Modeling and Verifying Ad Hoc Routing Protocols[1]

Mathilde Arnaud[a,b], Véronique Cortier[b], Stéphanie Delaune[a]

[a]*LSV, CNRS UMR 8643, ENS de Cachan & INRIA Futurs, France*
[b]*CNRS, Loria, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France*

Abstract

Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their infrastructure. In such networks, a central issue, addressed by routing protocols, is to find a route from one device to another. These protocols use cryptographic mechanisms in order to prevent malicious nodes from compromising the discovered route.

Our contribution is twofold. We first propose a calculus for modeling and reasoning about security protocols, including in particular secured routing protocols. Our calculus extends standard symbolic models to take into account the characteristics of routing protocols and to model wireless communication in a more accurate way. Our second main contribution is a decision procedure for analyzing routing protocols for any network topology. By using constraint solving techniques, we show that it is possible to automatically discover (in NPTIME) whether there exists a network topology that would allow malicious nodes to mount an attack against the protocol, for a bounded number of sessions. We also provide a decision procedure for detecting attacks in case the network topology is given a priori. We demonstrate the usage and usefulness of our approach by analyzing protocols of the literature, such as SRP applied to DSR and SDMSR.

1. Introduction

Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their communication infrastructure: each node provides the function of a router and relays packets on paths to other nodes. Finding these paths is a crucial functionality of any ad hoc network. Specific protocols, called *routing protocols*, are designed to ensure this functionality known as *route discovery*.

Prior research in ad hoc networking has generally studied the routing problem in a non-adversarial setting, assuming a trusted environment. Thus, many

[1]The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 258865 (ERC ProSecure project). This work has also been partially supported by the project JCJC VIP ANR-11-JS02-006.

of the currently proposed routing protocols for mobile ad hoc networks are assumed to be used in a friendly environment (e.g., [29, 21]). Recent research has recognized that this assumption is unrealistic and that attacks can be mounted [18, 26, 11]. Since an adversary can easily paralyze the operation of a whole network by attacking the routing protocol, it is crucial to prevent malicious nodes from compromising the discovered routes. Since then, secured versions of routing protocols have been developed to ensure that mobile ad hoc networks can work even in an adversarial setting [37, 18, 27]. Those routing protocols use cryptographic mechanisms such as encryption, signature, MAC, in order to prevent a malicious node from inserting and deleting nodes inside a path. We call these protocols *secured routing protocols* in the sense that efforts have been made to add security although some of these protocols may still have flaws.

Formal modeling and analysis techniques are well-adapted for checking correctness of security protocols. Formal methods have for example been successfully used for analyzing authentication or key establishment security protocols and a multitude of effective frameworks have been proposed (e.g., the Paulson inductive model [28], the strand spaces model [35], the applied-pi calculus [1] or constraints systems [32] to cite only a few). While secrecy and authentication properties are undecidable in the general case [15], many decision procedures have been proposed. For example, secrecy and authentication become NP-complete for a bounded number of sessions [32] and Blanchet has developed a procedure for security protocols encoded as Horn clauses [9]. This yielded various efficient tools for detecting flaws and proving security (e.g., ProVerif [10] or Avispa [6]).

While key-exchange protocols are well-studied in traditional networks, there are very few attempts to develop formal techniques allowing an automated analysis of secured routing protocols. To the best of our knowledge, tools that would allow the security analysis of routing protocols are also missing. Those protocols indeed involve several subtleties that cannot be reflected in existing work. For example, the underlying network topology is crucial to define who can receive the messages sent by a node and the intruder is localized to some specific nodes (possibly several nodes). Moreover, the security properties include e.g., the validity of a route, which differ from the usual secrecy and authentication properties.

Our contributions. The first main contribution of this paper is proposing a calculus, inspired from CBS# [26], which allows ad hoc networks and their security properties to be formally described and analyzed. As in standard formal models for security protocols, we model cryptography as a black box (the perfect cryptography assumption), thus the attacker cannot break cryptography, e.g., decrypt a message without having the appropriate decryption key. In order to represent routing protocols in an accurate way, some features need to be taken into account. Among them:

- *Local knowledge*: not only do nodes perform cryptographic tests (e.g., checking signatures), they also use their local knowledge of the network, e.g., they can check that some nodes are their neighbors.

- *Network topology*: nodes can only communicate (in a direct way) with their neighbors.
- *Broadcast communication*: the main mode of communication is broadcasting and only adjacent nodes receive messages.
- *Internal states*: nodes are not memory-less but store some information in routing tables with impact on future actions.

To take these features into account, we first propose a logic to express the tests performed by the nodes at each step. For instance, it allows a node to check whether a route is "locally" valid, given the information known by that node. There are also some implications for the attacker model. Indeed, in most existing formal approaches, the attacker controls the entire network. This abstraction is suitable for reasoning about classical protocols. However, in the context of routing protocols, this attacker model is too strong and leads to a number of false attacks. The constraints on communication also apply to the attacker. Our model reflects the fact that a malicious node can interfere directly only with his neighbors. It should be noted that we do not take mobility into account in the sense that the topology of the network does not change during our analysis. There are two main reasons for this limitation. First, many flaws can already be detected without any change in the network topology. Second, properties like the validity of a route are of course (temporarily) invalidated during a network topology modification. Therefore, such properties have to be analyzed once the network is stabilized, previous routing protocol executions being possibly included in the initial knowledge of the attacker.

We would like to emphasize that our model is not strictly dedicated to routing protocols but can be used to model many other classes of protocols. In particular, by considering a special network topology where the attacker is at the center of the network, we retrieve the classical model where the attacker controls all the communications. We can thus model as usual all the key exchange and authentication protocols presented e.g., in the Clark & Jacob library [13]. Moreover, since we provide each node with a memory, our model can also capture protocols where a state global to all sessions is assumed for each agent. For example, protocols where an agent should check that a key has not already been accepted in a previous session, in order to protect the protocol against replay attacks.

Our formal model represents all possible executions against an adversary that controls some of the nodes and acts maliciously in these nodes by sending any message that he can construct. Our model is thus infinitely branching. As a first step towards automation, we provide an alternative symbolic semantics, based on constraint systems and we show its correctness and completeness w.r.t. the concrete semantics. This result holds for arbitrary processes (possibly with replication) and for any set of primitives.

Our second main contribution is to provide two NP decision procedures for analyzing routing protocols for a bounded number of sessions and for a large set of standard primitives. For a fixed set of roles and sessions, our first decision procedure enables us to discover whether there exists a network topology and a malicious behavior of some nodes that yield an attack. Using similar ingredients, we can also decide whether there exists an attack, for a network topology chosen by the user. Our two procedures hold for any property that can be expressed in our logic, which includes classical properties such as secrecy as well as properties more specific to routing protocols such as route validity.

The main ingredients of our decision procedures are as follows. Even if we consider a bounded number of sessions, the messages sent by the adversary can be arbitrarily large and may contain arbitrarily many node names. The key result for decidability is that whenever there is an attack then there exists an attack that make use of messages of limited size and of limited number of node names. In particular, we need to show that it is possible to bound the size of the lists that are carried out by the nodes. To this purpose, we first propose a symbolic semantics for our execution model and show how the analysis of routing protocols can be reduced to (generalized) constraint systems solving. We then adapt and generalize existing techniques [14] for solving our more general constraint systems. We show in particular that minimal attacks (whether the underlying network topology is fixed or not) require at most a polynomially bounded number of nodes. We demonstrate the usage and usefulness of our model and techniques by analyzing SRP (Secured Routing Protocol) [27] applied on the protocol DSR (Dynamic Source Routing Protocol) [21]. This allows us to retrieve an attack presented first in [11]. We also analyze the security of SDMSR [8], discovering an attack.

Related work. Recently, several results have been proposed for studying routing protocols. For example, Yang and Baras [36] provide a first symbolic model for routing protocols based on strand spaces, modeling the network topology. They implement a semi-decision procedure to search for attacks and find an attack on AODV [29], a routing protocol (built for friendly environments) that does not include cryptography. Their approach however does not apply to routing protocols using cryptographic primitives for securing communications. Schaller *et al* [34] propose a symbolic model that allows an accurate representation of the physical properties of the network, in particular the speed of the communication. This allows in particular to study distance bounding protocols. Several security proofs are provided for some fixed protocols, formalized in Isabelle/HOL. No generic procedure is proposed for proving security. Even if cryptographic primitives are modeled, this work focuses on timing properties or physical properties. They do not consider e.g., the validity of a route.

Several case studies of important secured routing protocols have been performed. Godskesen [16] provides an analysis of a simplified version of the ARAN [33] protocol with ProVerif, for a given configuration, and captures a relay attack. Marshall [20] uses Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES) to specify the SRP protocol and analyze it. The encoding of SRP is performed on a precise fixed topology, without broadcast, and a relay attack is retrieved. Benetti, Merro and Viganò [7] use the AVISPA tool to automatically analyse some execution scenarios of the ARAN and endairA protocols, and find some attacks on ARAN.

While these last results focus on particular routing protocols, some frameworks have been proposed to model wireless communication and/or routing protocols in a more generic way. Buttyán and Vajda [11] provide a model for routing protocols, in a cryptographic setting. Their model enables them to find attacks on SRP and Ariadne [18]. They provide a security proof (by hand) for a fixed protocol they propose, endairA. Àcs, Buttyàn and Vajda then develop their framework for distance vector routing protocols [3], analysing SAODV [37] and ARAN. They also apply their framework to sensor networks [4], analyzing TinyOS [30]. The work closest to ours is the one of Nanz and Hankin [26]. They propose a process calculus to model the network topology and broadcast communications. They analyze scenarios with special topologies and attacker configuration by computing an over-approximation of reachable states. Their analysis is safe in the sense that it does not find flaws if the protocol is secure. The model proposed in this paper is inspired from their work, adding in particular a logic for specifying the tests performed at each step by the nodes on the current route and to specify the security properties.

To our knowledge, our paper presents the first decidability and complexity result for routing protocols, for arbitrary intruders and network topologies. Moreover, since we reuse existing techniques on solving constraint systems, our decision procedure seems amenable to implementation, re-using existing tools (such as Avispa [6]).

Outline. Section 2 presents our formal model for routing protocols. It is illustrated with the modeling of the SRP protocol. We then give an alternative symbolic semantics, in Section 3, based on constraint systems and more amenable to automation. We show its correctness and completeness w.r.t. the concrete semantics. This result is of independent interest. In order to provide decision procedures for routing protocols, we first show in Section 4 how to transform the constraint systems corresponding to routing protocols into *solved* constraint systems. We then need to decide security properties such as route validity on solved constraint systems. We show in Section 5 that whenever there is an attack, there is a small one. Wrapping all the results together we provide our two decision procedures in Section 6. We provide applications of our results in Section 7. Some concluding remarks can be found in Section 8. The technical details of the proofs can be found in Appendix.

2. Model for protocols

2.1. Messages

Cryptographic primitives are represented by function symbols. More specifically, we consider a *signature* $(\mathcal{S}, \mathcal{F})$ made of a set of *sorts* $\mathcal{S}$ and a set of *function symbols* $\mathcal{F}$ together with arities of the form $ar(f) = s_1 \times \ldots \times s_k \to s$.

We consider an infinite set of *variables* $\mathcal{X}$ and an infinite set of *names* $\mathcal{N}$ that typically represent nonces or agent names. In particular, we consider a special sort loc for the nodes of the network. We assume that names and variables are given with sorts. We also assume an infinite subset $\mathcal{N}_{\mathsf{loc}}$ of names of sort loc. The set of *terms of sort s* is defined inductively by:

$$\begin{array}{rcll} t & ::= & & \text{term of sort } s \\ & | & x & \text{variable } x \text{ of sort } s \\ & | & a & \text{name } a \text{ of sort } s \\ & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F} \end{array}$$

where $ar(f) = s_1 \times \ldots \times s_k \to s$ and t_i is a term of some sort s_i.

We assume a special sort terms that subsumes all the other sorts and such that any term is of sort terms. We write $var(t)$ (resp. $names(t)$) for the set of variables (resp. names) occurring in a term t and $St(t)$ for the set of syntactic subterms of t. Sometimes, for sake of readability, we will write $var(t_1, t_2)$ (resp. $names(t_1, t_2)$ and $St(t_1, t_2)$) instead of $var(\{t_1, t_2\})$ (resp. $names(\{t_1, t_2\})$ and $St(\{t_1, t_2\})$) or $var(T)$ (resp. $names(T)$ and $St(T)$) when T is a set of terms. The term t is said to be a *ground* term if $var(t) = \emptyset$.

