi(H(\text{priv}(t_2))) = \pi(H(\text{priv}(t_2))) \rightarrow \pi(H(\cdot)) \).

2. \( \exists u \in Q\text{Sub}(S) \) such that \( \text{aenc}(t_1, t_2) \rightarrow u \sigma \). Then \( \pi(H(\text{aenc}(t_1, t_2))) = \pi(H(\cdot)) \cup \pi(H(t_2)) \). Using **Lemma 10**, we have \( \pi(H(\cdot)) \cup \pi(H(t_2)) \subseteq \text{Der}(\{\pi(H(\text{aenc}(t_1, t_2)))\}) \). And then, by **Lemma 10** we have that \( \pi(H(\cdot)) \subseteq \text{Der}(\{\pi(H(\text{aenc}(t_1, t_2)))\}) \). Therefore, by **Lemma 14** and **Lemma 13**, \( \pi(H(\cdot)) = \pi(H(t_1)) \in \text{Der}(\{\pi(H(\text{aenc}(t_1, t_2)))\}) \). □

Using **Propositions 2 and 3** we will show that transformation \( \pi(H(\cdot)) \) preserves the property of a substitution to be a model.

**Proposition 4.** Given a normalized constraint system \( S \) and its normalized model \( \sigma \). Then substitution \( \pi(H(\cdot)) \) also satisfies \( S \).

**Proof.** Suppose, without loss of generality, \( S = \{E_i \uparrow t_1\}_{i=1, \ldots, n} \). Let us take any constraint \( (E \uparrow t) \in S \). As \( \sigma \) is a model of \( S \), there exists a derivation \( D = A_0, \ldots, A_k \) such that \( A_0 = E \sigma \) and \( \pi(t_2) \in A_k \).

By **Proposition 3** and **Lemma 14** we can easily prove that if \( k > 0 \), then \( \pi(H(A_j)) \subseteq \text{Der}(\pi(H(A_j \uparrow t))) \), \( j = 1, \ldots, k \). Then, applying transitivity of \( \text{Der}(\cdot) \) (Lemma 14) \( k \) times, we have that \( \pi(H(A_k)) \subseteq \text{Der}(\pi(H(A_0))) \). In the case where \( k = 0 \), the statement \( \pi(H(A_k)) \subseteq \text{Der}(\pi(H(A_0))) \) is also true.

Using **Proposition 2** we have that \( \pi(H(A_0)) = \pi(H(E \sigma)) = \pi(E \pi(H(\sigma))) \), as \( E \subseteq \text{Sub}(S) \). The same for \( \pi(H(\cdot)) = \pi(t \pi(H(\cdot))) \), and as \( \pi(t_2) \in A_k \), we have \( \pi(t_2) \pi(H(\cdot)) \in \pi(H(A_k)) \). Therefore,
we have that \( \tau_p (H(\sigma))^\tau \in \pi(H(A_k)) \subseteq \text{Der}(\pi(H(A_0))) = \text{Der}(\tau_E \pi(H(\sigma))^\tau) \), that means \( \pi(H(\sigma)) \) satisfies any constraint of \( S \).  \( \Box \)

Till the end of subsection we will study a useful property of \( \pi(H(\sigma)) \). Proposition 5 and its corollary show that if a constraint system has a normalized model \( \sigma \) which maps different variables to different values, then there exists another normalized model \( \pi(H(\sigma)) \) that maps any variable of its domain to an ACI-set of some non-variable subterms of the constraint system instantiated by itself and some private keys built with atoms of the constraint system. This kind of model will be called conservative.

**Lemma 15.** Given a normalized substitution \( \sigma \) and normalized term \( u \). If \( \tau u \sigma ^\tau = \text{bin}(p, q) \) and \( u \not\in \mathcal{X} \) and \( x \sigma \neq y \sigma \) for all \( x \neq y \) then there exists \( s \in \text{Sub}(u) \) such that \( s = \text{bin}(p', q') \) and \( \tau s \sigma ^\tau = \text{bin}(p, q) \). The same is true in the case of \( \tau u \sigma ^\tau = \text{priv}(p) \).

**Proof.** As \( u = \tau u \sigma ^\tau = \text{bin}(p, q) \), we have:

- \( u \) not in form of \( \cdot \). Then, as \( u \not\in \mathcal{X} \) and \( \tau u \sigma ^\tau = \text{bin}(p, q) \), we have \( u = \text{bin}(p', q') \) (where \( \tau p' \sigma ^\tau = p \) and \( \tau q' \sigma ^\tau = q \)). Then we can choose \( s = \text{bin}(p', q') \).
- \( u = (t_1, \ldots, t_m) \). Every root \( t_i \neq \cdot \) and \( m > 1 \) since \( u = \tau u \sigma ^\tau \). Then for all \( i \), \( t_i \) is either a variable, or \( \text{bin}(p'_i, q'_i) \). But as \( x \sigma \neq y \sigma \) for all \( x \neq y \) and as \( \sigma \) is normalized, we can claim that the set \( \{t_1, \ldots, t_m\} \) contains at most one variable \( (\exists j : t_j \in \mathcal{X} \implies \forall k \neq j, k \not\in \mathcal{X}) \). Since \( \tau u \sigma ^\tau = \text{bin}(p, q) \), we have \( \forall i, j \tau t_i \sigma ^\tau = \tau t_j \sigma ^\tau \), and then, as \( m > 1 \), there exists \( i \) such that \( t_i = \text{bin}(p'_i, q'_i) \). Then, by definition of normalization function and from \( \tau u \sigma ^\tau = \text{bin}(p, q) \) we have that \( \tau \text{elems}(u \sigma ^\tau) = \{\text{bin}(p, q)\} \). Thus, \( \tau \text{bin}(p'_i, q'_i) \sigma ^\tau = \text{bin}(p, q) \), i.e. we can choose \( s = t_i \), as \( t_i \in \text{Sub}(u) \) and \( t_i = \text{bin}(p'_i, q'_i) \).

The other case (\( \text{priv} \)) can be proved similarly.  \( \Box \)

**Definition 3.14.** A substitution \( \sigma \) is a conservative model of a constraint system \( S \), iff

1. \( \sigma \) is normalized;
2. \( \sigma \) is a model of \( S \);
3. For any \( x \in \text{dom}(\sigma) \) there exist \( k \in \mathbb{N} \) and \( s_1, \ldots, s_k \in \text{Sub}(S) \cup \text{priv}(\text{Sub}(S) \cap \mathcal{A}) \) such that \( x \sigma = \pi((s_1 \sigma \ldots, s_k \sigma)) \) and \( s_i \neq s_j, \) if \( i \neq j \) and root \( s_i \neq \cdot \) for all \( i \).

**Proposition 5.** Assume we are given a normalized constraint system \( S \) and its normalized model \( \sigma \) such that \( \forall x, y \in \text{dom}(\sigma), x \neq y \implies x \sigma \neq y \sigma \). Then \( \pi(H(\sigma)) \) is a conservative model of \( S \).

**Proof.** From Proposition 4 we obtain that \( \pi(H(\sigma)) \) is a normalized solution of \( S \). By definition, \( x \pi(H(\sigma)) = \pi(H(x \sigma)) \). Let us take any \( s \in H(x \sigma) \) (note that \( s \) is a ground term). Then, by definition of \( H(\cdot) \) we have:

- either \( s \in \mathcal{A} \). Then, by definition of \( H(\cdot), s \in (\mathcal{A} \cap \text{Sub}(S)) \). Thus, \( s \pi(H(\sigma)) = s, s \in \text{Sub}(S), s \neq \cdot \);
- or \( s = \text{bin}(\pi(H(t_1)), \pi(H(t_2))) \) and \( \exists u \in \text{Sub}(S) \) such that \( \tau u \sigma ^\tau = \text{bin}(t_1, t_2) \succ = \text{bin}(t_1, t_2) \). As all conditions of Lemma 15 are satisfied, \( \exists v \in \text{Sub}(u) \) such that \( \tau v \sigma ^\tau = \text{bin}(t_1, t_2) \). Then, as \( u \) and \( v \) are both in \( S \) and \( \cdot \in \text{Sub}(S) \), then \( v \in \text{Sub}(S) \). By Proposition 2, \( \tau v \pi(H(\sigma)) = \pi(H(\cdot)) \succ = \pi(H(\cdot)) \succ = \pi(H(t_1, t_2)) = \pi(\{\text{bin}(\pi(H(t_1)), \pi(H(t_2)))\}) = \text{bin}(\pi(H(t_1)), \pi(H(t_2))) = s \). That means, \( \exists v \in \text{Sub}(S), v \neq \cdot \) such that \( s = \tau v \pi(H(\sigma)) \).
- or \( s = \text{priv}(\pi(H(t_1))) \). In this case, as \( s \) is ground, \( \pi(H(t_1)) \) must be an atom (we recall that \( \pi(H(\cdot)) \) transforms valid terms to valid ones), moreover, by definition of \( H(\cdot) \), this atom is from \( (\mathcal{A} \cap \text{Sub}(S)) \). Therefore, \( s = \text{priv}(a) \), where \( a \in \mathcal{A} \cap \text{Sub}(S) \) (and of course, \( s \neq \cdot \)).
Thus, for all $s \in H(\pi), there exists $v \in \text{Sub}(S) \cup \text{priv}(\text{Sub}(S) \cap A) \setminus X$ such that $s = \forall v \pi (H(\pi))^\vartriangleleft$. Therefore, as $x \pi (H(\pi)) = \pi (H(x \pi))$, we have that $x \pi (H(\pi)) = \pi (\{s_1 \pi (H(\pi)), \ldots, s_k \pi (H(\pi))\}) = \pi (\{s_1, \ldots, s_k \pi (H(\pi))\}), where s_1, \ldots, s_k \in \text{Sub}(S) \cup \text{priv}(\text{Sub}(S) \cap A)$ and $s_i \neq \cdot (L), \forall 1 \leq i \leq k$. That proves the proposition. □

**Corollary 6.** Given a normalized constraint system $S$ and its normalized model $\sigma'$ such that $x \neq y \implies x\sigma' \neq y\sigma'$. Then $S$ admits a conservative solution.