Example 1. *For example, we will consider the specific signature $(\mathcal{S}_1, \mathcal{F}_1)$ defined by $\mathcal{S}_1 = \{\mathsf{loc}, \mathsf{lists}, \mathsf{terms}\}$ and $\mathcal{F}_1 = \{\mathsf{hmac}, \langle\rangle, ::, [], \{\_\}_\_, \mathsf{priv}, \{\!|\_|\!\}_\_, [\![\_]\!]_\_\}$, with the following arities:*

- $\mathsf{hmac}, \langle\_,\_\rangle, \{\_\}_\_, \{\!|\_|\!\}_\_, [\![\_]\!]_\_ : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$*,*
- $:: : \mathsf{loc} \times \mathsf{lists} \to \mathsf{lists}$*,*
- $[] : \to \mathsf{lists}$*,*
- $\mathsf{priv} : \mathsf{terms} \to \mathsf{terms}$*.*

The sort lists represents lists of terms of sort loc. We assume that there is no name of sort lists. The symbol $::$ is the list constructor. $[]$ is a constant representing an empty list. The term $\mathsf{hmac}(m, k)$ represents the keyed hash message authentication code computed over message m with key k while $\langle\rangle$ is a pairing operator. The terms $\{m\}_k$ and $\{\!|m|\!\}_k$ represent respectively the message m encrypted with the symmetric (resp. asymmetric) key k. The term $[\![m]\!]_k$ represents the message m signed by the key k. The term $\mathsf{priv}(a)$ represents the private key of the agent a. For simplicity, we identify the agent names with their public keys. (Or conversely, we claim that agent identities are defined by their public keys). We write $\langle t_1, t_2, t_3\rangle$ for the term $\langle t_1, \langle t_2, t_3\rangle\rangle$, and $[t_1; t_2; t_3]$ for $t_1 :: (t_2 :: (t_3 :: []))$.

Substitutions are written $\sigma = \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$ with $dom(\sigma) = \{x_1, \ldots, x_n\}$. We only consider *well-sorted* substitutions, that is substitutions for which x_i and t_i have the same sort. The substitution σ is *ground* if all of the t_i are ground. We denote by $\mathsf{img}(\sigma)$ *the image of* σ, i.e., $\mathsf{img}(\sigma) = \{x\sigma \mid x \in dom(\sigma)\}$. The application of a substitution σ to a term t is written $\sigma(t)$ or $t\sigma$. A most general unifier of two terms t and u is a substitution denoted by $\mathsf{mgu}(t, u)$. We write $\mathsf{mgu}(t, u) = \perp$ when t and u are not unifiable.

The ability of the intruder is modeled by a deduction relation $\vdash \subseteq 2^{\mathsf{terms}} \times \mathsf{terms}$. The relation $T \vdash t$ represents the fact that the term t is computable from the set of terms T. The deduction relation can be arbitrary in our model and is thus left unspecified. It is typically defined through a deduction system like the one provided in Example 2.

Example 2. *Consider the term algebra* $(\mathcal{S}_1, \mathcal{F}_1)$ *defined in Example 1, the deduction system presented in Figure 1 reflects the ability for the intruder to compose messages by pairing, encrypting, and signing messages provided he has the corresponding keys. He can also compute a hmac when he knows the key and build lists. Conversely, he can retrieve components of a pair or a list. He can also decompose messages by decrypting provided he holds the decryption keys. For signatures, the intruder is also able to verify whether a signature* $[\![m]\!]_k$ *and a message* m *match (provided he has the verification key), but this does not produce any new message: this capability needs not to be represented in the deduction system. We also consider an optional rule*

$$\frac{T \vdash [\![u]\!]_v}{T \vdash u}$$

that expresses the ability to retrieve the whole message from its signature. This property may or may not hold depending on the signature scheme, and that is why this rule is optional. Some of our results will be based on this deduction system and will hold in both cases, whether or not this rule is considered in the deduction relation.

2.2. Process calculus

Several calculi already exist for modelling security protocols (e.g. [2, 1]). However, modeling ad-hoc routing protocols requires several additional features. For instance, a node of the network may store some information, e.g. the content of its routing table. We also need to take into account the network topology and to model broadcast communication. Such features can not be easily modeled in these calculi. Our calculus is inspired from CBS# [26], which allows mobile wireless networks and their security properties to be formally described and analyzed. However, we extend this calculus to allow nodes to perform some sanity checks on the routes they receive, such as neighborhood properties, as it is the case in the context of secured routing protocols.

The intended behavior of each node of the network can be modeled by a *process* defined by the grammar given in Figure 2. Our calculus is parameterized by a set $\mathcal{L}$ of formulas. The process $\mathsf{out}(u).P$ emits u and then behaves like P. The process $\mathsf{in}\ u[\Phi].P$ expects a message m of the form u such that Φ is true

$$\frac{T \vdash a \quad T \vdash l}{T \vdash a :: l} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v} \qquad \frac{T \vdash \{u\}_v \quad T \vdash v}{T \vdash u}$$

$$\frac{T \vdash a :: l}{T \vdash a} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash u} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \{\!|u|\!\}_v} \qquad \frac{T \vdash \{\!|u|\!\}_v \quad T \vdash \mathsf{priv}(v)}{T \vdash u}$$

$$\frac{T \vdash a :: l}{T \vdash l} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash v} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash [\![u]\!]_v} \qquad \frac{T \vdash [\![u]\!]_v}{T \vdash u}\ \textit{(optional)}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{hmac}(u, v)} \qquad \frac{u \in T \cup \{[]\}}{T \vdash u}$$

Figure 1: Deduction system associated to the signature $(\mathcal{S}_1, \mathcal{F}_1)$.

$P, Q :=$	processes
null	null process
$\mathsf{out}(u).P$	emission
$\mathsf{in}\ u[\Phi].P$	reception, $\Phi \in \mathcal{L}$
$\mathsf{store}(u).P$	storage
$\mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q$	reading
$\mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional, $\Phi \in \mathcal{L}$
$P \mid Q$	parallel composition
$!P$	replication
$\mathsf{new}\ m.P$	fresh name generation

Figure 2: Processes

and then behaves like $P\sigma$ where σ is such that $m = u\sigma$. If Φ is the true formula, we simply write $\mathsf{in}\ u.P$. The process $\mathsf{store}(u).P$ stores u in its storage list and then behaves like P. The process $\mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q$ looks for a message of the form u in its storage list and then, if such an element m is found, it behaves like $P\sigma$ where σ is such that $m = u\sigma$. If no element of the form u is found, then it behaves like Q. The process $P \mid Q$ runs P and Q in parallel. The process $!P$ executes P some arbitrary finite number of times. The restriction $\mathsf{new}\ m$ is used to model the creation in a process of new random numbers (e.g., nonces or key material). The process $\mathsf{new}\ m.P$ is the process that invents a new name m and continues as P. Sometimes, for the sake of clarity, we will omit the null process. We also omit the else part when $Q = \mathsf{null}$. We write $fv(P)$ for the set of free variables of P. A process P is *ground* when $fv(P) = \emptyset$.

The store and read primitives are particularly important when modeling routing protocols, in order to avoid multiple answers to a single request or to allow nodes to store and retrieve already known routes. These primitives can also be used to represent other classes of protocols, where a global state is assumed for each agent, in order to store some information (black list, already used keys. *etc.*) throughout the sessions.

Secured routing protocols typically require that nodes perform some checks on the messages they receive before accepting them. We will typically consider the logic $\mathcal{L}_{\mathsf{route}}$ defined by the grammar given in Figure 3. $\mathsf{check}(a, b)$ represents the fact that the two nodes a and b are neighbors. $\mathsf{checkl}(c, l)$ holds when l is a plausible route from the view of c, i.e., c occurs in the list (exactly once) and the previous and successive nodes in the list are neighbors of c. These checks typically allow a node to control that the route they are forwarding looks valid, from their point of view. It might help to detect when a malicious node has altered a route in a previous phase (e.g. request phase). The predicate route represents the validity of a route and will be used to express security properties. Lastly, $\mathsf{loop}(l)$ checks the existence of a loop in l.

$\Phi :=$	formula
$\mathsf{check}(a, b)$	adjacency tests
$\mathsf{checkl}(c, l)$	
$\mathsf{route}(l)$	validity of a route
$\mathsf{loop}(l)$	existence of a loop in a list
$\Phi_1 \wedge \Phi_2$	conjunction
$\Phi_1 \vee \Phi_2$	disjunction
$\neg\Phi$	negation

Figure 3: Logic $\mathcal{L}_{\mathsf{route}}$

Given an undirected graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$, the formal semantics $[\![\Phi]\!]_G$ of a formula $\Phi \in \mathcal{L}_{\mathsf{route}}$ is recursively defined as follows:

- $[\![\mathsf{check}(a, b)]\!]_G = \mathsf{true}$ iff $(a, b) \in E$.
- $[\![\mathsf{checkl}(c, l)]\!]_G = \mathsf{true}$ iff l is of sort lists, c appears exactly once in l, and for any l' sub-list of l,
  - if $l' = a :: c :: l_1$, then $(a, c) \in E$.
  - if $l' = c :: b :: l_1$, then $(c, b) \in E$.
- $[\![\mathsf{route}(l)]\!]_G = \mathsf{true}$ iff l is of sort lists, $l = [a_1; \ldots; a_n]$, for every $1 \leq i < n$, $(a_i, a_{i+1}) \in E$, and for every $1 \leq i, j \leq n$, $i \neq j$ implies that $a_i \neq a_j$.
- $[\![\mathsf{loop}(l)]\!]_G$ iff l is of sort lists and there exists an element appearing at least twice in l.
- $[\![\Phi_1 \wedge \Phi_2]\!]_G = [\![\Phi_1]\!]_G \wedge [\![\Phi_2]\!]_G$.
- $[\![\Phi_1 \vee \Phi_2]\!]_G = [\![\Phi_1]\!]_G \vee [\![\Phi_2]\!]_G$.
- $[\![\neg\Phi]\!]_G = \neg[\![\Phi]\!]_G$.

Our model is defined for any kind of logic $\mathcal{L}$, provided that the semantics $[\![\Phi]\!]_G$ of a closed formula Φ is defined, as soon as the underlying graph G is provided.

2.3. Example: modelling the SRP protocol

We consider the secured routing protocol SRP introduced in [27], assuming that each node already knows his neighbors (running e.g. some neighbor discovery protocol [31]). SRP is not a routing protocol by itself, it describes a generic way for securing source-routing protocols. We model here its application to the DSR protocol [21]. DSR is a protocol which is used when an agent S (the source) wants to communicate with another agent D (the destination), who is not his immediate neighbor. In an ad hoc network, messages can not always be sent directly to the destination, but sometimes have to travel along a path of nodes.

To discover a route to the destination, the source constructs a request packet and broadcasts this packet to its neighbors. The request packet contains its name S, the name of the destination D, an identifier of the request id, a list containing the beginning of a route to D, and a hmac computed over the content of the request with a key K_{SD} shared by S and D. The source then waits for an answer containing a route to D with a hmac matching this route, and checks that it is a plausible route by checking that the route does not contain a loop and that its neighbor in the route is indeed a real neighbor in the network.

In what follows, we consider the signature given in Example 1, and we use the following notations: x_S, x_D, and x_a are variables of sort loc; req, rep, and id are names; x_{id}, and x_m are variables of sort terms; and x_L, x_l, and x_r are variables of sort lists. We also use some variables as parameters: z_S, z_D, and z_V variables of sort loc, and $z_{K_{SD}}$ is a parameter used to store the key shared between z_S and z_D.