### 3.5. Bounds on conservative solutions

To get a decidability result, we first show an upper bound on the size of a conservative model. Then we reduce any satisfiable constraint system to another satisfiable one that admits a conservative model (this reduction is, in fact, using one name for the variables on which some preliminary fixed model returns equal values). Moreover the considered conservative model of the obtained constraint system can be easily extended to a conservative model (of the same size) of the initial constraint system. We also show that the reduced constraint system has not a larger size than the original one. This means that the original constraint system has a model which is bounded with regard to the size of the constraint system. Thus, we prove the existence of a model with bounded size for any satisfiable constraint system.

**Lemma 16.** For any term $t$ and substitution $\sigma$, $\forall s \in \text{Sub}(t) [\text{Sub}(t^\sigma s) \geq |\text{Sub}(t^\sigma s)|].$

**Proof.** From Lemma 5 we can easily obtain $\text{Sub}(s^\sigma) \subseteq \text{Sub}(t^\sigma)$. Then we need to prove that $|\text{Sub}(t^\sigma)| \leq |\text{Sub}(t^\sigma s)|$, if $\text{Sub}(p) \subseteq \text{Sub}(q)$. The proof is mainly based on the fact that if $\text{Sub}(p) \subseteq \text{Sub}(q)$ then $\text{Sub}(p^\sigma) \setminus \text{Sub}(q^\sigma) \subseteq \text{Sub}(q^\sigma)$. Let us consider several cases.

- If $p = q$ then the statement is trivial.
- If there exists $v \in \text{Sub}(q)$ such that $v = \text{bin}(p, p')$ or $v = \text{bin}(p', p)$ or $v = \text{priv}(p)$. Then note that by definition of the normalization, since root $(v) \neq -, we have $\forall v \in \text{Sub}(q^\sigma)$. Then (without loss of generality we consider only case $v = \text{bin}(p, p')$ $\forall v \in \text{Sub}(q^\sigma)$ and thus $\forall v \in \text{Sub}(q^\sigma)$). Therefore, (since $\forall v \in \text{Sub}(q^\sigma)$ $\forall v \in \text{Sub}(q^\sigma)$ and then $\forall v \in \text{Sub}(q^\sigma)$). From this follows $\forall v \in \text{Sub}(q^\sigma)$.
- Otherwise, there exists $v \in \text{QSub}(q)$ such that $v = \cdot (L)$ and $\text{elems}(p) \subseteq \text{elems}(v)$ (note that $v \neq p$, otherwise we are in the one of the two cases above). From Lemma 2 we have $\text{elems}(p^\sigma) \subseteq \text{elems}(v^\sigma)$. Note that by definition of the normalization and since $v \in \text{QSub}(q)$ (and thus $v = q$ or there exists $v' \in \text{Sub}(q)$ such that $v' = \text{bin}(v, p')$ or $v' = \text{bin}(p', v)$ or $v' = \text{priv}(v)$) we have that $\forall v \in \text{Sub}(q^\sigma)$, moreover, $\forall v \in \text{Sub}(q^\sigma)$. Then from Lemma 6 we have $\forall v \in \text{Sub}(q^\sigma)$, consequently, $\forall (\text{elems}(p^\sigma)) \subseteq \text{Sub}(q^\sigma), q^\sigma)$. Now, if $\forall (\text{elems}(p^\sigma)) \subseteq \text{Sub}(q^\sigma), q^\sigma)$. Since we have already shown that $\forall (\text{elems}(p^\sigma)) \subseteq \text{Sub}(q^\sigma), q^\sigma)$, to prove the statement it is enough to show that there exists $p' \in \text{Sub}(q^\sigma)$ such that $p' \neq \text{Sub}(\text{elems}(p^\sigma))$. For such value we can choose $p' = \forall v \quad (\forall v \neq \text{Sub}(\text{elems}(p^\sigma))$ cannot be in $\text{elems}(p^\sigma), \text{elems}(p^\sigma) \subseteq \text{elems}(v^\sigma) \subseteq \text{Sub}(v^\sigma)$ and in the current case $\text{root}(v^\sigma) = \cdot$.) □

**Lemma 17.** Given a normalized constraint system $S$ and its conservative model $\sigma$. Then $\forall x \in \text{Vars}(S), \text{Sub}(x^\sigma) \subseteq \text{Sub}(S) \sigma \cap \text{priv}(\text{Sub}(S) \cap A)$.

**Proof.** A ground substitution $\sigma$, let us define a strict total order on variables: $x \sqsubset y \iff (|\text{Sub}(x^\sigma)| < |\text{Sub}(y^\sigma)|) \lor (|\text{Sub}(x^\sigma)| = |\text{Sub}(y^\sigma)|) \land x < y$.

By Proposition 5 for any $x \in \text{Vars}(S)$ we have $x^\sigma = \pi (\{s_1^x \sigma, \ldots, s_k^x \sigma\}), where s_i^x \sigma \in (\text{Sub}(S) \setminus X) \cup \text{priv}(\text{Sub}(S) \cap A)$ and $s_i^x \sigma \neq \cdot (L)$. Moreover, $s_i^x$ are normalized (as $S$ is normalized).
Let us show that if \( y \in \text{Vars}(s^x_i) \) for some \( i \), then \( y \in \text{Vars}(s^x_i) \) and \( x \not\in y \). Suppose that \( y \in \text{Vars}(s^x_i) \) and \( x \not\in y \). Then \( |\text{Sub}(x\sigma)| = |\text{Sub}(\{(s^x_1\sigma, \ldots, s^x_k\sigma)\})| = |\text{Sub}(\langle s^x_1\sigma, \ldots, s^x_k\sigma \rangle)\rangle | \geq (\text{by Lemma 16}) \geq |\text{Sub}(\langle s^x_1\sigma \rangle)\rangle \geq |\text{Sub}(\langle y\sigma\rangle)\rangle \), because we know that \( s^x_i = \text{bin}(p, q) \) or \( s^x_i = \text{priv}(p) \) and \( y \in \text{Vars}(s^x_i) \). (for example, in the first case, \( |\text{Sub}(\langle s^x_1\sigma \rangle)\rangle = |\text{Sub}(\langle p\sigma\neg, q\sigma\neg \rangle)\rangle| = 1 + |\text{Sub}(\langle p\sigma\neg, q\sigma\neg \rangle)\rangle| \\) and as \( y \in \text{Vars}(\langle p, q \rangle) \), using Lemma 16, we get \( |\text{Sub}(\langle s^x_1\sigma \rangle)\rangle \geq 1 + |\text{Sub}(\langle y\sigma\rangle)\rangle \). Since \( |\text{Sub}(\langle y\sigma\rangle)\rangle = |\text{Sub}(\langle y\sigma \rangle)\rangle \) we have \( y \not\in x \). Contradiction.

Now we show by induction the main property of this lemma.

- let \( x = \min_\prec(\text{Vars}(S)) \). Then \( x\sigma = \pi(\{(s^x_1\sigma, \ldots, s^x_k\sigma)\}) = \text{r}(s^x_1\sigma, \ldots, s^x_k\sigma) \) and all \( s^x_i \) are ground (as \( \not\in y \subseteq x \)). Then \( x_\sigma = \text{r}(s^x_1\sigma, \ldots, s^x_k\sigma) \). Since \( s^x_i \) are normalized with non- root, we have that \( \text{Sub}(x\sigma) = \text{r}(s^x_1\sigma, \ldots, s^x_k\sigma) \cup \text{Sub}(s^x_1\sigma) \cup \ldots \cup \text{Sub}(s^x_k\sigma) \subseteq \text{r} \text{Sub}(S) \sigma \cup \text{priv}(\text{A} \cap \text{Sub}(S)) \), as \( y \in \text{Sub}(s^x_i) \), \( s \in \text{Sub}(S) \) or \( s = \text{ priv}(a) \) or \( s = a \), where \( a \in \text{Sub}(S) \cap A \), therefore \( s = s_\sigma = s_\sigma = s_\sigma = s_\sigma \in \text{r} \text{Sub}(S) \sigma \cap \text{priv} \text{Sub}(S) \cap A \) and \( r \text{Sub}(s^x_i) \sigma \cap \text{priv} \text{Sub}(S) \cap A \).
- Suppose, for all \( z \subseteq y \), \( \text{Sub}(z\sigma) \subseteq \text{r} \text{Sub}(S) \sigma \cap \text{priv} \text{Sub}(S) \cap A \).
- Show that \( \forall y \subseteq \text{Sub}(S) \cap A \).

We know that \( y\sigma = \pi(\{(s^x_1\sigma, \ldots, s^x_k\sigma)\}) = \text{r}(s^x_1\sigma, \ldots, s^x_k\sigma) \) and \( \forall z \in \text{Vars}(s^x_i) \), \( z \subseteq y \). Then we have \( \text{Sub}(y\sigma) = \text{r} \text{Sub}(s^x_i) \sigma \cup \text{Sub}(s^x_i) \sigma \cup \ldots \cup \text{Sub}(s^x_k) \sigma \). We know that \( y\sigma \in \text{r} \text{Sub}(S) \sigma \sigma \).