The process executed by a source node z_S initiating the search of a route towards a destination node z_D is $P_{\mathsf{init}}(z_S, z_D, z_{K_{SD}}) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].\mathsf{null}$ where:

$$\begin{aligned}
u_1 &= \langle \mathsf{req}, z_S, z_D, id, z_S :: [], \mathsf{hmac}(\langle \mathsf{req}, z_S, z_D, id\rangle, z_{K_{SD}})\rangle\\
u_2 &= \langle \mathsf{rep}, z_D, z_S, id, x_L, \mathsf{hmac}(\langle \mathsf{rep}, z_D, z_S, id, x_L\rangle, z_{K_{SD}})\rangle\\
\Phi_S &= \mathsf{checkl}(z_S, x_L) \wedge \neg \mathsf{loop}(x_L).
\end{aligned}$$

The names of the intermediate nodes are accumulated in the route request packet. Intermediate nodes relay the request over the network, except if they have already seen it, which is modeled by checking that the session id has not been already stored. An intermediate node also checks that the received request is locally correct by verifying whether the head of the list in the request is one of its neighbors. The process executed by an intermediate node z_V when forwarding a request is as follows:

$$P_{\mathsf{req}}(z_V) = \mathsf{in}\ w_1[\Phi_V].\mathsf{read}\ t\ \mathsf{then}\ \mathsf{null}\ \mathsf{else}\ (\mathsf{store}(t).\mathsf{out}(w_2))$$

where $\begin{cases} w_1 = \langle \mathsf{req}, x_S, x_D, x_{id}, x_a :: x_r, x_m \rangle \\ \Phi_V = \mathsf{check}(z_V, x_a) \\ t = \langle x_S, x_D, x_{id} \rangle \\ w_2 = \langle \mathsf{req}, x_S, x_D, x_{id}, z_V :: (x_a :: x_r), x_m \rangle \end{cases}$

When the request reaches the destination z_D, it checks that the request has a correct hmac and that the first node in the route is one of its neighbors. Then, the destination z_D constructs a route reply, in particular it computes a new hmac over the route accumulated in the request packet with $z_{K_{SD}}$, and sends the answer back over the network.The process executed by the destination node z_D is $P_{\mathsf{dest}}(z_D, z_S, z_{K_{SD}}) = \mathsf{in}\ v_1[\Phi_D].\mathsf{out}(v_2).\mathsf{null}$ where:

$$\begin{aligned} &v_1 = \langle \mathsf{req}, z_S, z_D, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{req}, z_S, z_D, x_{id} \rangle, z_{K_{SD}}) \rangle \\ &\Phi_D = \mathsf{check}(z_D, x_a) \\ &v_2 = \langle \mathsf{rep}, z_D, z_S, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{rep}, z_D, z_S, x_{id}, x_a :: x_l \rangle, z_{K_{SD}}) \rangle \end{aligned}$$

Then, the reply travels along the route back to z_S. The intermediate nodes check that the route in the reply packet is locally correct (i.e., they check that their name appears once in the list and that the nodes before and after them are their neighbors) before forwarding it. The process executed by an intermediate node z_V when forwarding a reply is the following:

$$P_{\mathsf{rep}}(z_V) = \mathsf{in}\ w'[\Phi'_V].\mathsf{out}(w').\mathsf{null}$$

where $\begin{cases} w' = \langle \mathsf{rep}, x_D, x_S, x_{id}, x_r, x_m \rangle \\ \Phi'_V = \mathsf{checkl}(z_V, x_r) \end{cases}$

2.4. Execution model

Each process is located at a specified node of the network. Unlike classical Dolev-Yao model, the intruder does not control the entire network but can only interact with his neighbors. More specifically, we assume that the topology of the network is represented by an undirected graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$, where an edge in the graph models the fact that two nodes are neighbors. We will only consider finite graphs, i.e., such that E is finite. We also assume that we have a set of nodes $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ that are controlled by the attacker. These nodes are then called *malicious*. Our model is not restricted to a single malicious node. Our results allow us to consider the case of several compromised nodes that collaborate by sharing their knowledge, using out-of-band resources or hidden channels (e.g. running other instances of the routing protocols). However, it is well-known that the presence of several colluding malicious nodes often yields straightforward attacks [19, 24].

A (ground) *concrete configuration* of the network is a triplet $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ where:

- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$ where null processes, i.e., expressions of the form $\lfloor \mathsf{null} \rfloor_n$ are removed. $\lfloor P \rfloor_n$ represents the (ground) process P located at node $n \in \mathcal{N}_{\mathsf{loc}}$. We will write $\lfloor P \rfloor_n \cup \mathcal{P}$ instead of $\{\lfloor P \rfloor_n\} \cup \mathcal{P}$.

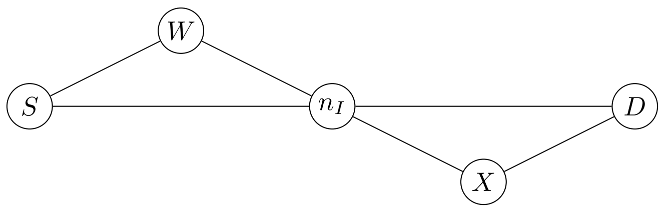
- $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in \mathcal{N}_{\mathsf{loc}}$ and t a ground term. $\lfloor t \rfloor_n$ represents the fact that the node n has stored the term t.
- $\mathcal{I}$ is a set of ground terms representing the messages seen by the intruder.

A *configuration with a hole* is a triplet of the form $(\lfloor P[\_] \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$ where $\mathcal{P}$, $\mathcal{S}$ and $\mathcal{I}$ are defined as above, and $P[\_]$ is a process with a hole, i.e., a process with a hole $\_$ instead of a process. This is useful for describing part (e.g. the beginning) of a process, while leaving the hole to represent the other part that will be filled in later.

Example 3. *Let S and D be names of sort loc, and K_{SD} be another name that intuitively represents the key shared between the nodes S and D. sort Continuing our modeling of SRP, a possible initial configuration for the SRP protocol is*

$$K_0 = (\lfloor P_{\mathsf{init}}(S, D, K_{SD}) \rfloor_S \mid \lfloor P_{\mathsf{dest}}(D, S, K_{SD}) \rfloor_D; \emptyset; \mathcal{I}_0)$$

where both the source node S and the destination node D wish to communicate. A more realistic configuration would include intermediate nodes but as shown in the following examples, this initial configuration is already sufficient to present an attack. We assume that each node has an empty storage list and that the initial knowledge of the intruder is given by $\mathcal{I}_0$. A possible network configuration is modeled by the graph $G_0 = (\mathcal{N}_{\mathsf{loc}}, E_0)$ below. We assume that there is a single malicious node, i.e., $\mathcal{M}_0 = \{n_I\}$. The nodes W and X are two extra (honest) nodes. We do not need to assume that the intermediate nodes W and X execute the routing protocol.



In routing protocols, each honest node broadcasts its messages to all its neighbors. To capture more malicious behaviors, we allow the nodes controlled by the intruder to send messages only to some specific neighbor. The communication system is formally defined by the rules of Figure 4. They are parameterized by the underlying graph G and the set of malicious nodes $\mathcal{M}$. A node expecting a message of the form u will accept any ground term t, provided that t is an instance of u, that is $t = u\sigma$ with $dom(\sigma) = var(u)$. We assume that each node that is sent a message that matches what it expects does indeed proceed the message. This is reflected in the three bullets of the COMM rule. We could model unreliable communications by removing these three conditions, yielding an actually simpler model.

COMM $(\{\lfloor \mathsf{in}\ u_j[\Phi_j].P_j \rfloor_{n_j} \mid \sigma_j \neq \bot, [\![\Phi_j\sigma_j]\!]_G = \mathsf{true}, (n, n_j) \in E\} \cup \lfloor \mathsf{out}(t).P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
$\rightarrow_{G,\mathcal{M}} \quad (\{\lfloor P_j\sigma_j \rfloor_{n_j}\} \cup \lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}')$
where σ_j is such that $t = u_j\sigma_j$, $\mathcal{I}' = \mathcal{I} \cup \{t\}$ if $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$ and $\mathcal{I}' = \mathcal{I}$ otherwise. Moreover, $\lfloor P' \rfloor_{n'} \in \mathcal{P}$ implies that:

- $(n, n') \notin E$, or
- P' is not of the form $\mathsf{in}\ u'[\Phi'].Q'$, or
- $P' = \mathsf{in}\ u'[\Phi'].Q'$ and
  - either there does not exist σ such that $t = u'\sigma$,
  - or $[\![\Phi'\sigma]\!]_G = \mathsf{false}$ where σ is such that $t = u'\sigma$ with $dom(\sigma) = var(u')$.

IN $(\lfloor \mathsf{in}\ u[\Phi].P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \quad \rightarrow_{G,\mathcal{M}} \quad (\lfloor P\sigma \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
if $(n_I, n) \in E$ for some $n_I \in \mathcal{M}$, $\mathcal{I} \vdash t$, $t = u\sigma$ and $[\![\Phi\sigma]\!]_G = \mathsf{true}$
for some substitution σ with $dom(\sigma) = var(u)$

STORE $(\lfloor \mathsf{store}(t).P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \quad \rightarrow_{G,\mathcal{M}} \quad (\lfloor P \rfloor_n \cup \mathcal{P}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I})$

READ-THEN $(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I})$
$\rightarrow_{G,\mathcal{M}} \quad (\lfloor P\sigma \rfloor_n \cup \mathcal{P}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I})$
where σ is a substitution with $dom(\sigma) = var(u)$ and such that $u\sigma = t$

READ-ELSE $(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
$\rightarrow_{G,\mathcal{M}} \quad (\lfloor Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
if for all t such that $\lfloor t \rfloor_n \in \mathcal{S}$, there does not exist any substitution such that $u\sigma = t$.

IF-THEN $(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
$\rightarrow_{G,\mathcal{M}} \quad (\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \qquad$ if $[\![\Phi]\!]_G = \mathsf{true}$

IF-ELSE $(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
$\rightarrow_{G,\mathcal{M}} \quad (\lfloor Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \qquad$ if $[\![\Phi]\!]_G = \mathsf{false}$

PAR $(\lfloor P_1 \mid P_2 \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \quad \rightarrow_{G,\mathcal{M}} \quad (\lfloor P_1 \rfloor_n \cup \lfloor P_2 \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$

REPL $(\lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \quad \rightarrow_{G,\mathcal{M}} \quad (\lfloor P\alpha \rfloor_n \cup \lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
where α is a renaming of the bound variables of P

NEW $(\lfloor \mathsf{new}\ m.P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}) \quad \rightarrow_{G,\mathcal{M}} \quad (\lfloor P\{^{m'}/_{m}\} \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I})$
where m' is a fresh name

Figure 4: Concrete transition system.

The relation $\rightarrow^*_{G,\mathcal{M}}$ is the reflexive and transitive closure of $\rightarrow_{G,\mathcal{M}}$. We may write $\rightarrow_{\mathcal{M}}$, $\rightarrow_G$, $\rightarrow$ instead of $\rightarrow_{G,\mathcal{M}}$ when the underlying network topology G or the underlying set $\mathcal{M}$ is clear from the context.

Note that in the case where we assume that there is a single malicious node and each honest node is connected to it (and not to any other node), we retrieve the model where the attacker is assumed to control all the communications.

Example 4. *Continuing Example 3, the following sequence of transitions is enabled from the initial configuration* K_0*:*

$$K_0 \rightarrow^*_{G_0,\mathcal{M}_0} (\lfloor \mathsf{in}\ u_2[\Phi_S].0 \rfloor_S \cup \lfloor P_{\mathsf{dest}}(D, S, K_{SD}) \rfloor_D ; \emptyset ; \mathcal{I}_0 \cup \{u_1\})$$

where u_1, u_2, Φ_S *are defined as follows:*

$$\begin{aligned} u_1 &= \langle \mathsf{req}, S, D, id, S :: [], \mathsf{hmac}(\langle \mathsf{req}, S, D, id \rangle, K_{SD}) \rangle \\ u_2 &= \langle \mathsf{rep}, D, S, id, x_L, \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, x_L \rangle, K_{SD}) \rangle \\ \Phi_S &= \mathsf{checkl}(S, x_L) \wedge \neg \mathsf{loop}(x_L) \end{aligned}$$

During this transition, S broadcasts to its neighbors a request in order to find a route to D. The intruder n_I is a neighbor of S in G_0, so he learns the request message. Assuming that the intruder knows the names of his neighbors, i.e., $W, X \in \mathcal{I}_0$, he can then build the following fake message request:

$$m = \langle \mathsf{req}, S, D, id, [X; W; S], \mathsf{hmac}(\langle \mathsf{req}, S, D, id \rangle, K_{SD}) \rangle$$

and send it to D. Since $(X, D) \in E_0$, the node D accepts this message and the resulting configuration is $(\lfloor \mathsf{in}\ u_2[\Phi_S].\mathsf{null} \rfloor_S \cup \lfloor \mathsf{out}(v_2\sigma).\mathsf{null} \rfloor_D ; \emptyset ; \mathcal{I}_0 \cup \{u_1\})$ *where*

$$\begin{cases} v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle D, S, x_{id}, x_a :: x_l \rangle, K_{SD}) \rangle \\ \sigma = \{{}^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_l}\} \end{cases}$$

As usual, an attack is defined as a reachability property.