Let us show that \( \text{Sub}(s^x_i) \sigma \cap \text{priv} \text{Sub}(S) \cap A \). By Theorems 5 and 9 we have \( \text{Sub}(s^x_i) \sigma \cap \text{priv} \text{Sub}(S) \cap A \); then from the induction hypothesis and the statement proved above we have \( \text{Sub}(s^x_i) \sigma \subseteq \text{r} \text{Sub}(S) \sigma \cap \text{priv} \text{Sub}(S) \cap A \). Thus, \( \text{Sub}(y\sigma) \subseteq \text{r} \text{Sub}(S) \sigma \cap \text{priv} \text{Sub}(S) \cap A \).

**Proposition 7.** For a normalized constraint system \( S \) that has a conservative model \( \sigma \), \( \forall x \in \text{Vars}(S) \), \( |\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(S)| \).

**Proof.** As \( |\text{Sub}(S)\sigma\sigma| \leq |\text{Sub}(S)\sigma| \leq |\text{Sub}(S)| \), we have that \( |\text{Sub}(x\sigma)| \leq |\text{Sub}(S)\sigma\sigma \cup \text{priv}(\text{A} \cap \text{Sub}(S))| \leq |\text{Sub}(S)\sigma\sigma| \cup |\text{priv}(\text{A} \cap \text{Sub}(S))| \leq (\text{as } |\text{T}\sigma| \leq |T| \text{ and } |T\sigma| \leq |T|) \leq |\text{Sub}(S)| \cup |\text{A} \cap \text{Sub}(S)| \leq 2 \times |\text{Sub}(S)| \); thus, \( |\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(S)| \).

From this proposition and Corollary 6 we obtain the existence of a bounded model for any normalized constraint system that has a model mapping different variables to different values. We will reduce an arbitrary constraint system to the already studied case. The target properties are stated in Proposition 8 and Corollary 9.

**Lemma 18.** Given any constraint system \( S \) and any substitution \( \theta \) such that \( \text{dom}(\theta) = \text{Vars}(S) \) and \( \text{img}(\theta) \subseteq \text{dom}(\theta) \), then \( |\text{Sub}(\theta\sigma)| \leq |\text{Sub}(S)| \).

**Proof.** By Lemma 5, \( |\text{Sub}(S\theta)| \leq |\text{Sub}(S)\theta \cup \text{Vars}(S\theta)| \), but \( \text{Vars}(S) \theta \subseteq \text{dom}(\theta) = \text{Vars}(S) \); \( \text{Vars}(S) \theta \) consists only of variables, and then \( \text{Sub}(\text{Vars}(S) \theta) \theta = \text{Vars}(S) \theta \). As \( \text{Vars}(S) \subseteq \text{Sub}(S) \), we have \( \text{Sub}(S) \theta \cup \text{Sub}(\text{Vars}(S) \theta) \theta = \text{Sub}(S) \theta \). Thus, \( |\text{Sub}(S\theta)| \leq |\text{Sub}(S)\theta| \leq |\text{Sub}(S)| \). □

**Definition 3.15.** Let \( \sigma \) and \( \delta \) be two substitutions. Then \( \sigma[\delta] \) is a substitution such that \( \text{dom}(\sigma[\delta]) = \text{dom}(\delta) \) and for all \( x \in \text{dom}(\sigma[\delta]) \), \( x\sigma[\delta] = (x\delta)\sigma \).

**Lemma 19.** Let \( \theta \) and \( \sigma \) be two substitutions such that \( \text{img}(\theta) = \text{dom}(\sigma) \), \( \text{dom}(\sigma) \subseteq \text{dom}(\theta) \). Then, for any term \( t \), \( (t\theta)\sigma = t\sigma[\theta] \).

**Proof.** When we apply \( \theta \) to \( t \), every variable \( x \) of \( t \) such that \( x \in \text{dom}(\theta) \) is replaced by \( x\theta \); then we apply \( \sigma \) to \( t\theta \); every variable \( y \) of \( t\theta \) is replaced by \( y\sigma \), thus, every variable \( x \) from \( \text{dom}(\theta) \) will...
be replaced by \((x\theta)\sigma\) (as \(\text{img}(\theta) = \text{dom}(\sigma)\)); and no other variables will be replaced (as \(\text{dom}(\sigma) \subseteq \text{dom}(\theta)\)). This corresponds with the definition of \(\sigma[\theta]\). □

**Proposition 8.** Given a satisfiable constraint system \(S\) there exists a model \(\sigma\) of \(S\) such that \(\forall x \in \text{dom}(\sigma), |\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(\tau S')|\)

**Proof.** Given a normalized model \(\sigma'\) of \(S\) we build a substitution \(\theta\) that maps different variables whose \(\sigma'\)-values are equal to a single new variable. In this way we obtain a new constraint system \(\tau S'\) and its normalized model. Then we apply Corollary 6 and get a conservative model \(\sigma''\) of \(\tau S'\). By applying Proposition 7 we get a bound on the size for this model. On the other hand, we use Lemma 19 to show that \(\sigma''[\theta]\) is a model of \(\tau S'\). Then, using the obtained bound and Lemma 18, we show the existence of a model with the desired property. The detailed proof is given in Appendix B.6 □

**Corollary 9.** A constraint system \(S\) is satisfiable iff there exists a normalized model of \(S\) defined on \(\text{Vars}(S)\) which maps a variable to a ground term in \(T(A \cap Q\text{Sub}(\tau S'), \emptyset)\) of size \(\leq 2 \times |\text{Sub}(S)|\).

Using this result we get Algorithm 1 for solving constraint systems:

**Algorithm 1:** Solving constraint systems.

| Input: A constraint system \(S = \{E_i \triangleright t_i\}_{i=1,...,n}\) |
| Output: A model \(\sigma\), if it exists else \(\perp\) |

1. **Guess** for every variable \(x\) of \(S\) a value for \(x\sigma\) of size \(\leq 2 \times |\text{Sub}(S)|\);
2. **if** \(\sigma\) satisfies \(E_i \triangleright t_i\) for all \(i = 1,\ldots,n\) **then** return \(\sigma\) **else** return \(\perp\)

**Proposition 10.** Algorithm 1 is correct.

**Proof.** Let \(\sigma\) be an output of Algorithm 1. Then \(\sigma\) is a ground substitution and \(\sigma\) satisfies all constraints from \(S'\) and therefore, satisfies all constraints from \(S\). This means, \(\sigma\) is a model of \(S\). □

**Proposition 11.** Algorithm 1 is complete.

**Proof.** Suppose \(S\) is satisfiable. Then, by Corollary 9, there exists a guess of values for every element of \(\text{Vars}(S)\) with size \(\leq 2 \times |\text{Sub}(S)|\) which defines a model \(\sigma\) of \(S\). Thus Algorithm 1 will return this \(\sigma\). □

### 4. Complexity analysis

In this section we study the complexity of the proposed algorithm. First, we expose a representation of constraint systems to justify the selected measure for algorithm inputs. Then, we remark that the normalization algorithm is polynomial and we show that the derivability algorithm is polynomial too. As a consequence the algorithm for solving general constraint systems within the DY + ACI model is in \(\text{NP}\). Since constraint solving is known to be \(\text{NP}\)-hard for well-formed constraints, we deduce it is \(\text{NP}\)-complete for general constraints.

To determine the algorithm complexity we first define a reasonable measure for inputs. Remark that \(|\text{Sub}(\cdot)|\) does not approximate polynomially the number of bits needed to write its argument (a term, a set of terms, or a constraint system). As a measure for terms and sets of terms we will use \(|\text{Sub}(\cdot)| + |E(\cdot)|\), where \(E(\cdot)\) is a set of edges in the DAG-representation of its argument. For the size of a system of constraints \(S = \{E_i \triangleright t_i\}_{i=1,...,n}\) we will use \(n \times |\text{Sub}(S)| + |E(S)|\). The justification is given below.
Definition 4.1. The DAG-representation of a constraint system \( S = \{ E_i \mid t_i \} \) is a tagged graph with labeled edges \( G = (V, E, \text{tag}) \) (\( V \) is a set of vertices and \( E \) is a set of edges; \( \text{tag} \) is a tagging function defined on \( V \)) such that:

- there exists a bijection \( f : V \mapsto \text{Sub}(S) \);
- \( \forall v \in V \) \( \text{tag}(v) = (s, m) \), where
  \( s = \text{root}(f(v)) \);
  \( m \) is a \( 2n \)-bit integer, where \( m[2i-1] = 1 \) if and only if \( f(v) \in E_i \) and \( m[2i] = 1 \) if and only if \( f(v) = t_i \);
- \( v_1 \mapsto v_2 \in E \) if and only if \( \exists p \in T : (\exists \text{bin} : f(v_1) = \text{bin}(f(v_2), p)) \lor f(v_1) = \text{priv}(f(v_2)) \);
- \( v_1 \mapsto v_2 \in E \) if and only if \( \exists p \in T : \exists \text{bin} : f(v_1) = \text{bin}(p, f(v_2)) \);
- \( v_1 \mapsto v_2 \in E \) if and only if \( f(v_1) = \cdot (L) \land L[i] = f(v_2) \);

Example 7. A constraint system
\[
S = \begin{cases} 
\{ \text{enc}(a, x), \text{pair}(b, \text{enc}(a, a)), c \} & \triangleright a \\
\{ \text{priv}(b), c \} & \triangleright y \\
\{ \text{enc}(\text{sig}(a, \text{priv}(c)), y), \text{aenc}(x, b) \} & \triangleright \text{pair}(\text{enc}(a, x), c) 
\end{cases}
\]
will be represented as shown\(^2\) in Fig. 4. The nodes of this graph represent the elements from \( \text{Sub}(S) \) by indicating their root symbols (first part of their tags) and pointers to their children.

The given representation takes less than \( P(n \times |V(S)| + |E(S)|) \) bits, where \( n \) is the number of constraints, \( V(\cdot) \) is the set of nodes and \( E(\cdot) \) is the set of edges in the DAG-representation, and \( P \) is some polynomial with non-negative coefficients. As we have a bijection between \( V(S) \) and \( \text{Sub}(S) \), we obtain \( |V(S)| = |\text{Sub}(S)| \). Therefore we will estimate the complexity of our algorithm by taking \( n \times |\text{Sub}(S)| + |E(S)| \) as the measure of the constraint system \( S \).

The DAG-representation of a term \( t \) has a structure similar to the one of a constraint system except that the second part of the tagging function is dropped: we need only root \( f(v) \) as a node’s tag. The size of this representation will be polynomially bounded by \( |\text{Sub}(t)| + |E(t)| \). Thus we give the following definition:

\(^2\) Label “1” (resp. “2”) of an edge is represented by a left (resp. right) side of its source node.
Definition 4.2. The measure of a term 𝑡 is: \( \text{measure}(𝑡) = |\text{Sub}(𝑡)| + |\text{E}(𝑡)| \). The measure of a constraint system \( S = \{ E_i \rightarrow t_i \}_{i=1, \ldots, n} \) is: \( \text{measure}(S) = \pi \times |\text{Sub}(S)| + |\text{E}(S)| \).

We remark that the given measure is not the exact number of bits needed to write the argument down (this value involves a logarithm factor and would complexify the notation), but is a polynomial approximation of it which is sufficient for our purpose.

Note that for normalized terms and constraint systems the number of edges in their DAG-representation are polynomially bounded w.r.t. the number of vertices:

Lemma 20. For any normalized term \( t \), \( |\text{E}(t)| < (|\text{Sub}(t)|)^2 \). For any normalized constraint system \( S \), \( |\text{E}(S)| < (|\text{Sub}(S)|)^2 \).

Proof. Since the term (resp. constraint system) is normalized, there are at most two edges between two nodes. A binary node has at most two outgoing edges per destination node, an unary node – one, and \( \neg \)-node has at most one outgoing edge per destination node because of normalization. Therefore, as the graph is directed and acyclic, with at most two edges between two nodes, we have less than \(|\text{Sub}(x)| \times (|\text{Sub}(x)| - 1) \) edges (where \( x \) is a term \( t \) or a constraint system \( S \)). \( \square \)

4.1. Satisfiability of a general \( DY + ACI \) constraint system is in \( NP \)

Lemma 21. The normalization of a term \( t \) can be done in polynomial time on measure \( (t) \). The same holds for a constraint system \( S \): normalization can be done in polynomial time on measure \( (S) \).

Proof idea (for the case of terms). The algorithm of term normalization works bottom-up by flattening nested ACI-sets, sorting children of ACI-set nodes, merging duplicated nodes while removing unnecessary duplicating edges and removing nodes without incoming edges (except the root-node of \( t \)). \( \square \)

Lemma 22. For a term \( t \), \( |\text{Sub}(\neg t)\neg| \leq |\text{Sub}(t)| \); for a set of terms \( T \), \( |\text{Sub}(\neg T)\neg| \leq |\text{Sub}(T)| \); for a constraint system \( S \), \( |\text{Sub}(\neg S)\neg| \leq |\text{Sub}(S)| \).

The proof for the case of terms is given in Appendix B.5 \( \square \)

Proposition 12. Satisfiability of general \( DY + ACI \) constraint systems is in \( NP \).

Proof. Algorithm 1 returns a solution if it exists. We have to show that the verification of this solution is polynomial with regard to the input. To do this, we will normalize \( S\sigma \) and then apply the result described in § 3.3 for checking derivability. Using the fact that \( |\text{Sub}(\neg x\sigma)\neg| \leq 2 \times |\text{Sub}(S)| \) and normalization is polynomial, checking the validity of terms in \( \neg S\sigma \) and the derivability, we can get an upper bound of the execution time that is polynomial on measure \( (S) \). The details of the proof are given in Appendix B.3. \( \square \)

On the other hand, we can reuse a technique presented in Rusinowitch and Turuani (2003) to show that the satisfiability of a constraint system is an \( NP \)-hard problem. The authors encode the 3-SAT problem into an insecurity problem of a single-session sequential protocol. Because the steps of the protocol are linearly ordered, finding an attack is reduced to the satisfiability problem of a single well-formed constraint system. Since the class of constraint systems we consider contains the case presented in this work, we can conclude to the \( NP \)-hardness of our problem. Thus,

Theorem 1. Satisfiability of general \( DY + ACI \) constraint systems is \( NP \)-complete.
5. Satisfiability of general DY constraint systems

The constraint solving algorithm for the DY + ACI theory can be adapted to the classical DY case. But we cannot apply it directly, as a derived solution may contain an ACI symbol. Thus, we need to do some more work to prove the decidability of the DY case. The scheme we follow to solve a constraint system within the DY deduction system is shown in Fig. 5.

First, we show that if a constraint system is satisfiable within DY, then it is satisfiable within DY + ACI (Proposition 13). Second, as we know, we can find a model of a given constraint system within DY + ACI. Third, we transform the model obtained from previous step (which is in DY + ACI) in such a way, that the resulting substitution will be a model of the initial constraint system within DY (Theorem 2). The transformation δ to be used is simple: we replace any ACI list of terms \( \cdot (t_1, \ldots, t_n) \) by nested pairs: \( \text{pair}(t_1, \text{pair}(\ldots, t_n)) \). Note that this transformation has a linear complexity and the transformed model will have a DAG-size not more than twice bigger than the initial one. This keeps the complexity in NP, for the problem of satisfiability of general constraint system within the DY model.

**Definition 5.1.** We define a replacement \( \delta(t) : \mathcal{T}_g \mapsto \mathcal{T}_g \) in the following way:

\[
\delta(t) = \begin{cases} 
  t, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\
  \text{bin}(\delta(p), \delta(q)), & \text{if } t = \text{bin}(p, q); \\
  \text{priv}(\delta(p)), & \text{if } t = \text{priv}(p); \\
  \delta(t_1), & \text{if } t = \cdot(t_1); \\
  \text{pair}(\delta(t_1), \delta(\cdot(t_2, \ldots, t_m))), & \text{if } t = \cdot(t_1, \ldots, t_m), m > 1; 
\end{cases}
\]

**Definition 5.2.** Given a substitution \( \sigma \). Then \( \delta(\sigma) = \{x \mapsto \delta(x\sigma)\}_{x \in \text{dom}(\sigma)} \). For \( T \subseteq \mathcal{T}_g \), \( \delta(T) = \{\delta(t) : t \in T\} \).

To recall the Dolev–Yao deduction system (DY), see Table 1.

**Definition 5.3.** A constraint system \( \mathcal{S} \) is standard if for all \( s \in \text{Sub}(\mathcal{S}) \) \( \text{root}(s) \neq \cdot \). The definition is extended in natural way to terms, sets of terms and substitutions.

We can redefine the notion of derivation for Dolev–Yao deduction system in a natural way, and denote it as Der\(_{DY}\).

**Lemma 23.** Any standard constraint system is normalized.

**Lemma 24.** Let \( t \) be a standard term and \( \sigma \) be a normalized substitution. Then \( t\sigma \) is normalized.
Proposition 13. If a standard constraint system $S$ has a model $\sigma$ within the DY deduction system, then $S$ has a model within the $\text{DY + ACI}$ deduction system.

Proof. It is enough to consider the same model $\sigma$ in $\text{DY + ACI}$. As $S\sigma$ is normalized and as $\text{DY + ACI}$ includes all the rules from DY, it is easy to show using the same derivation that proves $\sigma$ to be a model in DY, that $\sigma$ remains a model of $S$ in $\text{DY + ACI}$. □

Next we show that we can build a model of a constraint system within DY from a model of this constraint system within $\text{DY + ACI}$.

Proposition 14. For any $\text{DY + ACI}$ rule $l_1, \ldots, l_k \rightarrow r$, if $l_i$ are normalized for all $i = 1, \ldots, k$ then $\delta(r) \in \text{Der}_{\text{DY}}(\{\delta(l_1), \ldots, \delta(l_k)\})$.

Proof. Let us consider all possible rules:

1. $t_1, t_2 \rightarrow \text{bin}(t_1, t_2)$ for bin $\neq \text{sig}$. As $t_1$ and $t_2$ are normalized, then $\text{bin}(t_1, t_2) = \text{bin}(t_1, t_2)$. We can see, that $\delta(\text{bin}(t_1, t_2)) = \text{bin}(\delta(t_1), \delta(t_2)) \in \text{Der}_{\text{DY}}(\{\delta(t_1), \delta(t_2)\})$.

2. $t_1, \text{priv}(t_2) \rightarrow \text{sig}(t_1, \text{priv}(t_2))$. As $t_1$ and $\text{priv}(t_2)$ are normalized, then $\text{sig}(t_1, \text{priv}(t_2)) = \text{sig}(t_1, \text{priv}(t_2))$. We can see that $\delta(\text{sig}(t_1, \text{priv}(t_2))) = \text{sig}(\delta(t_1), \delta(\text{priv}(t_2))) \in \text{Der}_{\text{DY}}(\{\delta(t_1), \delta(\text{priv}(t_2))\})$.

3. $t_1, \ldots, t_m \rightarrow \text{bin}(r(t_1, \ldots, t_m))$. The fact that $\text{elem}(r(t_1, \ldots, t_m)) = \bigcup_{i=1}^{m} \text{elem}(t_i)$ follows from $t_i \rightarrow \text{bin}(t_i)$ (for all $i$) and Lemma 4.

We can (DY)-derive from $\{\delta(t_i)\}$ any term in $\delta(\text{elem}(t_i))$ trivially if $t_i \neq \cdot(L)$ and by applying rules $\delta(\text{pair}(s_1, s_2)) \rightarrow \text{bin}(s_1)$ and $\delta(\text{pair}(s_1, s_2)) \rightarrow \text{bin}(s_2)$ otherwise (by induction on the size of $t_i$).