Definition 1 ($\mathcal{M}$-attack). *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M}$ be a set of nodes. There is an* $\mathcal{M}$-attack *on a configuration with a hole $(\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ for the network topology G and the formula Φ if there exist $n, \mathcal{P}', \mathcal{S}', \mathcal{I}'$ such that:*

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})]; \mathcal{S}; \mathcal{I}) \ \rightarrow^*_{G,\mathcal{M}} \ (\lfloor \mathsf{out}(\mathsf{error}) \rfloor_n \cup \mathcal{P}', \mathcal{S}', \mathcal{I}')$$

where error *is a special symbol not occurring in the configuration $(\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$.*

The usual secrecy property can be typically encoded by adding a witness process in parallel. For example, the process $Q = \mathsf{in}\ s.\_$ can only evolve if it receives the secret s. Thus the secrecy preservation of s on a configuration $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ for a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ can be defined by the (non) existence of an $\{n_I\}$-attack on the configuration $(\mathcal{P} \cup \lfloor Q \rfloor_n; \mathcal{S}; \mathcal{I})$ and the formula true for the graph $G' = (\mathcal{N}_{\mathsf{loc}}, E \cup \{(n, n_I)\})$ where n is a name of sort loc that does not occur in $\mathcal{P}$.

Example 5. *For the SRP protocol, the property we want to check is that the list of nodes obtained by the source through the protocol represents a path in the graph. We can easily encode this property by replacing the null process in $P_{\mathsf{init}}(z_S, z_D, z_{K_{SD}})$ by a hole, and checking whether the formula $\neg\mathsf{route}(x_L)$ holds. Let $P'_{\mathsf{init}}(z_S, z_D, z_{K_{SD}})$ be the resulting process.*

$$P'_{\mathsf{init}}(z_S, z_D, z_{K_{SD}}) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].P$$

where $P = \mathsf{if}\ \neg\mathsf{route}(x_L)\ \mathsf{then}\ \mathsf{out}(\mathsf{error})$. Then, we recover the attack mentioned in [11] with the topology G_0 given in Example 3, and from the initial configuration:

$$K'_0 = (\lfloor P'_{\mathsf{init}}(S, D, K_{SD})\rfloor_S \mid \lfloor P_{\mathsf{dest}}(D, S, K_{SD})\rfloor_D; \emptyset; \mathcal{I}_0).$$

The attack scenario is the following. The source S sends a route request towards D. The request reaches the node n_I. Thus, the attacker receives the following message $\langle \mathsf{req}, S, D, id, S :: [], \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle$. The attacker then sends the following message to D in the name of X:

$$\langle \mathsf{req}, S, D, id, [X; W; S], \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle.$$

Since D is a neighbor of n_I, it will hear the transmission. In addition, since the list of nodes $[X; W; S]$ ends with X, which is also a neighbor of D, the destination D will process the request and will send the following route reply back to S: $\langle \mathsf{rep}, D, S, id, [X; W; S], \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, [X; W; S]\rangle, K_{SD})\rangle$. This reply will reach S through the malicious node n_I. More precisely, the attacker will send the reply to S in the name of W. Since W is a neighbor of S, the source will accept this reply which contains a false route.

In our framework, we have that:

$$\begin{aligned} K'_0 &\to^* (\lfloor \mathsf{in}\ u_2[\Phi_S].P\rfloor_S \cup \lfloor \mathsf{out}(m').\mathsf{null}\rfloor_D; \emptyset; \mathcal{I}) \\ &\to (\lfloor \mathsf{in}\ u_2[\Phi_S].P\rfloor_S \cup \lfloor \mathsf{null}\rfloor_D; \emptyset; \mathcal{I}') \\ &\to (\lfloor \mathsf{if} \neg\mathsf{route}([X; W; S])\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\rfloor_S; \emptyset; \mathcal{I}') \\ &\to (\lfloor \mathsf{out}(\mathsf{error}).\mathsf{null}\rfloor_S; \emptyset; \mathcal{I}') \end{aligned}$$

$$\textit{where} \left\{ \begin{array}{l} m' = \langle \mathsf{rep}, D, S, id, [X; W; S], \mathsf{hmac}(\langle D, S, id, [X; W; S]\rangle, K_{SD})\rangle \\ \mathcal{I} = \mathcal{I}_0 \cup \{u_1\}, \textit{and} \\ \mathcal{I}' = \mathcal{I}_0 \cup \{u_1\} \cup \{m'\}. \end{array} \right.$$

2.5. Contributions

We are now ready to state our two main decidability results.

Simple properties like secrecy are undecidable when considering an unbounded number of role executions, even for classical protocols [15]. Since our class of processes encompasses classical protocols, the existence of an attack is also undecidable. In what follows, we thus consider a finite number of sessions, that is processes without replication. In most existing frameworks, the intruder is given as initial knowledge a finite number of messages (e.g. some of the secret keys or messages learned in previous executions). However, in the context of routing protocols, it is important to give an a priori unbounded number of node names to the attacker that he can use at will, in particular for possibly passing some disequality constraints and for creating false routes.

We say that a process is *finite* if it does not contain the replication operator. A concrete configuration $K = (\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ is said *initial* if K is ground, $\mathcal{P}$ is finite, $\mathcal{S}$ is a finite set of terms and $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}'$ where $\mathcal{I}'$ is a finite set of terms (the intruder is given all the node names in addition to its usual initial knowledge).

Note that unlike many approaches for standard protocols, fixing in advance the number of names an adversary may use would limit its power. There are two main reasons. First, the nodes may be checking that the route they are carrying does not contain a loop, which means that all nodes of the route are distinct. Due to routing protocol, an attack may require a route of a certain length. Thus the adversary may need some fresh node names just to ensure that all names are different. Second, our framework allows both disequality tests and typed variables. Distinct nodes names may be needed to pass some disequality tests. Since variables are typed, it is not always possible to simulate distinct names by simply pairing (concatenating) a single one.

We show that accessibility properties are decidable for finite processes of our process algebra, which models secured routing protocols, for a bounded number of sessions but an unbounded number of node names. Of course, any attack will use just finitely many names but the exact number of names is not fixed in advance. We actually provide two decision procedures, according to whether the network is *a priori* given or not. In the case where the network topology is not fixed in advance, our procedure enables us to automatically decide whether there exists a (worst-case) topology that would yield an attack.

Theorem 1. *Let $K = (\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ be an initial concrete configuration with a hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ such that there is an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Theorem 2. *Let $K = (\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ be an initial concrete configuration with a hole, $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a finite graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

NP-hardness is an immediate consequence of NP-hardness of secrecy [32] for standard security protocols, which can be easily encoded in our framework, considering a graph where a malicious node is connected to all the other nodes. Our contribution therefore consists in providing two NP algorithms. We provide here a very general sketch of the proof, describing the remaining sections of the paper.

- We first provide an alternative symbolic semantics (Section 3) based on constraint systems and more amenable to automation. We show its correctness and completeness w.r.t. the concrete semantics.
- We show how to transform the constraint systems corresponding to routing protocols into *solved* constraint systems (Section 4) .
- We show how to bound the size of a minimal attack on a solved constraint system (Section 5).
- We can then conclude to decidability, providing an NPTIME complexity bound (Section 6).

We provide applications of our results in Section 7.

3. Symbolic semantics

It is difficult to directly reason with the transition system defined in Figure 4 since it is infinitely branching: a potentially infinite number of distinct messages can be sent at each step by the intruder node. Indeed, the intruder has at his disposal an infinite number of names, and he can also build large terms using e.g., lists and encryptions. That is why it is often interesting to introduce a *symbolic* transition system where each intruder step is captured by a single rule (e.g., [5]).

3.1. Constraint systems

As in [25, 14, 32], groups of executions can be represented using constraint systems. However, compared to previous work, we have to enrich constraint systems in order to cope with the formulas that are checked upon the reception of a message and also in order to cope with generalized disequality tests for reflecting cases where agents reject messages of the wrong form. Indeed, since messages can be broadcast to all neighbors, we need to determine for each message which agents will accept the message and which agents will not accept it.

Definition 2 (constraint system). *A* constraint system $\mathcal{C}$ *is a finite conjunction of* constraints *of the form $v = u$ (unification constraint), $\mathcal{I} \Vdash u$ (deduction constraint), $\forall X.\, v \neq u$ (disequality constraint), and Φ (formula of $\mathcal{L}_{\mathsf{route}}$), where v, u are terms, $\mathcal{I}$ is a set of terms that contains at least a term of each sort, and X is a set of variables. Moreover, we assume that the constraints in $\mathcal{C}$ can be ordered $C_1, \dots, C_n$ in such a way that the following properties hold:*

- *(monotonicity) If $C_i = (\mathcal{I}_i \Vdash u_i)$ and $C_j = (\mathcal{I}_j \Vdash u_j)$ with $i < j$ then $\mathcal{I}_i \subseteq \mathcal{I}_j$;*
- *(origination) If $C_i = (\mathcal{I}_i \Vdash u_i)$ (resp. $C_i = (v_i = u_i)$) then for all $x \in var(\mathcal{I}_i)$ (resp. $x \in var(v_i)$), there exists $j < i$ such that*

– *either* $C_j = (\mathcal{I}_j \Vdash u_j)$ *with* $x \in var(u_j)$*;*

– *or* $C_j = (v_j = u_j)$ *with* $x \in var(u_j)$*.*

Lastly, we assume that $var(\mathcal{C}) \subseteq rvar(\mathcal{C})$ *where* $rvar(\mathcal{C})$ *represents the set of variables introduced in* $\mathcal{C}$ *in the right-hand-side of a unification constraint or a deduction constraint.*

The origination property ensures that variables are always introduced by a unification constraint or a deduction constraint, which is always the case when modeling protocols.

Note that our disequality constraints are rather general since they do not simply allow one to check that two terms are different ($u \neq v$), but they also allow to ensure that no unification was possible at a certain point of the execution, which is a necessary check due to our broadcast primitive.

A *solution* to a constraint system $\mathcal{C}$ for a graph G is a ground substitution θ such that $dom(\theta) = rvar(\mathcal{C})$ and:

- $v\theta = u\theta$ for all $v = u \in \mathcal{C}$;
- $\mathcal{I}\theta \vdash u\theta$ for all $I \Vdash u \in \mathcal{C}$;
- for all $(\forall X. v \neq u) \in \mathcal{C}$, the terms $v\theta$ and $u\theta$ are not unifiable (even renaming the variables of X with fresh variables); and
- $[\![\Phi\theta]\!]_G = \mathsf{true}$ for every formula $\Phi \in \mathcal{C}$.

Example 6. *Consider the following set of constraints:*

$$\mathcal{C} = \left\{ \begin{array}{l} \mathcal{I}_0 \cup \{u_1\} \Vdash v_1 \ \wedge \ \Phi_D \ \wedge \\ \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2 \ \wedge \ \Phi_S \ \wedge \ \neg\mathsf{route}(x_L) \end{array} \right\}$$

with:

$$\begin{array}{l} u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, \mathsf{hmac}(\langle \mathsf{req}, S, D, id \rangle, K_{SD}) \rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_L, \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, x_L \rangle, K_{SD}) \rangle \\ \Phi_D = \mathsf{check}(D, x_a) \\ \Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg\mathsf{loop}(x_L) \\ v_1 = \langle \mathsf{req}, S, D, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{req}, S, D, x_{id} \rangle, K_{SD}) \rangle \\ v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l \rangle, K_{SD}) \rangle \end{array}$$

We have that $\mathcal{C}$ *is a constraint system, and assuming that* $X, W \in \mathcal{I}_0$*, we have that the substitution*

$$\theta = \{{}^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_l}, {}^{[X;W;S]}/_{x_L}\}$$

is a solution of the constraint system $\mathcal{C}$ *for graph* G_0 *defined in Example 2.3.*

3.2. Transition system

Concrete executions can be finitely represented by executing the transitions *symbolically.* A *symbolic configuration* is a quadruplet $(\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C})$ where

- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$ where null processes are removed. $\lfloor P \rfloor_n$ represents the process P located at node $n \in \mathcal{N}_{\mathsf{loc}}$;
- $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in \mathcal{N}_{\mathsf{loc}}$ and t a term (not necessarily ground).
- $\mathcal{I}$ is a set of terms (not necessarily ground) representing the messages seen by the intruder.
- $\mathcal{C}$ is a constraint system such that $T \subseteq \mathcal{I}$ for every constraint $T \Vdash u \in \mathcal{C}$.

Such a configuration is *ground* when: $fv(\mathcal{P}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C})$.

Compared to concrete configurations, terms exchanged by processes in symbolic configurations are not necessarily ground anymore but have to satisfy some (unification, deduction or disequality) constraints. We defined the associated symbolic transitions in Figure 5. They mimic concrete ones. In particular, for the communication rule, the set I of processes ready to input a message is split into three sets: the set J of processes that accept the message t, the set K of processes that reject the message t because t does not unify with the expected pattern u_k, and the set L that reject the message t because the condition Φ_l is not fulfilled. Considering a calculus with unreliable communication would have some consequences on the symbolic transition system. In particular, it would simplify the $\textsc{Comm}_s$ rule. Indeed, only the constraints corresponding to the set J of processes (processes that accept the message) would need to be added in $\mathcal{C}$. Processes that do not accept a message do not have to satisfy any additional condition. We believe that our NP decision procedures would also work in this simplified setting.

Whenever $(\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\mathcal{P}';\mathcal{S}';\mathcal{I}';\mathcal{C}')$ where $(\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C})$ is a (ground) symbolic configuration then $(\mathcal{P}';\mathcal{S}';\mathcal{I}';\mathcal{C}')$ is still a (ground) symbolic configuration. This invariant will often be useful in proofs. This is formally stated below and proved in Appendix A.

Lemma 1. *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$, and $K_s = (\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C})$ be a ground symbolic configuration. If K_s' is such that $K_s \rightarrow^s_{G,\mathcal{M}} K_s'$, then K_s' is a ground symbolic configuration.*

Example 7. *Executing the same transitions as in Example 5 symbolically, we reach the following configuration:*

$$K_s = (\lfloor \mathsf{out}(\mathsf{error}).0 \rfloor_S; \emptyset; \mathcal{I}_0 \cup \{u_1, v_2\}; \mathcal{C})$$

where $\mathcal{C}, u_1, v_2$ are defined as in Example 6.

Comm$_s$ $(\lfloor \mathsf{out}(t).P \rfloor_n \cup \{\lfloor \mathsf{in}\ u_i[\Phi_i].P'_i \rfloor_{n_i} \mid i \in I\} \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$

$\rightarrow^s_{G,\mathcal{M}}\ (\lfloor P \rfloor_n \cup \{\lfloor \mathsf{in}\ u_k[\Phi_k].P'_k \rfloor_{n_k} \mid k \in K \cup L\}$

$\cup \{\lfloor P'_j \rfloor_{n_j} \mid j \in J\} \cup \mathcal{P}; \mathcal{S}; \mathcal{I}'; \mathcal{C}')$

where:

- $\lfloor P' \rfloor_{n'} \in \mathcal{P}$ implies that $(n, n') \notin E$ or P' is not of the form $\mathsf{in}\ u'[\Phi'].Q'$,
- $I = J \uplus K \uplus L$ and $(n_i, n) \in E$ for every $i \in I$,
  I indexes the set of processes that are ready to input a messages. I is split into three distinct subsets: J for processes that will accept the message, K for processes that will reject it because it does match the expected form u_i, and L for processes that will reject it because the test fails.
- $\mathcal{C}' = \mathcal{C} \wedge \{t = u_j \wedge \Phi_j \mid j \in J\} \wedge \{\forall (var(u_k) \smallsetminus rvar(\mathcal{C})) \,.\, t \neq u_k \mid k \in K\} \wedge \{t = u_l\alpha_l \wedge \neg\Phi_l\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u_l) \smallsetminus rvar(\mathcal{C})$ by fresh variables,
- $\mathcal{I}' = \mathcal{I} \cup \{t\}$ when $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$, and $\mathcal{I}' = \mathcal{I}$ otherwise.

In$_s$ $(\lfloor \mathsf{in}\ u[\Phi].P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \mathcal{I} \Vdash u \wedge \Phi)$
if $(n_I, n) \in E$ for some $n_I \in \mathcal{M}$

Store$_s$ $(\lfloor \mathsf{store}(t).P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{P}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I}; \mathcal{C})$

Read-Then$_s$ $(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$

$\rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge t = u)$
where $\lfloor t \rfloor_n \in \mathcal{S}$

Read-Else$_s$ $(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$

$\rightarrow^s_{G,\mathcal{M}} (\lfloor Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \{\forall X \,.\, t \neq u \mid \lfloor t \rfloor_n \in \mathcal{S}\})$
where $X = var(u) \smallsetminus rvar(\mathcal{C})$

If-Then$_s$ $(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \Phi)$

If-Else$_s$ $(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \neg\Phi)$

Par$_s$ $(\lfloor P_1 \mid P_2 \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_1 \rfloor_n \cup \lfloor P_2 \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$

Repl$_s$ $(\lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\alpha \rfloor_n \cup \lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$
where α is a renaming of the bound variables of P that are not in $rvar(\mathcal{C})$.

New$_s$ $(\lfloor \mathsf{new}\ m.P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\{^{m'}/_m\} \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$
where m' is a fresh name

Figure 5: Symbolic transition system.

$$\begin{tikzcd}
K_s \arrow[r, dashed, "s", "G,\mathcal{M}"'] \arrow[d, bend right, "\theta"'] & K'_s \arrow[d, dashed, bend left, "\theta'"] \\
K_c \arrow[r, "G,\mathcal{M}"'] & K'_c
\end{tikzcd}
\qquad
\begin{tikzcd}
K_s \arrow[r, "s", "G,\mathcal{M}"'] \arrow[d, dashed, bend right, "\theta"'] & K'_s \arrow[d, bend left, "\theta'"] \\
K_c \arrow[r, dashed, "G,\mathcal{M}"'] & K'_c
\end{tikzcd}$$

Figure 6: Diagrams illustrating Proposition 1 (left) and Proposition 2 (right).

3.3. Soundness and completeness

We show that our symbolic transition system reflects exactly the concrete transition system, i.e., each concrete execution of a process is captured by one of the symbolic executions (see Figure 6). More precisely, a concrete configuration is represented by a symbolic configuration if it is one of its instances, called *concretization*.

Definition 3 (θ-concretization). *A concretization of a symbolic configuration $K_s = (\mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ is a concrete configuration $K_c = (\mathcal{P}; \mathcal{S}; \mathcal{I})$ such that there exists a solution θ of $\mathcal{C}$ and, furthermore, $\mathcal{P}_s\theta = \mathcal{P}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$. We say that K_c is a θ-concretization of K_s.*

Note that the θ-concretization of a ground symbolic configuration is a ground concrete configuration. Now, we show that each concrete transition can be matched by a symbolic one.

Proposition 1 (completeness). *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ be a ground symbolic configuration and θ be a solution of $\mathcal{C}$. Let K_c be the θ-concretization of K_s. Let K'_c be a concrete configuration such that $K_c \to_{G,\mathcal{M}} K'_c$. Then there exists a ground symbolic configuration K'_s and a substitution θ' such that:*

- *K'_c is the θ'-concretization of K'_s, and*
- *$K_s \to^s_{G,\mathcal{M}} K'_s$.*

The proof is performed by studying each rule of the concrete transition system, showing that the corresponding symbolic rule covers all possible cases. In particular, disequality constraints allow to faithfully model cases where nodes reject a message because the message does not match the expected pattern. Conversely, we have that each symbolic transition can be instantiated in a concrete one.

Proposition 2 (soundness). *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ and $K_s' = (\mathcal{P}_s'; \mathcal{S}_s'; \mathcal{I}_s'; \mathcal{C}')$ be two ground symbolic configurations such that $K_s \rightarrow^s_{G,\mathcal{M}} K_s'$. Let θ' be a solution of $\mathcal{C}'$ and θ be the restriction of θ' to $rvar(\mathcal{C})$. Let K_c be the θ-concretization of K_s. There exists a ground concrete configuration K_c' such that:*

- *$K_c \rightarrow_{G,\mathcal{M}} K_c'$, and*
- *K_c' is the θ'-concretization of K_s'.*

The proof is again obtained by inspection of the rules. We deduce from these two propositions that checking for a concrete attack can be reduced to checking for a symbolic one.

Theorem 3. *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be a ground concrete configuration with a hole, and Φ be a formula. There is an $\mathcal{M}$-attack on K and Φ for graph G if, and only if,*

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})]; \mathcal{S}; \mathcal{I}; \emptyset) \ \rightarrow^{s*}_{G,\mathcal{M}} \ (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

and the constraint system $\mathcal{C} \wedge u = \mathsf{error}$ has a solution.

Proof. We show the two directions separately.
($\Rightarrow$) First, let us suppose that there is an attack on K and Φ for graph G. By definition of an attack, there exists a concrete configuration K' such that:

- K' is of the form $(\lfloor \mathsf{out}(\mathsf{error}) \rfloor_n \cup \mathcal{P}'; \mathcal{S}'; \mathcal{I}')$, and
- $K \rightarrow^*_G K'$.

By applying Proposition 1 recursively, we deduce that there exists a ground symbolic configuration K_s' and a substitution θ' such that:

- $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}; \emptyset) \rightarrow^{s*}_G K_s'$, and
- K' is the θ'-concretization of K_s.

Consequently, K_s' is of the form $(\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s'; \mathcal{S}_s'; \mathcal{I}_s'; \mathcal{C}')$, θ' is a solution of $\mathcal{C}'$, and $u\theta' = \mathsf{error}$. Hence we have that θ' is a solution of $\mathcal{C}' \wedge u = \mathsf{error}$.

($\Leftarrow$) Conversely, assume that

$$K_s = (\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}; \emptyset) \ \rightarrow^{s*}_G \ (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) = K_s'$$

and the constraint system $\mathcal{C} \wedge u = \mathsf{error}$ has a solution. Let θ' be a solution of $\mathcal{C} \wedge u = \mathsf{error}$. Note that, since K_s' is a ground symbolic configuration (thanks to Lemma 1), we have that $var(u) \subseteq rvar(\mathcal{C})$. Hence, we have that θ' is a solution of $\mathcal{C}$ and $u\theta' = \mathsf{error}$.

First note that K_s is a ground symbolic configuration whose concretization is $K = (\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I})$. Thanks to Lemma 1, we know that the symbolic configurations involved in this derivation are ground. Hence, by applying recursively Proposition 2, we know that there exists a ground concrete configuration K' such that:

- $K \rightarrow_G^* K'$, and
- K' is the θ'-concretization of K'_s.

Moreover, since $u\theta' = \mathsf{error}$, we easily deduce that $K' = (\lfloor \mathsf{out}(\mathsf{error}) \rfloor_n \cup \mathcal{P}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$. Hence, there is an attack on K and Φ for graph G. $\square$

Note that our result holds for any signature, for any choice of predicates, and for processes possibly with replication. Of course, it then remains to decide the existence of a constraint system that has a solution.

Example 8. *Consider our former example of an attack on* SRP, *with initial configuration* K_0*. We can reach the configuration* K_s*, and the constraint system* $\mathcal{C}$ *has a solution* σ *for graph* G_0 *(cf. Example 6), so there is an* $\{n_I\}$*-attack on* K_0 *for* G_0*.*

In the remaining of the paper, our goal is to show how to solve the resulting constraint systems. We restrict our attention to the fixed signature $(\mathcal{S}_1, \mathcal{F}_1)$ (defined in Example 1) and the logic $\mathcal{L}_{\mathsf{route}}$ (defined in Section 2.2). We also assume the associated deduction system $\vdash$ defined in Figure 1.

4. Turning constraint systems into solved forms

It has been shown in [14] that the existence of a solution of a constraint system (with only deduction constraints) can be reduced to the existence of a solution of a *solved* constraint system, where right-hand-sides of the constraints are just variables. However, the result of [14] assumes that the deduction constraints have finite left-hand sides and does not allow one to deal with equality and disequality tests and formulas of $\mathcal{L}_{\mathsf{route}}$.

In this section, we show how the existence of a solution for a constraint system (without unification constraint) can be reduced to the existence of a solution for a solved deduction constraint system together with disequality constraints and formulas, for our extended signature. We do not consider unification constraints since we will see that we can easily get rid of them by first computing a mgu.