One can observe, that $\delta(t)$ is a pairing (composition of $\cdot(L)$) operator with itself of $\delta(\text{elem}(t))$ (by definition of $\delta(\cdot)$ and normalization function). Then, as $\delta(t)$ is limited in size, we can (DY)-derive $\delta(t)$ from $\delta(\text{elem}(t))$ by iterative use of rule $\delta(\text{pair}(s_1, s_2)) \rightarrow \text{bin}(s_1, s_2)$, if needed.

Thus, first we can derive $\delta(\text{elem}(t_i))$ for all $i$, and then rebuild (derive with composition rules) $\delta(\text{bin}(t_1, \ldots, t_m))$.

4. $\text{enc}(t_1, t_2) \rightarrow \text{bin}(t_1, t_2)$. Since $\text{enc}(t_1, t_2)$ is normalized, we have $t_1 = \text{bin}(t_1)$ and $t_2 = \text{bin}(t_2)$. Thus, $\delta(t_1, t_2) = \text{bin}(\delta(t_1), \delta(t_2))$ and this is what we need, as $\delta(\text{enc}(t_1, t_2)) = \delta(\text{bin}(t_1), \delta(t_2))$.

5. $\text{pair}(t_1, t_2) \rightarrow \text{bin}(t_1)$. Similar case.

6. $\text{pair}(t_1, t_2) \rightarrow \text{bin}(t_2)$. Similar case.

7. $\text{aenc}(t_1, t_2) \rightarrow \text{bin}(t_1)$. Similar case. Note, that $\delta(\text{pair}(t_2)) = \text{bin}(\delta(t_2))$.

8. $\cdot(t_1, \ldots, t_m) \rightarrow \text{bin}(t_1)$. As said above, $\delta(\text{elem}(\cdot(t_1, \ldots, t_m))) \subseteq \text{Der}_{\text{DY}}(\{\delta(\cdot(t_1, \ldots, t_m))\})$; and as $\delta(\text{elem}(t_i)) \subseteq \delta(\text{elem}(\cdot(t_1, \ldots, t_m)))$, we can (DY)-derive (by composition rules) $\delta(t_1)$ from $\delta(\text{elem}(t_i))$. □

Proposition 15. Given a standard constraint system $S$ and a normalized model $\sigma$ of $S$ in $\text{DY + ACI}$ then, for any $t \in \text{Sub}(S)$ we have $\delta(t\sigma) = t\delta(\sigma)$.

Proof. The proof is by induction as in Proposition 2.

- Let $|\text{Sub}(t)| = 1$. Then either $t \in A$ or $t \in X$. Both are trivial cases.
- Assume that for some $k \geq 1$ if $|\text{Sub}(t)| \leq k$, then $\delta(t\sigma) = t\delta(\sigma)$.
- Show, that for $t$ such that $|\text{Sub}(t)| \geq k + 1$, where $t = \text{bin}(p, q)$ or $t = \text{priv}(p)$ and $|\text{Sub}(p)| \leq k$ and $|\text{Sub}(q)| \leq k$, statement $\delta(t\sigma) = t\delta(\sigma)$ is still true. We have:
Proof. Let $E$ be a standard constraint system and $\sigma$ be a normalized model in $DY + ACI$. Then $\delta(\sigma)$ is a model in $DY$ of $S$.

Theorem 2. Let $\mathcal{S} = \{E_i \vdash t_i\}_{i=1,...,n}$ be a standard constraint system and $\sigma$ be a normalized model in $DY + ACI$. Then $\delta(\sigma)$ is a model in $DY$ of $S$.

Proof. Let $E \vdash t$ be any element of $\mathcal{S}$. As $\sigma$ is a model of $\mathcal{S}$, then $E \sigma \vdash \sigma t \sigma \vdash \mathcal{S}$. As $\sigma$ is normalized and $\mathcal{S}$ is standard, using Lemma 24 we have $E \sigma \vdash t \sigma$ and $\mathcal{S} \sigma \vdash \mathcal{S}$. Then, $\sigma \vdash \mathcal{S}$. That means there exists a $DY + ACI$ derivation $D = \{A_0, \ldots, A_k\}$ such that $A_0 = E \sigma$ and $t \sigma \in A_k$.

By Proposition 14 and Lemma 14 (which also works for $DY$ case) we can easily prove that if $k > 0$, $\delta(A_j) \subseteq \operatorname{Der}_{DY}(\delta(A_j \sigma \vdash \mathcal{S} \sigma \vdash \mathcal{S}))$, $j = 1, \ldots, k$. Note, that $\delta(A)$ is a set of standard terms (and thus, normalized) for any set of terms $A$. Then, applying the transitivity property of $\operatorname{Der}_{DY}(\cdot)$ (Lemma 14 for $DY$) $k$ times, we have that $\delta(A_k) \subseteq \operatorname{Der}_{DY}(\delta(A_0))$. In the case where $k = 0$, the statement $\delta(A_k) \subseteq \operatorname{Der}_{DY}(\delta(A_0))$ is also true.

Using Proposition 15 we have that $\delta(A_0) = \delta(E \sigma) = E \delta(\sigma)$, as $E \subseteq \mathcal{S}$. The same for $t$: $\delta(t \sigma) = t \delta(\sigma)$, and as $t \sigma \in A_k$, we have $t \delta(\sigma) \in \delta(A_k)$.

Thus, we have that $t \delta(\sigma) \in \delta(A_k) \subseteq \operatorname{Der}_{DY}(\delta(A_0)) = \operatorname{Der}_{DY}(E \delta(\sigma))$, that means $\delta(\sigma)$ $DY$-satisfies any constraint of $\mathcal{S}$. □

We present an example illustrating the theorem.

Example 8. Let us consider a standard constraint system similar to Example 5.

$$
\mathcal{S} = \left\{ \begin{array}{l}
\text{enc}(x, a) \cdot \text{pair}(c, a) \vdash b \\
\text{pair}(x, c) \vdash a \end{array} \right\}.
$$

Using Algorithm 1, we can get a model of $\mathcal{S}$ within $DY + ACI$, as in Example 6, $\sigma = \{x \mapsto \cdot(a, b, c)\}$. Then, by applying the transformation $\delta(\cdot)$, we will get $\sigma' = \delta(\sigma) = \{x \mapsto \text{pair}(a, \text{pair}(b, c))\}$. We can see that $\sigma'$ is also a model of $\mathcal{S}$ within $DY$ (as it was proven in Theorem 2).

Corollary 16 (of Theorem 2 and Proposition 13). A standard constraint system $\mathcal{S}$ is satisfiable within $DY$ iff it is satisfiable within $DY + ACI$.

Theorem 3. Satisfiability of $DY$ constraint systems is in $NP$.

Proof. It is a consequence of Proposition 12 and Corollary 16. □

Then, since the problem is known to be $NP$-hard (Rusinowitch and Turuani, 2003) for the particular case of well-formed constraints, we obtain the $NP$- hardness and thus, $NP$-completeness:

Corollary 17. Satisfiability of $DY$ constraint systems is $NP$-complete.

6. Conclusions

In this work we have presented a decision algorithm for the satisfiability of general constraint systems within Dolev–Yao deduction system possibly extended with an ACI symbol that can be used to represent sets of terms. The complexity of the algorithm was proved to be in $NP$.

We have also given two applications of the presented result: protocol insecurity with non-communicating intruders and discovering XML-based attacks. Moreover, recent works (Mödersheim et al., 2013; Kassem et al., 2013) have shown the interest of general intruder constraints for analyzing
security problems of mobile code encountered in web-browsers, smartphones, and virtualized infrastructures. In this approach, an attacker can construct new processes from his knowledge by using the constructors of the ambient calculus, in the same way as a Dolev Yao intruder constructs messages in protocol verification.

As future works one may consider the verification of equivalence properties such as anonymity or secrecy of a ballot in vote. For instance one has to show that there is no action for an attacker that makes distinguishable two protocol executions with different identities or vote values. In presence of multiple intruders this may require to extend a decision procedure for trace equivalences (Baudet, 2005) using our techniques. Another challenging extension of our result would be a decision procedure for general constraints with negation. Negation constraints are particularly useful to model non-disclosure policies when generating distributed orchestration of secure services, following the approach introduced in Avanesov et al. (2012).

Appendix A. General constraints for subterm theories

We demonstrate here that the well-formedness property is a strong restriction for constraint systems: in some theories satisfiability of well-formed constraints may be decidable, while satisfiability of general constraints may be undecidable.

We define a subterm deduction system to be a finite set of rules of the following forms:

1. composition rules: for all public functional symbols \( f, x_1, \ldots, x_k \rightarrow f(x_1, \ldots, x_k) \)
2. decomposition rules: \( t_1, \ldots, t_m \rightarrow s \), where \( s \) is a subterm of \( t_i \) for some \( i \).

We show that the satisfiability of a constraint system within a subterm deduction system is undecidable in general. More precisely:

**Instance:** a subterm deduction system \( D \), a constraint system \( C \).

**Question:** is \( C \) satisfiable?

To show this, we reduce the halting problem of a Deterministic Turing Machine (TM) \( M \) that works on a single tape. We consider the tape alphabet \( \Gamma = \{0, 1, b\} \), and \( b \) is the blank symbol. The states of the TM \( M \) are in a finite set \( Q = \{q_1, q_2, \ldots, q_n\} \). W.l.o.g. we can assume that \( q_1 \) (resp. \( q_n \)) is the unique initial (resp. accepting) state.

In order to represent Turing machine configurations as terms we shall introduce a set of variables \( \mathcal{X} \) and an alphabet \( \mathcal{F} \)

\[
\mathcal{F} := \{0, 1, b, \bot\} \cup \mathcal{Q},
\]

where \( \mathcal{F} \setminus \{\bot\} \) are public functional symbols.