4.1. Simplification rules

We say that a constraint system $\mathcal{C}$ is a *deduction constraint system* if all of its constraints are deduction constraints. Such a system is in *solved form* if $\mathcal{C} = T_1 \Vdash x_1 \wedge \ldots \wedge T_n \Vdash x_n$ where $x_1, \ldots, x_n$ are distinct variables. It has been shown in [14] that the existence of a solution for a deduction constraint system with finite left-hand sides can be reduced to the existence of a solution for a *solved* deduction constraint system by applying the transformation rules presented in Figure 7.

$$\mathsf{R}_1 \qquad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow \quad \mathcal{C} \qquad \text{if } T \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_2 \qquad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathsf{mgu}(t, v), t \in St(T), v \in St(u), t \neq v, t, v \text{ not variables}$$

$$\mathsf{R}_3 \qquad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathsf{mgu}(t_1, t_2), t_1, t_2 \in St(T), t_1 \neq t_2, t_1, t_2 \text{ not variables}$$

$$\mathsf{R}'_3 \qquad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathsf{mgu}(t_2, t_3), \{\!|t_1|\!\}_{t_2}, \mathsf{priv}(t_3) \in St(T), t_2 \neq t_3, t_2 \text{ or } t_3 \text{ (or both) is a variable}$$

$$\mathsf{R}_4 \qquad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow \quad \bot \qquad \text{if } \mathit{var}(T \cup \{u\}) = \emptyset \text{ and } T \not\vdash u$$

$$\mathsf{R}_\mathsf{f} \qquad \mathcal{C} \wedge T \Vdash \mathsf{f}(u, v) \quad \rightsquigarrow \quad \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v \qquad \text{for } \mathsf{f} \in \{\langle\rangle, ::, \mathsf{hmac}, \{\_\}_\_, \{\!|\_|\!\}_\_, [\![\_]\!]_\_\}$$

Figure 7: Simplification rules

The only difference between the rules presented in Figure 7 and those proposed in [14] is in the rule R_f. We adapt this rule in order to deal with hmac and lists. All the rules are indexed by a substitution. When there is no index then the identity substitution is implicitly assumed. We write $\mathcal{C} \rightsquigarrow^n_\sigma \mathcal{C}'$ if there are $\mathcal{C}_1, \ldots, \mathcal{C}_n$ with $n \geq 1$, $\mathcal{C}' = \mathcal{C}_n$, $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$, and $\sigma = \sigma_n \circ \sigma_{n-1} \circ \cdots \circ \sigma_1$. We write $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ if $\mathcal{C} \rightsquigarrow^n_\sigma \mathcal{C}'$ for some $n \geq 1$, or if $\mathcal{C}' = \mathcal{C}$ and σ is the identity substitution.

However, the result of [14] assumes that the deduction constraints are of the form $T \Vdash u$ where T is a *finite* set of terms. We have extended this result to the case where T contains an infinite set of names. For this, we introduce the notion of a special constraint system $(\mathcal{C}, \mathcal{I}_0)$ where $\mathcal{I}_0$ is an infinite set of names. Moreover, we assume that those names do not occur in $\mathcal{C}$, i.e., $St(\mathcal{C}) \cap \mathcal{I}_0 = \emptyset$ where $St(\mathcal{C})$ represents the set of subterms that occur in $\mathcal{C}$.

Definition 4 (special constraint system). *Let $\mathcal{C} = T_0 \Vdash u_0 \wedge \ldots \wedge T_n \Vdash u_n$ be a deduction constraint system where all left-hand sides of constraints are finite, and $\mathcal{I}_0$ be an infinite set of names such that $\mathcal{N}_{\mathsf{loc}} \subseteq T_0 \cup \mathcal{I}_0$. We say that*

$(\mathcal{C}, \mathcal{I}_0)$ *is a* special constraint system *if* $St(\mathcal{C}) \cap \mathcal{I}_0 = \emptyset$. *The deduction constraint system* $\overline{\mathcal{C}}^{\mathcal{I}_0}$ *associated to* $(\mathcal{C}, \mathcal{I}_0)$ *is inductively defined by*

$$\overline{\mathcal{C} \wedge T \Vdash u}^{\mathcal{I}_0} = \overline{\mathcal{C}}^{\mathcal{I}_0} \wedge ((\mathcal{I}_0 \cup T) \Vdash u).$$

A substitution θ *is a* solution of a special constraint system $(\mathcal{C}, \mathcal{I}_0)$ *if for every* $T \Vdash u \in \mathcal{C}$, $(T \cup \mathcal{I}_0)\theta \Vdash u\theta$, *i.e.,* θ *is a solution of* $\overline{\mathcal{C}}^{\mathcal{I}_0}$.

Example 9. *We retrieve the following set of deduction constraints from Example 6:*

$$\mathcal{C}'' = \mathcal{I}_0 \cup \{u_1\} \Vdash v_1 \ \wedge \ \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2$$

The set of terms $\mathcal{I}_0$ *represents the initial knowledge of the attacker and we assume that* $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}'_0$ *where* $\mathcal{I}'_0$ *is a finite set of terms. This assumption is reasonable if we consider that the names in* $\mathcal{N}_{\mathsf{loc}}$ *represent IP addresses, and that the intruder can easily create any IP address.*

Let $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \smallsetminus names(\mathcal{I}'_0, S, D)$, *and* $T_0 = \mathcal{I}'_0 \cup (\mathcal{N}_{\mathsf{loc}} \cap names(\mathcal{I}'_0, S, D))$. *We have that* $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}'_0 = T_0 \cup \mathcal{I}$. *Moreover,* T_0 *is a finite set of terms. We consider the following set of deduction constraints:*

$$\mathcal{C}_1 = T_0 \cup \{u_1\} \Vdash v_1 \ \wedge \ T_0 \cup \{u_1, v_2\} \Vdash u_2$$

We have that $\mathcal{C}_1$ *is a deduction constraint system where all left-hand sides of constraints are finite. Furthermore,* $\mathcal{N}_{\mathsf{loc}} \subseteq \mathcal{I} \cup T_0$. *As* $St(\mathcal{C}_1) \cap \mathcal{I} = \emptyset$, *we have that* $(\mathcal{C}_1, \mathcal{I})$ *is a special constraint system. Note also that* $\overline{\mathcal{C}_1}^{\mathcal{I}} = \mathcal{C}''$.

We show that solving a special constraint system $(\mathcal{C}, \mathcal{I}_0)$ can be done by applying our simplification rules to $\mathcal{C}$.

Proposition 3. *Let* $(\mathcal{C}, \mathcal{I})$ *be a special constraint system.*

1. *If* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ *then* $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \overline{\mathcal{C}'}^{\mathcal{I}}$ *by applying the same simplification rule, and* $(\mathcal{C}', \mathcal{I})$ *is a special constraint system.*
2. *If* $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \mathcal{C}'_s$, *then there exists* $\mathcal{C}'$ *such that* $\mathcal{C}'_s = \overline{\mathcal{C}'}^{\mathcal{I}}$ *and* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ *by applying the same simplification rule. Furthermore, we have that* $(\mathcal{C}', \mathcal{I})$ *is a special constraint system.*

The proof of this proposition relies on the following lemma, which intuitively states that adding an infinite set of disjoint names does not provide additional deduction power to the intruder.

Lemma 2. *Let* T *be a set of terms that contains at least one term of each sort,* u_0 *be a term, and* $\mathcal{I}$ *be a set of names* $St(T \cup \{u_0\}) \cap \mathcal{I} = \emptyset$. *If* $T \cup \mathcal{I} \vdash u_0$, *then we have that* $T \vdash u_0$.

As shown in [14], the transformation rules terminate but the length of a derivation might be exponential. Getting a polynomial bound on the length of simplification sequences can be achieved by considering a (complete) strategy in order to avoid getting twice the same constraint. We consider the following strategy $\mathcal{S}$:

- apply R_4 eagerly;
- apply the substitution rules, namely R_2, R_3, and R'_3, up to a point, then stop applying them at all.

Assuming that all the constraints are uncolored at the beginning.

- Consider the uncolored constraint with the largest right-hand side. Either color it or apply R_f to it.
- When the system is entirely colored, apply R_1.

For deduction constraint systems with finite left-hand sides, the strategy $\mathcal{S}$ is complete and yields derivations of polynomial length (see Section 4.7 in [14]). It remains to show that the procedure also works for special constraint systems.

4.2. Soundness, completeness, and termination

The proof of Theorem 4 is mainly an adaptation of the result in [14] and relies on Proposition 3. This theorem shows that when solving a special constraint system $(\mathcal{C}, \mathcal{I}_0)$, it is sufficient to apply the transformation rules to $\mathcal{C}$.

Theorem 4. *Let $(\mathcal{C}_0, \mathcal{I}_0)$ be a special constraint system, and Φ be a set of formulas and disequality constraints,*

1. *(Soundness) If $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$ for some $\mathcal{C}'$ and some substitution σ, and if θ is a solution for $\Phi\sigma$ and $(\mathcal{C}', \mathcal{I}_0)$, then $\theta \circ \sigma$ is a solution for Φ and $(\mathcal{C}_0, \mathcal{I}_0)$.*
2. *(Completeness) If θ is a solution for $(\mathcal{C}_0, \mathcal{I}_0)$ and Φ, then there exists a deduction constraint system $\mathcal{C}'$ in solved form and substitutions σ, θ' such that $\theta = \theta' \circ \sigma$, $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$, and θ' is a solution for $(\mathcal{C}', \mathcal{I}_0)$ and $\Phi\sigma$.*
3. *(Termination/Complexity) If $\mathcal{C}_0 \rightsquigarrow^n_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$ for some deduction constraint system $\mathcal{C}'$ and some substitution σ, then n is polynomially bounded in the size of $\mathcal{C}_0$. Moreover, we have that $St(\mathcal{C}') \subseteq St(\mathcal{C}_0\sigma) \subseteq St(\mathcal{C}_0)\sigma$.*

Proof. We show the three items separately.

Soundness. If $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$, then by applying Proposition 3 inductively, we get that $\overline{\mathcal{C}_0}^{\mathcal{I}_0} \rightsquigarrow^*_\sigma \overline{\mathcal{C}'}^{\mathcal{I}_0}$. By hypothesis, we have that θ is a solution of $\Phi\sigma$ and $(\mathcal{C}', \mathcal{I}_0)$. In particular, we have that θ is a solution of $\overline{\mathcal{C}'}^{\mathcal{I}_0}$. Thanks to Theorem 4.3 in [14] (straightforwardly extended to constraint systems with infinite number of names in the left-hand sides), we deduce that $\theta \circ \sigma$ is a solution of $\overline{\mathcal{C}_0}^{\mathcal{I}_0}$. Hence, we have that $\theta \circ \sigma$ is a solution for Φ and $(\mathcal{C}_0, \mathcal{I}_0)$.

Completeness. Let θ be a solution for $(\mathcal{C}_0, \mathcal{I}_0)$ and Φ. In particular, we have that θ is a solution of $\overline{\mathcal{C}_0}^{\mathcal{I}_0}$ and Φ. By applying Theorem 4.3 in [14], there exists a constraint system $\mathcal{C}_s$ in solved form and substitutions σ, θ' such that $\theta = \theta' \circ \sigma$, $\overline{\mathcal{C}_0}^{\mathcal{I}_0} \rightsquigarrow^*_\sigma \mathcal{C}'_s$ by a derivation in $\mathcal{S}$, and θ' is a solution for $\mathcal{C}'_s$ and $\Phi\sigma$. By applying Proposition 3 recursively, we get that there exists $\mathcal{C}'$ such that $\mathcal{C}'_s = \overline{\mathcal{C}'}^{\mathcal{I}_0}$, and $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$. Hence, we have that θ' is a solution of $(\mathcal{C}', \mathcal{I}_0)$. Furthermore, $\mathcal{C}'$ is in solved form, since $\overline{\mathcal{C}'}^{\mathcal{I}_0}$ is in solved form. We know that the series of rules applied in the derivation $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ is the same as in derivation $\overline{\mathcal{C}_0}^{\mathcal{I}_0} \rightsquigarrow^*_\sigma \mathcal{C}'_s$. Moreover, the right-hand sides of the constraints in $\overline{\mathcal{C}}^{\mathcal{I}_0}$ and $\mathcal{C}$ are the same for any $\mathcal{C}$. Hence, we have that the derivation $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ is in $\mathcal{S}$.

Termination/Complexity. First, the strategy yields derivations of polynomial length (see Section 4.7 in [14]). Now, it remains to show that $St(\mathcal{C}') \subseteq St(\mathcal{C}_0\sigma) \subseteq St(\mathcal{C}_0)\sigma$. First, we can show that $var(\mathcal{C}_0\sigma) = var(\mathcal{C}')$ when $\mathcal{C}' \neq \perp$ (by induction on the length of the derivation). We will use this result to prove the second inclusion.

We show the two inclusions by induction on the length of the derivation $\mathcal{C}_0 \rightsquigarrow^n_\sigma \mathcal{C}'$.

Base case: $n = 0$. The result trivially holds.

Induction step: $n > 1$. In such a case, we have that $\mathcal{C}_0 \rightsquigarrow^{n-1}_{\sigma_1} \mathcal{C}_1$, $\mathcal{C}_1 \rightsquigarrow_{\sigma_2} \mathcal{C}'$, and $\sigma = \sigma_2 \circ \sigma_1$. By induction hypothesis, we have that $St(\mathcal{C}_1) \subseteq St(\mathcal{C}_0\sigma_1) \subseteq St(\mathcal{C}_0)\sigma_1$. We distinguish two cases depending on the simplification rules involved in $\mathcal{C}_1 \rightsquigarrow_{\sigma_2} \mathcal{C}'$. For the rules R_1, R_4, and R_5, we have that $St(\mathcal{C}') \subseteq St(\mathcal{C}_1)$ and σ_2 is the identity. Hence, we easily conclude:

$$St(\mathcal{C}') \subseteq St(\mathcal{C}_1) \subseteq St(\mathcal{C}_0\sigma_1) = St(\mathcal{C}_0\sigma) \subseteq St(\mathcal{C}_0)\sigma_1 = St(\mathcal{C}_0)\sigma.$$

For the rules R_2, R_3, and R'_3, we have that $St(\mathcal{C}') = St(\mathcal{C}_1\sigma_2)$. Moreover, we can show that $St(\mathcal{C}_1\sigma_2) \subseteq St(\mathcal{C}_1)\sigma_2$ (see Lemma 4 in Appendix B). By relying on our induction hypothesis, we have that $St(\mathcal{C}_1)\sigma_2 \subseteq St(\mathcal{C}_0)\sigma$. Hence, we can conclude for the first inclusion. We have that:

$$St(\mathcal{C}') = St(\mathcal{C}_1\sigma_2) \subseteq St(\mathcal{C}_1)\sigma_2 \subseteq St(\mathcal{C}_0)\sigma \subseteq St(\mathcal{C}_0\sigma).$$

We have that $dom(\sigma_2) \subseteq var(\mathcal{C}_1)$ since $\sigma_2 = \mathsf{mgu}(t_1, t_2)$ with $t_1, t_2 \in St(\mathcal{C}_1)$. Hence, $\bigcup_{x \in var(\mathcal{C}_1)} St(x\sigma_2) \subseteq St(\mathcal{C}_1\sigma_2) \subseteq St(\mathcal{C}_1)\sigma_2$ (thanks to Lemma 4). We have that:

$$\begin{array}{rcll}
St(\mathcal{C}_0\sigma) \subseteq St(\mathcal{C}_0\sigma_1\sigma_2) & \subseteq & St(\mathcal{C}_0\sigma_1)\sigma_2 \cup \bigcup_{x \in var(\mathcal{C}_0\sigma_1)} St(x\sigma_2) & \\
& = & St(\mathcal{C}_0\sigma_1)\sigma_2 \cup \bigcup_{x \in var(\mathcal{C}_1)} St(x\sigma_2) & \\
& \subseteq & St(\mathcal{C}_0\sigma_1)\sigma_2 \cup St(\mathcal{C}_1)\sigma_2 & \\
& \subseteq & St(\mathcal{C}_0\sigma_1)\sigma_2 & \text{by induction hypothesis} \\
& \subseteq & St(\mathcal{C}_0)\sigma & \text{by induction hypothesis}
\end{array}$$

□

Example 10. *Consider our former example of a special constraint system, i.e., $(\mathcal{C}_1, \mathcal{I})$ (see Example 9), we can simplify the constraint system $\mathcal{C}_1$ following strategy $\mathcal{S}$:*

- $\mathsf{R}_2, \mathsf{R}_3, \mathsf{R}_3'$*:* $\mathcal{C}_1 \rightsquigarrow^*_\sigma \mathcal{C}_2$ *with* $\mathcal{C}_2 = \mathcal{C}_1\sigma$ *where* $\sigma = \{{}^{id}/_{x_{id}}, {}^{x_a::x_l}/_{x_L}\}$,
- R_f*:* $(\mathcal{C}_2, \mathcal{I}) \rightsquigarrow^* (\mathcal{C}_3, \mathcal{I})$ *with*

$$\mathcal{C}_3 = \begin{cases} T_0\sigma \cup \{u_1\sigma\} \Vdash \mathsf{req} & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash \mathsf{rep} & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash S & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash D & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash D & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash S & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash id & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash id & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash x_a & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash x_a & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash x_l & \wedge \quad T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash x_l & \wedge \\ T_0\sigma \cup \{u_1\sigma\} \Vdash \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD}) & & \wedge \\ T_0\sigma \cup \{u_1\sigma, v_2\sigma\} \Vdash \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, x_a :: x_l\rangle, K_{SD}) & & \end{cases}$$

- R_1*:* $(\mathcal{C}_3, \mathcal{I}) \rightsquigarrow^* (\mathcal{C}', \mathcal{I})$ *with* $\mathcal{C}'$ *in solved form defined as follows:*

$$\mathcal{C}' = T_0 \cup \{u_1\} \Vdash x_a \ \wedge T_0 \cup \{u_1\} \Vdash x_l$$

Let $\Phi = \Phi_D \wedge \Phi_S \wedge \neg\mathsf{route}(x_L)$ *(with* Φ_D *and* Φ_S *defined in Example 6). In order to find a solution for* $\mathcal{C}$*, it is equivalent to find a solution for* $(\mathcal{C}_1, \mathcal{I})$ *and*

$$\Phi\sigma = \mathsf{check}(D, x_a) \wedge \mathsf{checkl}(S, x_a :: x_l) \wedge \neg\mathsf{loop}(x_a :: x_l) \wedge \neg\mathsf{route}(x_a :: x_l).$$

The constraint system $(\mathcal{C}', \mathcal{I})$ *is in solved form, and we have that the substitution* θ *defined in Example 6 is such that* $\theta = \theta' \circ \sigma$ *where* $\theta' = \{{}^{X}/_{x_a}, {}^{[W;S]}/_{x_l}\}$ *is a solution of* $(\mathcal{C}', \mathcal{I})$ *and* $\Phi\sigma$*.*

5. Bounding the size of minimal solutions for solved forms

Applying the technique described in the previous section, we are left to decide the existence of a solution for a constraint system $\mathcal{C}$ in solved form together with disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. In this section, we show how to bound the size of a minimal solution for solved constraint systems.

5.1. Bounding variables which are not of sort loc *or* lists

We first prove that given any solution of $\mathcal{C}$, the variables which are not of sort loc or lists can be instantiated by any fresh name, still preserving the solution.

Lemma 3. *Let* $(\mathcal{C}, \mathcal{I})$ *be a special constraint system in solved form,* Φ_1 *be a formula of* $\mathcal{L}_{\mathsf{route}}$*,* Φ_2 *be a set of disequality constraints, and* $G = (\mathcal{N}_{\mathsf{loc}}, E)$ *be a graph. Consider* σ *a solution of* $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ *for graph* G*. There is a solution* σ' *of* $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ *for graph* G *such that:*

- $x\sigma' = x\sigma$ *for every variable* x *of sort* loc *or* lists*;*
- $x\sigma' \in \mathcal{I}$ *otherwise.*

Proof. Since $(\mathcal{C}, \mathcal{I})$ is a special constraint system in solved form, we have that

$$\mathcal{C} = T_1 \Vdash x_1 \wedge \ldots \wedge T_n \Vdash x_n$$

where:

- $x_1, \ldots, x_n$ are distinct variables, and
- $var((\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2) = \{x_1, \ldots, x_n\} = rvar(\mathcal{C})$.

We show the result by induction on:

$$\mu(\sigma) = \#\{x \in rvar(\mathcal{C}) \mid x \text{ is neither of sort } \mathsf{loc} \text{ nor of sort } \mathsf{lists} \wedge x\sigma \notin \mathcal{I}\}.$$

Base case: $\mu(\sigma) = 0$. In such a case, since $rvar(\mathcal{C})$ contains all the variables that occur in the constraint system, we easily conclude. The substitution σ is already of the right form.

Induction step: $\mu(\sigma) > 0$. Let i_0 be the maximal index $1 \leq i_0 \leq n$ such that $x_{i_0}\sigma \notin \mathcal{I}$ and x_{i_0} is not of sort loc or lists. Let a be a name in $\mathcal{I}$ that does not occur elsewhere. Let $\sigma' = \tau \cup \{x_{i_0} \mapsto a\}$ where $\tau = \sigma|_X$ with $X = dom(\sigma) \smallsetminus \{x_{i_0}\}$. Clearly, we have that $\mu(\sigma') < \mu(\sigma)$. In order to conclude, it remains to show that σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$.

1. *We show that σ' is a solution of $(\mathcal{C}, \mathcal{I})$.* For every $i < i_0$, since σ is a solution of $(\mathcal{C}, \mathcal{I})$, we have that $T_i\sigma \cup \mathcal{I} \vdash x_i\sigma$. Since x_{i_0} does not occur in this constraint, we also have that $T_i\sigma' \cup \mathcal{I} \vdash x_i\sigma'$. Since $a \in \mathcal{I}$, we have that $T_{i_0}\sigma' \cup \mathcal{I} \vdash x_{i_0}\sigma'$.
   For every $i > i_0$, according to the definition of i_0, either x_i is of sort loc or lists, or $x_i\sigma \in \mathcal{I}$. In the first case, as for every term t of sort loc or lists, $\mathcal{N}_{\mathsf{loc}} \vdash t$, we have that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma$. In the second case, $\mathcal{I} \vdash x_i\sigma$. Hence, in both cases, we have that $T_i\sigma' \cup \mathcal{I} \vdash x_i\sigma'$.
2. *We show that σ' is a solution of Φ_1.* All the variables appearing in Φ_1 are of type loc or lists. Hence, we have that $\Phi_1\sigma = \Phi_1\sigma'$. This allows us to conclude.
3. *Lastly, we show that σ' is a solution of Φ_2.* Let $\forall Y.u \neq v$ be a disequality constraint in Φ_2. Assume w.l.o.g. that $dom(\sigma) \cap Y = \emptyset$. Since σ is a solution of $\forall Y.u \neq v$, we know that $u\sigma$ and $v\sigma$ are not unifiable.
   Assume by contradiction that there exists a substitution θ' such that $u\sigma'\theta' = v\sigma'\theta'$ (i.e., σ' does not satisfy $\forall Y.u \neq v$). We can assume w.l.o.g. that $u\sigma'\theta'$ and $v\sigma'\theta'$ are ground terms, and $x_{i_0} \notin dom(\theta')$. In such a case, we have that:

   $$\begin{array}{l} (u\sigma')\theta' = ((u\tau)\{x_{i_0} \mapsto a\})\theta' = ((u\tau)\theta')\{x_{i_0} \mapsto a\} \\ (v\sigma')\theta' = ((v\tau)\{x_{i_0} \mapsto a\})\theta' = ((v\tau)\theta')\{x_{i_0} \mapsto a\} \end{array}$$

   Since a is fresh, we deduce that $(u\tau)\theta' = (v\tau)\theta'$. Hence, we have also that:

   $$((u\tau)\theta')\{x_{i_0} \mapsto x_{i_0}\sigma\} = ((v\tau)\theta')\{x_{i_0} \mapsto x_{i_0}\sigma\}$$

   i.e., $u\sigma\theta' = v\sigma\theta'$. This contradicts the fact that $u\sigma$ and $v\sigma$ are not unifiable.

Hence, σ' is a solution of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$. $\square$

5.2. Bounding variables of sort loc *or* lists

We then show that it is possible to find a solution in which lists are polynomially bounded. We need to prove two separate propositions, according to whether the network topology is fixed or not. The proofs of these propositions use the fact that disequality constraints can be satisfied using fresh node names and that the predicates of the logic $\mathcal{L}_{\mathsf{route}}$ check only a finite number of nodes.

5.2.1. Case of a fixed topology

In case the network topology is fixed, we show that we can bound the size of an attack, where the bound depends on the size of the graph and the size of the constraints.

Proposition 4. *Let $(\mathcal{C},\mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph. If there is a solution of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, then there exists a solution σ of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G that is polynomially bounded in the size of Φ_1, Φ_2 and E.*

In particular, we show that, if there is a solution of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G, then there exists a substitution σ such that σ is a solution of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, and variables of sort lists are instantiated by lists of length at most M where M is a bound that depends on Φ_2, Φ_1, and E.

To prove this result, we consider a *smallest* solution σ of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G (i.e., a solution that minimizes the length of the lists that occur in $\mathsf{img}(\sigma)$), and we assume that there exists a variable x_ℓ of sort lists such that $x_\ell\sigma$ is a list of length greater than M. We built a solution σ' (smaller that σ) by changing only the value of $x_\ell\sigma$ in order to reduce its length, preserving the satisfiability of our constraints. We build $x_\ell\sigma'$ by first marking the names we want to keep in $x_\ell\sigma$ obtaining a *marked list*, i.e., a list in which some elements are marked.