The TM configuration with tape \( \bot abcd \) \( \bot \) (where \( \bot \) is an endmarker), with symbol \( d \) under the head, and state \( q \) will be represented by the following term of \( q(c(b(a(\bot)), d(e(\bot), x)) \) where \( x \in \mathcal{X} \) and \( a, b, c, d, e \in \{0, 1, b\} \).

The composition rules we consider for the TM are \( u \rightarrow f(u) \) for each \( f \in \{0, 1, b\} \) and \( u, v, w \rightarrow q(u, v, w) \) for each \( q \in \mathcal{Q} \). For each TM transition of \( M \) we will introduce some decomposition deduction rule that can be applied on a term representation \( q(u, v, q'(u', v', x')) \) iff the transition can be applied to a configuration represented by \( q(u, v, \bot) \) and generate a configuration represented by \( q'(u', v', \bot) \).

For each TM instruction of type: ‘In state \( q \) reading a go to state \( q' \) and write \( b \)’, we define the following rule for \( a, b \in \{0, 1, b\} \):

\[
q(u, a(v), q'(u, b(v), x)) \rightarrow q'(u, b(v), x).
\]

For each instruction of type: ‘In state \( q \) reading a go to state \( q' \) and move right’, we define the following rules for \( a \in \{0, 1, b\} \):

\[
q(u, a(v), q'(a(u), v, x)) \rightarrow q'(a(u), v, x).
\]
A rule is for extending the tape on the right when needed:

\[ q(u, \bot, q'(u, b(\bot), x)) \rightarrow q'(u, b(\bot), x). \]

For each instruction of type: “In state \( q \) reading a go to state \( q' \) and move left”, we define the following rules for \( a \in \{0, 1, b\} \) for each \( b \in \{0, 1, b\} \):

\[ q(b(u), a(v), q'(u, b(a(v)), x)) \rightarrow q'(u, b(a(v)), x). \]

A rule is for extending the tape on the left when needed:

\[ q(\bot, v, q'(\bot, b(v), x)) \rightarrow q'(\bot, b(v), x). \]

The resulting deduction system \( D_M \) is obviously a subterm deduction system.

Let us consider a constraint \( S \) to be solved modulo \( D_M \):

\[ \{q_1 (\bot, \bot, x)\} \triangleright q_n (\bot, \bot, y). \]

This constraint is satisfiable iff there is a sequence of transitions of \( M \) from a configuration with initial state \( q_1 \) and empty tape to a configuration with an accepting state and empty tape. Hence the constraint solving problem is undecidable.

Let us recall the definition of some properties of constraint systems. These two properties are natural for modeling standard security protocols:

**Variable origination:** \( \forall i, \forall x \in \text{Vars}(E_i) \exists j < i \ x \in \text{Vars}(t_j), \)

**Monotonicity:** \( j < i \implies E_j \subseteq E_i. \)

Note that \( \{q_1 (\bot, \bot, x)\} \triangleright q_n (\bot, \bot, y) \} \) is obviously monotonic.

As a consequence, satisfiability of monotonic constraint systems (but without variable origination) is undecidable. Here is another constraint system, where variable origination is satisfied, but monotony is not. It can be used for reducing the halting problem again:

\[ \{[\bot] \triangleright x, \{q_1 (\bot, \bot, x)\} \triangleright q_n (\bot, \bot, y) \}. \]

As a consequence, satisfiability of constraint systems with variable origination (but without monotonicity) is undecidable.

We should note by contrast (see Baudet, 2005) that constraint solving in subterm convergent theories is decidable if the constraint system \( S = \{E_i \triangleright t_i\}_{i=1}^{n} \) satisfies both variable origination and monotonicity. Moreover, the problem of constraint system satisfiability for a subterm deduction system encoding a Deterministic Turing Machine as above can be reduced to the problem of constraint system satisfiability within a subterm convergent equational theory.

If we consider a Universal Turing Machine\(^3\) with a single final state \( q_n \) and that halts with an empty tape if and only if the Turing Machine with the given code terminates on the given input, then by reusing the encoding above we will obtain a fixed subterm deduction system \( D_U \) for which the following problem is undecidable:

**Instance:** A constraint system \( C \).

**Question:** is \( C \) satisfiable within \( D_U \)?

This is due to the fact that the halting problem of a TM with code \( TMcode \) on input \( Input \) is encoded into solving the following constraint:

\[ \{q_1 (\bot, TMcode + Input, x)\} \triangleright q_n (\bot, \bot, y). \]

---

\(^3\) Thanks to R. Küsters for the idea.
Appendix B. Proofs

B.1. Proof of Lemma 12

By induction on \(|\text{Sub}(t)|\).

- If \(|\text{Sub}(t)| = 0\), implies \(t = a \in A\) and then \(\text{elems}(a) = \{a\}\), i.e., the equality becomes trivial.
- Suppose that for any \(t : |\text{Sub}(t)| < k (k > 1)\), \(H(t) = \bigcup_{p \in \text{elems}(t)} H(p)\) holds.
- Given a term \(t : |\text{Sub}(t)| = k, k > 1\). We should prove \(H(t) = \bigcup_{p \in \text{elems}(t)} H(p)\).
  - \(t = \text{priv}(t)\) or \(t = \text{bin}(p, q)\). In both cases, \(\text{elems}(t) = \{t\}\), and thus, the equality is trivial.
  - \(t = \cdot (L)\). Note that \(\forall s \in L, |\text{Sub}(s)| < k\). Then, on one hand, \(H(L) = \bigcup_{p \in \text{elems}(L)} H(p) = \bigcup_{p \in \text{elems}(p)} H(p)\). On the other hand, \(\bigcup_{p \in \text{elems}(L)} H(p) = \bigcup_{p \in \text{elems}(p)} H(p)\). Thus, \(H(t) = \bigcup_{p \in \text{elems}(t)} H(p)\).

B.2. Proof of Lemma 6

First we prove that \(\text{elems}(t) \subseteq \text{QSub}(t)\). We use a proof by induction on \(|\text{Sub}(t)|\).

- If root \((t) \neq \cdot\), then \(\text{elems}(t) = \{t\} \subseteq \text{QSub}(t)\). This case includes all \(t\) such that \(|\text{Sub}(t)| = 1\). Thus we need to consider only \(t = \cdot (L)\).
- Suppose that for any \(t : |\text{Sub}(t)| < k (k > 1)\), the statement holds.
- If for some \(t\) we have \(|\text{Sub}(t)| = k, k > 1\), then \(\text{elems}(t) = \bigcup_{p \in \text{L}} \text{elems}(p)\) and \(\text{QSub}(t) = \{t\} \bigcup_{p \in \text{L}} \text{QSub}(p)\). And since \(|\text{Sub}(p)| < k\) using the induction supposition we obtain the wanted statement.

Now we show that \(\text{QSub}(t) \subseteq \text{Sub}(t)\). Again, applying proof by induction on \(|\text{Sub}(t)|\) we have:

- If \(|\text{Sub}(t)| = 0\), then \(\text{QSub}(t) = \text{Sub}(t) = \{t\}\).
- Suppose that for any \(t : |\text{Sub}(t)| < k (k > 1)\), the statement holds.
- If for some \(t\) we have \(|\text{Sub}(t)| = k, k > 1\) then
  - \(t = \text{bin}(t_1, t_2)\). Then \(\text{QSub}(t) = \{t\} \bigcup_{t \subseteq \text{QSub}(t_1)} \text{QSub}(t_2)\) and \(\text{Sub}(t) = \{t\} \bigcup_{t \subseteq \text{Sub}(t_1)} \text{Sub}(t_2)\), where \(\max(|\text{Sub}(t_1)|, |\text{Sub}(t_2)|) < k\). And then using induction supposition we can conclude for this case.
  - \(t = \cdot (t_1, \ldots, t_m)\). Then we have \(\text{QSub}(t) = \{t\} \bigcup_{p \in \text{Sub}(t_1, \ldots, t_m)} \text{QSub}(p) \subseteq \text{Sub}(t)\) (using the already proved part of the property) \(\subseteq \{t\} \bigcup_{p \in \text{Sub}(t_1, \ldots, t_m)} \text{QSub}(p) = \text{QSub}(t) = \{t\} \bigcup_{p \in \text{Sub}(t_1, \ldots, t_m)} \text{QSub}(p) \subseteq \text{Sub}(t)\).

B.3. Proof of Proposition 12

As was stated before, the measure of the problem input is \(\text{measure}(S) = n \times |\text{Sub}(S)| + |E(S)|\), where \(S = \{E_i \triangleright t_i\}_{i=1}^{n}\).

Algorithm 1 returns a normalized proof \(\sigma\) for the decision problem if it exists. Moreover, \(|\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(S)|\) for any \(x \in \text{Vars}(S)\).

First, we will normalize \(\sigma\). From Lemma 21 follows, that we can do it for the time \(T_{\tau \gamma} \leq P_{\tau \gamma} (\text{measure}(S \sigma))\), where \(P_{\tau \gamma}\) is some polynomial with non-negative coefficients of some degree \(m' > 0\).

From §3.2 we know that checking the derivability of a normalized ground term \(g\) from a set of normalized ground terms \(A\) takes a polynomial time (depending on \(|\text{Sub}(G \cup \{g\})|\)). That is, there exists a polynomial \(P_{g}\) with non-negative coefficients, such that the number of operations (execution time) to verify the derivability (\(g\) from \(G\)) will be limited by \(P_{g}(|\text{Sub}(G \cup \{g\})|)\).
Then the execution time for checking a set of ground constraints \(\{G_i \triangleright g_i\}_{i=1,...,n}\) will be limited by \(\sum_{i=1}^n P_g(|\text{Sub}(G_i \cup \{g_i\})|)\).

To prove that the algorithm is in \(NP\) we need to show that the execution time for checking a solution is polynomially limited by the input measure, i.e. there exists a polynomial \(P\), such that the execution time does not exceed \(O(P(n \times |\text{Sub}(S)| + |E(S)|))\) steps.

In our case, the execution time \(T\) of a check will be \(T = T_{\text{ground}} + T_{\text{val}} + T_{g}\), where \(T_{\text{val}}\) is the time needed to check the validity of terms in \(S\sigma\), and \(T_g\) is the time needed for checking ground derivability of \(S\sigma\): \(T_g \leq \sum_{i=1}^n P_g(|\text{Sub}(\tau_i)\sigma|)\). If \(P_g\) is a polynomial, let us say, of degree \(m' > 0\), with non-negative coefficients, we can use the fact that for any positive integers \(x_1,\ldots,x_k\) we have \(\sum_{i=1}^k P_g(x_i) \leq P_g(\sum_{i=1}^k x_i)\). Then we have \(T_g \leq P_g(\sum_{i=1}^n |\text{Sub}(\tau_i)\sigma|)\) and by Lemma 22 we have \(T_g \leq P_g(\sum_{i=1}^n |\text{Sub}(\tau_i)\sigma|)\); using the same lemma, we have

\[
T_g \leq P_g \left( \sum_{i=1}^n \left( |\text{Sub}(\tau_i)\sigma| + |\text{Sub}\left( \bigcup_x (x\sigma) \right)\right) \right) \\
\leq P_g \left( \sum_{i=1}^n \left( |\text{Sub}(\tau_i)\sigma| + |\text{Sub}(\tau_i)\sigma| + \sum_x |\text{Sub}(x\sigma)\sigma| \right) \right) \\
\leq P_g \left( \sum_{i=1}^n (2 \times |\text{Sub}(S)\sigma| + n \times \sum_x |\text{Sub}(x\sigma)\sigma| \right) \\
\leq P_g \left( 2 \times n \times |\text{Sub}(S)\sigma| + n \times \sum_x (2 \times |\text{Sub}(S)\sigma|) \right) \\
\leq P_g \left( 4 \times n \times |\text{Sub}(S)\sigma| + |\text{Sub}(S)\sigma| + |E(S)\sigma| \right) \\
= 0 \left( (\text{measure}(S))^{2m'} \right).
\]

On the other hand, let us consider \(T_{\text{ground}}\). We have \(T_{\text{ground}} \leq P_{\text{ground}}(n \times |\text{Sub}(S\sigma)| + |E(S\sigma)|)\). One can see that the number of edges in DAG-representation of \(S\sigma\) (where every variable \(x\) of \(S\) is replaced by \((x\sigma)\)) will not exceed the number of edges in \(S\) plus the number of edges of all \(x\sigma\): \(|E(S\sigma)| \leq |E(S)| + \sum_{x \in \text{Vars}(S)} |E(x\sigma)|\). And since \(\sigma\) is normalized, we can use Lemma 20: \(T_{\text{ground}} \leq P_{\text{ground}}(n \times |\text{Sub}(S\sigma)| + |E(S)| + \sum_{x \in \text{Vars}(S)} |\text{Sub}(x\sigma)|)\).

Then, using Lemma 5 we obtain \(\text{Sub}(S\sigma) = \text{Sub}(S)\sigma \cup \text{Sub}(\text{Vars}(S)\sigma)\), and thus, \(|\text{Sub}(S\sigma)| \leq |\text{Sub}(S)\sigma| + \sum_{x \in \text{Vars}(S)} |\text{Sub}(x\sigma)|\). From \(\forall T \subseteq T, |T\sigma| \leq |T|\) follows \(|\text{Sub}(S)\sigma| \leq |\text{Sub}(S)|\). Since \(|\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(S)|\) and \(|\text{Vars}(S)| \leq |\text{Sub}(S)|\), we obtain \(|\text{Sub}(S\sigma)| \leq |\text{Sub}(S)| + 2 \times (|\text{Sub}(S)|)^2\).

In the same way, \(\sum_{x \in \text{Vars}(S)} (|\text{Sub}(x\sigma)|)^2 \leq |\text{Sub}(S)| \times (2 \times (|\text{Sub}(S)|)^2\). Therefore, \(T_{\text{ground}} \leq P_{\text{ground}}(n \times ((|\text{Sub}(S)| + 2 \times (|\text{Sub}(S)|)^2) + |E(S)| + |\text{Sub}(S)| \times (2 \times (|\text{Sub}(S)|)^2)^2) = 0 \left( (\text{measure}(S))^{3m''} \right)\).

Note that once \(S\sigma\) is normalized, the validity can be checked in linear time \(T_{\text{val}} = O(\text{measure}(S\sigma))\).

Using the same reasoning as for \(T_{\text{ground}}\), we obtain \(T_{\text{val}} = O(\text{measure}(S)^3)\).

Summing up, \(T = O \left( (\text{measure}(S))^{3m''} + (\text{measure}(S))^{2m'\prime} + (\text{measure}(S)^3) \right)\). This shows that checking a solution returned by the algorithm takes polynomial time, giving the expected complexity.

### B.4. Proof of Proposition 3

Let us consider all the cases of \(DY + ACI\) rules:

- \(t_1, t_2 \rightarrow \top\) pair \(t_1, t_2\) \(\top\). We have two cases:
\[ \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{pair}(t_1, t_2)) = \pi u \sigma. \] Then we have \( \pi(H(\text{pair}(t_1, t_2))) = \pi(H(\text{pair}(\pi(t_1), \pi(t_2)))) = \pi(\pi(H(t_1)), \pi(H(t_2))) \) and then \( \pi(H(\text{pair}(t_1, t_2))) \in \text{Der}(\pi(H(t_1)), \pi(H(t_2))). \)

\[ \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{pair}(t_1, t_2)) = \pi u \sigma. \] Then by definition, \( \text{Lemma 13 and Lemma 10} \)
\[ \pi(H(\text{pair}(t_1, t_2))) = \pi(H(\pi(t_1) \cup H(t_2))) \in \text{Der}(H(t_1) \cup H(t_2)), \]
By \( \text{Lemma 10}, \pi(H(t_1)) \subseteq \text{Der}(\pi(H(t_1))) \) and \( H(t_2) \subseteq \text{Der}(\pi(H(t_2))), \)
then by \( \text{Lemma 14}, \pi(H(t_1)) \cup \pi(H(t_2)) \subseteq \text{Der}(\pi(H(t_1))) \cup \pi(H(t_2)), \)
Now, by applying \( \text{Lemma 14}, \) we have \( \pi(H(\text{pair}(t_1, t_2))) \in \text{Der}(\pi(H(t_1))) \cup \pi(H(t_2))) \).

So, in this case \( \pi(H(r)) \in \text{Der}(\pi(H(t_1)), \pi(H(t_2))). \)

- \( t_1, t_2 \rightarrow \text{\textit{enc}}(t_1, t_2). \) Proof of this case can be done by analogy of previous one.

- \( \{t_1, t_2\} \rightarrow \text{\textit{aenc}}(t_1, t_2). \) The same.

- \( t_1, \text{priv}(t_2) \rightarrow \text{\textit{sig}}(t_1, \text{priv}(t_2)). \)

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(t_1, \text{priv}(t_2)) = \pi u \sigma. \) Then \( \pi(H(\text{sig}(t_1, \text{priv}(t_2)))) = \pi(H(\pi(t_1), \text{priv}(t_2))) \) and then \( \pi(H(\text{sig}(t_1, \text{priv}(t_2)))) \in \text{Der}(\pi(H(t_1))), \pi(H(\text{priv}(t_2)))) \) as \( \pi(H(\text{priv}(t_2))) = \pi(H(t_2)). \)

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(t_1, \text{priv}(t_2)) = \pi u \sigma. \) This case can be proved in similar way as done for \( \{t_1, t_2\} \rightarrow \text{\textit{pair}}(t_1, t_2). \)

- \( t_1, \ldots, t_m \rightarrow \text{\textit{f}}(t_1, \ldots, t_m). \)

On one hand, \( \pi(H(\cdot(t_1, \ldots, t_m))) = \pi(H(\cdot(t_1, \ldots, t_m))) \in \pi(H(t_1) \cup \ldots \cup H(t_m)) \).
And thus, by \( \text{Lemma 14}, \) \( \pi(H(\cdot(t_1, \ldots, t_m))) \in \pi(H(t_1), \ldots, \pi(H(t_m))). \)

- \( \text{\textit{enc}}(t_1, t_2) \rightarrow \text{\textit{t}}(t_1) \). Here we have to show that \( \pi(H(\pi(t_1))) \) is derivable from \( \{\pi(H(\text{enc}(t_1, t_2))), \pi(H(\text{f}(t_1)))\}. \) Consider two cases:

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{enc}(t_1, t_2) = \pi u \sigma. \)

Then \( \pi(H(\text{enc}(t_1, t_2))) = \pi(\pi(H(t_1)), \pi(H(t_2))) \) and \( \pi(H(\pi(t_1))) = \pi(H(t_1)) \in \text{Der}((\pi(H(t_1)), \pi(H(t_2)))) \).

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{enc}(t_1, t_2)) = \pi u \sigma. \)

Then \( \pi(H(\text{enc}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2)). \) From \( \text{Lemma 10}, \) we have \( \pi(H(t_1) \cup H(t_2)) \subseteq \text{Der}((\pi(H(\text{enc}(t_1, t_2))))), \) then \( \pi(H(t_1)) \subseteq \text{Der}((\pi(H(\text{enc}(t_1, t_2))))). \)

Therefore, by \( \text{Lemma 13} \) and \( \text{Lemma 14}, \) we have \( \pi(H(t_1)) = \pi(H(t_1)) \in \text{Der}(\pi(H(\text{enc}(t_1, t_2)))) \).

- \( \text{\textit{aenc}}(t_1, t_2) \rightarrow \text{\textit{t}}(t_1). \) Here we have to show that \( \pi(H(\pi(t_1))) \) is derivable from \( \{\pi(H(\text{aenc}(t_1, t_2))), \pi(H(\text{f}(t_1)))\}. \) Consider two cases:

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{enc}(t_1, t_2)) = \pi u \sigma. \)

Then \( \pi(H(\text{enc}(t_1, t_2))) = \pi(\pi(H(t_1)), \pi(H(t_2))), \) and \( \pi(H(\pi(t_1))) = \pi(H(t_1)) \in \text{Der}((\pi(H(t_1)), \pi(H(t_2)))) \).

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{enc}(t_1, t_2)) = \pi u \sigma. \)

Then \( \pi(H(\text{enc}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2)). \) By \( \text{Lemma 10}, \) we have \( \pi(H(t_1) \cup H(t_2)) \subseteq \text{Der}((\pi(H(\text{enc}(t_1, t_2))))), \) then \( \pi(H(t_1)) \subseteq \text{Der}((\pi(H(\text{enc}(t_1, t_2))))). \)

Therefore, by \( \text{Lemma 13} \) and \( \text{Lemma 14}, \) we have \( \pi(H(t_1)) = \pi(H(t_1)) \in \text{Der}(\pi(H(\text{enc}(t_1, t_2)))) \).

- \( \text{\textit{pair}}(t_1, t_2) \rightarrow \text{\textit{t}}(t_1). \) Here, as usual, we consider two cases:

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{pair}(t_1, t_2)) = \pi u \sigma. \)

Then \( \pi(H(\text{pair}(t_1, t_2))) = \pi(H(t_1), \pi(H(t_2))) \) and then \( \pi(H(\pi(t_1))) = \pi(H(t_1)) \in \text{Der}((\pi(H(t_1), \pi(H(t_2))))). \)

- \( \exists u \in \text{Sub}(S) \text{ such that } \pi(\text{pair}(t_1, t_2)) = \pi u \sigma. \)

Then \( \pi(H(\text{pair}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2)). \) Then by \( \text{Lemma 10}, \) we obtain \( \pi(H(t_1) \cup H(t_2)) \subseteq \text{Der}((\pi(H(\text{pair}(t_1, t_2))))), \) then \( \pi(H(t_1)) \subseteq \text{Der}((\pi(H(\text{pair}(t_1, t_2))))). \)

Therefore, by \( \text{Lemma 14}, \) we have \( \pi(H(t_1)) \in \text{Der}(\pi(H(\text{pair}(t_1, t_2)))) \).

- \( \text{\textit{pair}}(t_1, t_2) \rightarrow \text{\textit{f}}(t_1). \) Proof like above.

- \( (t_1, \ldots, t_m) \rightarrow \text{\textit{f}}(t_1). \) We have \( \pi(H(\cdot(t_1, \ldots, t_m))) = \pi(H(t_1) \cup \ldots \cup H(t_m)). \) Then by \( \text{Lemma 10}, \)
\( \pi(H(t_1) \cup \ldots \cup H(t_m)) \subseteq \text{Der}(\pi(H(\cdot(t_1, \ldots, t_m)))) \).

As \( \pi(\langle t_i \rangle) \in \text{Der}(\langle H (\langle t_i \rangle) \rangle) \), by Lemma 14 we have \( \pi(H(\langle t_i \rangle)) = \pi(H(\langle t_i \rangle)) \in \text{Der}(\langle H(\langle t_1, \ldots, t_2 \rangle) \rangle) \).

As all possible cases satisfy lemma conditions, we proved the lemma.

B.5. Proof of Lemma 22

First, we show an auxiliary statement:

Lemma 25. For any term \( t \), \( \forall s \in \text{Sub}(\langle t \rangle) \exists s' \in \text{Sub}(t) : s = \langle s' \rangle \)

Proof. Suppose the opposite and let us take \( s \in \text{Sub}(\langle t \rangle) \) with maximal \( |\text{Sub}(s)| \) that does not satisfy the desired property. Note that the “biggest” term in \( \text{Sub}(\langle t \rangle) \), i.e. \( \langle t \rangle \), does satisfy the property, as we can choose \( s' = t \in \text{Sub}(t) \). By definition of \( \text{Sub}(\cdot) \) if \( s \in \text{Sub}(\langle t \rangle) \) and \( s \not\in \langle t \rangle \) then \( \exists r \in \text{Sub}(\langle t \rangle) \) such that

- \( r = \text{bin}(p, s) \) or \( r = \text{bin}(s, p) \) or \( r = \text{priv}(s) \). Without loss of generality we consider only the first case \( (r = \text{bin}(p, s)) \) as other ones are similar. As \( |\text{Sub}(r)| > |\text{Sub}(s)| \), there exists \( r' \in \text{Sub}(t) \) such that \( r = \langle r' \rangle \). By definition of \( \langle r' \rangle \):
  - either \( r' = \text{bin}(p', s') \) and \( \text{priv}(r') = p \) and \( \langle s' \rangle = s \). As \( s' \in \text{Sub}(r') \subseteq \text{Sub}(t) \) the property is proved.
  - or \( r' = \langle L \rangle \) and \( \text{dom}(L) = \{r\} \). Since \( \forall q \in \text{dom}(L), \text{root}(q) \neq \cdot \), then \( \exists q \in \text{dom}(L) : q = \text{bin}(p', s') \) and \( \text{priv}(q') = p \) and \( \langle s' \rangle = s \). Using Lemma 6 we have \( s' \in \text{Sub}(t) \).

- \( r = \langle L \rangle \) and \( s \in L \). Then \( \langle s \rangle \in \text{Sub}(t) \) we have \( \exists r' \in \text{Sub}(t) : \langle r' \rangle = r \). Thus \( r \) is normalized \( (r = \langle r' \rangle = \langle r'' \rangle = \langle r''' \rangle = \langle r'' \rangle = r \rangle \), and thus, root \( (s) \neq \cdot \). Then by definition of \( \langle r'' \rangle \) we have \( r'' = \langle L \rangle \) and \( L \equiv \text{dom}(L) \), and thus, \( s \in \text{dom}(L) \), that is \( \exists s' \in \text{dom}(L) : s = \langle s' \rangle \). Using again Lemma 6 we have \( s' \in \text{Sub}(t) \). \( \square \)

Then, using Lemma 25 and the fact that \( \langle r \rangle \) is deterministic, we obtain \( \forall p, q \in \text{Sub}(\langle t \rangle) : p \neq q \exists p', q' \in \text{Sub}(t) : p = \langle p' \rangle \land q = \langle q' \rangle \land p', \cdot \neq q \). And thus, \( |\text{Sub}(\langle t \rangle)| \leq |\text{Sub}(t)| \).

B.6. Proof of Proposition 8

Proof. From Proposition 1 we know that if \( \sigma' \) is a model of \( S \) then \( \langle \sigma' \rangle \) is a model of \( S \). Then, there exists a substitution \( \theta : \text{dom}(\theta) = \text{dom}(\langle \sigma' \rangle), \text{img}(\theta) \subseteq \text{dom}(\theta), \sigma'' = \langle \sigma' \rangle |_{\text{img}(\theta)} \) (where \( \theta |_{\cdot} \) is a substitution obtained from \( \theta \) by narrowing its domain to set \( V \) and \( \sigma'' \) is a model of \( \langle S \rangle \) such that \( x\sigma'' \neq y\sigma'' \), if \( x \neq y \) (this is true because we can show how to build \( \theta \); given \( \langle \sigma' \rangle \) — simply split \( \text{dom}(\langle \sigma' \rangle) \) into the classes of equivalence modulo \( \langle \sigma' \rangle \), i.e. \( x \equiv y \iff x \in \text{dom}(\sigma' \rangle) \). For every class choose one representative \( [x]_{\equiv} \), and then \( x\theta = [x]_{\equiv} \). Note that \( \theta|_{\cdot} \) is a model of \( \langle S \rangle \).

Then, as \( \sigma'' \) is a model of \( \langle S \rangle \) using Proposition 1, we can say that \( \sigma'' \) is a model of \( \langle \tau \rangle \). Moreover, \( \sigma'' \neq \tau \), if \( \forall x \in \text{dom}(\sigma' \rangle) \neq \cdot \) and \( \sigma'' \) is normalized. Then, we can apply Corollary 6, which gives us existence of conservative model \( \delta \) of \( \langle \tau \rangle \). That is why we can apply Proposition 7: \( \forall x \in \text{Vars}(\langle \tau \rangle), |\text{Sub}(x\delta)| \leq 2 \times |\text{Sub}(\langle \tau \rangle)| \).

Note that using Proposition 1, Lemma 19 and definition of “model”, we can easily show that \( \delta[\theta] \) is a model of \( \langle S \rangle \). Moreover, \( \delta[\theta] \) is normalized. By definition of \( \delta[\theta] \) we can say that \( \forall x \in \text{dom}(\delta[\theta]) \exists y \in \text{dom}(\theta) : x\delta[\theta] = y\delta[\theta] \) and as \( y \in X \) (by definition of \( \theta \)), then \( |\text{Sub}(x\delta[\theta])| = 2 \times |\text{Sub}(\langle \tau \rangle)| \). Applying Lemma 18, we have \( |\text{Sub}(x\delta[\theta])| \leq 2 \times |\text{Sub}(\langle S \rangle)| \).

Summing up, we have a normalized model \( \sigma = \delta[\theta] \) of \( \langle S \rangle \) such that for any \( x \in \text{dom}(\sigma) \) we have \( |\text{Sub}(x\sigma)| \leq 2 \times |\text{Sub}(\langle S \rangle)| \). \( \square \)
References