For instance, in order to ensure that a loop predicate will still be satisfied, two names are actually sufficient, whereas for a checkl predicate, three names are needed. Note that, to satisfy a positive occurrence of a route predicate, we know that the list contains at most $\#E$ names (since names in the list have to be distinct), thus we know that the variable x_ℓ is not involved in a positive occurrence of a route predicate. We also have to keep some names to take the disequality constraints into account.

Then, we consider the list extracted from $x_\ell\sigma$ by keeping the marked names plus an additional one, and we consider variations of this extracted list. Note that the length of this extracted list is bounded by the size of the graph and the size of the constraints.

Definition 5 (variation). *Let l' be marked list in which at least one of its element is not marked. A* variation *of $l' = [a'_1; \dots; a'_n]$ is a list $l = [a_1; \dots; a_n]$ such that:*

- *there exists* $1 \leq j_0 \leq n$ *such that* a'_{j_0} *is not marked and* a_{j_0} *is a fresh name,*
- *for all* $1 \leq i \leq n$ *such that* $i \neq j_0$*, we have that* $a_i = a'_i$.

Actually, instantiating x_ℓ by any variation of this extracted list allows us to ensure that our constraints are still satisfied.

5.3. *Case of an* a priori *unknown topology*

In case the network topology is not fixed, we show that we can bound the size of an attack. More precisely, we show that if there is an attack on an arbitrary graph, then it is possible to find a polynomially bounded attack, on a possibly slightly different graph.

Proposition 5. *Let* $(\mathcal{C}, \mathcal{I})$ *be a special constraint system in solved form,* Φ_1 *be a conjunction of atomic formulas of* $\mathcal{L}_{\mathsf{route}}$*,* Φ_2 *be a set of disequality constraints. If there is a solution of* $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ *for the graph* $G = (\mathcal{N}_{\mathsf{loc}}, E)$*, then there exists a graph* $G' = (\mathcal{N}_{\mathsf{loc}}, E')$ *and a substitution* σ *such that* σ *is a solution of* $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ *for* G'*, and* σ *is polynomially bounded in the size of* Φ_1 *and* Φ_2*. Moreover, we have that* G' *coincides with* G *on* $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$*, i.e.,* $E = \{(n_1, n_2) \in E' \mid n_1, n_2 \in V\}$.

The proof follows the same lines as the proof of Proposition 4. However, we can not consider the size of the graph to bound the size of the lists. This is used in the proof of Proposition 4 to deal with the case of route that occur positively in the formula. Here, we rely on the fact that we can change the graph to solve this problem, and we consider ubiquitous graphs.

Definition 6 (ubiquitous graph). *Let* $G = (\mathcal{N}_{\mathsf{loc}}, E)$ *be a finite graph (i.e., such that E is finite). Consider the sets of nodes* $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$*, and* $V_{ubi} \subseteq \mathcal{N}_{\mathsf{loc}} \smallsetminus V$*. The graph* $(\mathcal{N}_{\mathsf{loc}}, E \cup E_{ubi})$ *where* $E_{ubi} = \{(a, b) \mid a \in V \cup V_{ubi},\ b \in V_{ubi}\}$ *is called the* ubiquitous graph associated to G and V_{ubi}.

Moreover, we consider ubiquitous variations instead of variations. This is needed to satisfy a formula route that occurs positively.

Definition 7 (ubiquitous variation). *Let* l' *be a marked list and* n *be the number of unmarked elements in* l'*. Let* V_{ubi} *be a set of nodes such that* $\#V_{ubi} > n$ *and names in* V_{ubi} *do not occur in* l'*. A* ubiquitous variation *according to* V_{ubi} *of* $l' = [a'_1; \dots; a'_n]$ *is a list* $l = [a_1; \dots; a_n]$ *such that:*

- *for all* $1 \leq i \leq n$ *such that* a'_i *is not marked,* $a_i \in V_{ubi}$*,*
- *for all* $1 \leq i \leq n$ *such that* a'_i *is marked,* $a_i = a'_i$.

Moreover we require that the ubiquitous nodes of l *are all distinct.*

6. Decidability results

We are now ready to prove our two main decidability results.

Theorem 1. *Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with a hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ such that there is an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Theorem 2. *Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with a hole, $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a finite graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Note that Theorem 1 does not imply Theorem 2 and reciprocally. Theorems 1 and 2 ensure in particular that we can decide whether a routing protocol like SRP can guarantee that any route accepted by the source is indeed a route (a path) in the network (which can be fixed by the user or discovered by the procedure). The NP-hardness of the existence of an attack comes from the NP-hardness of the existence of a solution for deduction constraint systems [32]. The (NP) decision procedures proposed for proving Theorems 1 and 2 involve several steps, with many common ingredients.

Step 1. Applying Theorem 3, it is sufficient to decide whether there exists a sequence of symbolic transitions (and if G is not fixed then we guess which nodes occuring in $\mathcal{P}$ are connected by an edge in G).

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})]; \mathcal{S}; \mathcal{I}; \emptyset) \ \rightarrow^{s*}_{G,\mathcal{M}} \ (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

such that $\mathcal{C} \wedge u = \mathsf{error}$ admits a solution for the graph G. Since processes contain no replication and involve communication between a finite number of nodes, it is possible to guess the sequence of symbolic transitions yielding an attack (by guessing also the edges between the nodes that are either in $\mathcal{M}$ or involved in a communication step) and the resulting configuration stays of size polynomially bounded by the size of the initial configuration. Moreover, any left-hand-side of a deduction constraint in $\mathcal{C}$ is of the form $T \cup \mathcal{N}_{\mathsf{loc}}$ where T is a finite set of terms. It then remains to decide the existence of a solution for our class of constraint systems.

Step 2. We first get rid of unification constraints by computing a mgu of the set of equality constraints. Then applying Theorem 4, we deduce that it is sufficient to decide the existence of a solution for constraint systems in solved form, by (non-deterministically) guessing a polynomial sequence of transformation rules that yields a solved constrain system.

Step 3. We then show how to decide the existence of a solution for a constraint system, where each deduction constraint is solved, that is of the form $T \Vdash x$. It is not straightforward like in [14] since we are left with (non solved) disequality constraints and formulas. Since the system is in solved form and applying Lemma 3, we can replace any variable that are not of sort loc or lists by any fresh name. Then the key step consists in guessing a small solution. Propositions 4 and 5 (depending on whether the graph is fixed or not) ensure that the size of a minimal attack is polynomially bounded in the size of the constraint system (and in the size of the graph when it is fixed), which also bounds the total number of names that are used. We can therefore guess the instantiation of the variables of sort loc and lists.

These three steps describe a (non-deterministic) procedure for deciding the existence of an attack on any concrete configuration.

Complexity. Assume given a constraint system $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ are respectively the sets of unification constraints, deduction constraints, disequality constraints and formulas. We first apply an mgu of the unification constraints in $\mathcal{C}_1$, the size of the constraint system obtained is polynomial in the size of $\mathcal{C}$, so we can assume that $\mathcal{C}_1$ is empty. Then, by Theorem 4, we can apply (non-deterministically) a polynomial number of simplification rules and get a constraint system $\mathcal{C}_2\sigma \wedge \mathcal{C}_3\sigma \wedge \mathcal{C}_4\sigma$ where $\mathcal{C}_2\sigma$ is in solved form. Then, as shown in the proofs of Propositions 5 and 4, we can actually bound the size of a minimal solution polynomially in $\mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$.

7. Applications

7.1. Routing protocol SRP applied to DSR

Our decision procedure allows us to retrieve the attack on the protocol SRP applied to DSR, mentioned in Example 5. Indeed, consider the formal model of SRP applied to DSR (defined in Section 2.3) and of its desired property (defined in Example 5). We would first guess the graph G_0 defined in Example 3. Executing symbolically (non deterministically) the process modeling SRP applied to DSR, we would obtain the symbolic configuration of Example 7. Applying our transformation rules, we would then (non deterministically) obtain a solved constraint system (see Example 10). We can finally guess the (bounded) solution $\theta' = \{^{X}/_{x_a}, ^{[W;S]}/_{x_l}\}$.

7.2. Routing protocol SDMSR

The secured routing protocol SDMSR introduced in [8] is a multipath routing protocol that can be modeled in our framework. Its aim is to find several paths leading from the source S to the destination D. To achieve this, the intermediate nodes may process the same request several times. This protocol is based on two authentication mechanisms: RSA signatures and signatures based on hash chains. The purpose of the latter scheme is to decrease computation time. For the sake of simplicity, we describe the protocol without this mechanism. The description in [8] does not really state whether neighbor verification is performed in the protocol. To avoid straightforward attacks, we assume that this is the case: each node checks whether the received information is consistent with its knowledge of the network.

To discover a route to the destination, the source constructs a request packet and broadcasts it to its neighbors. The request packet contains its name S, the name of the destination D, an identifier of the request id, a list containing the beginning of a route to D, and a signature over the content of the request, computed with the private key $\mathsf{priv}(S)$. The source then waits for a reply containing a route to D signed by one of his neighbors, and checks that this route is plausible.

The process executed by a source node S initiating the search of a route towards a destination node D is

$$P_{\mathsf{init}}(S, D) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].\mathsf{null}$$

$$\text{where} \begin{cases} u_1 = \langle \mathsf{req}, S, D, id, S :: [], [\![\langle \mathsf{req}, S, D, id\rangle]\!]_{\mathsf{priv}(S)}\rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_A, x_L, [\![\langle \mathsf{rep}, D, S, id, x_L\rangle]\!]_{\mathsf{priv}(x_A)}\rangle \\ \Phi_S = \mathsf{check}(S, x_A) \wedge \mathsf{checkl}(S, x_L) \end{cases}$$

The names of the intermediate nodes are accumulated in the route request packet. Intermediate nodes relay the request over the network, except if they have already seen a shorter one. In order to simplify the presentation, we consider that they relay all requests as long as they contain different routes. An intermediate node also checks that the received request is correctly authenticated by checking the attached signature. Below, $V \in \mathcal{N}_{\mathsf{loc}}$, x_S, x_a and x_D are variables of sort loc whereas x_r is a variable of sort lists and x_{id} is a variable of sort terms. The process executed by an intermediate node V when forwarding a request is as follows:

$$P_{\mathsf{req}}(V) = \mathsf{in}\ w_1[\Phi_V].\mathsf{read}\ t\ \mathsf{then\ null\ else}\ (\mathsf{store}(t).\mathsf{out}(w_2))$$

$$\text{where} \begin{cases} w_1 = \langle \mathsf{req}, x_S, x_D, x_{id}, x_a :: x_r, [\![\langle \mathsf{req}, x_S, x_D, x_{id}\rangle]\!]_{\mathsf{priv}(x_S)}\rangle \\ \Phi_V = \mathsf{check}(V, x_a) \\ t = \langle x_S, x_D, x_{id}, x_a :: x_r\rangle \\ w_2 = \langle \mathsf{req}, x_S, x_D, x_{id}, V :: x_a :: x_r, [\![\langle \mathsf{req}, x_S, x_D, x_{id}\rangle]\!]_{\mathsf{priv}(x_S)}\rangle \end{cases}$$

When the request reaches the destination D, he checks that the request comes from one of his neighbors, has a correct signature, and that the list of accumulated nodes does not contain a loop. Then, the destination D constructs a route reply, in particular it computes a signature over the route accumulated in the request packet with its private key $\mathsf{priv}(D)$. He then sends the reply back over the network. The process executed by the destination node D is $P_{\mathsf{dest}}(D) = \mathsf{in}\ v_1[\Phi_D].\mathsf{out}(v_2).\mathsf{null}$ where:

$$v_1 = \langle \mathsf{req}, x_S, D, x_{id}, x_b :: x_l, [\![\langle \mathsf{req}, x_S, D, x_{id}\rangle]\!]_{\mathsf{priv}(x_S)}\rangle$$
$$\Phi_D = \mathsf{check}(D, x_b) \wedge \neg\mathsf{loop}(x_b :: x_l)$$
$$v_2 = \langle \mathsf{rep}, D, x_S, x_{id}, D, D :: x_b :: x_l, [\![\langle \mathsf{rep}, D, x_S, x_{id}, D :: x_b :: x_l\rangle]\!]_{\mathsf{priv}(D)}\rangle$$

Then, the reply travels along the route back to S. The intermediate nodes check that the signature in the reply packet is correct, and that the route is plausible, before forwarding it. Each node replaces the signature in the reply packet by its own signature. The process executed by an intermediate node V when forwarding a reply is the following one:

$$P_{\mathsf{rep}}(V) = \mathsf{in}\ w'[\Phi'_V].\mathsf{out}(w'').\mathsf{null}$$

$$\text{where} \quad \left\{ \begin{array}{l} w' = \langle \mathsf{rep}, x_D, x_S, x_{id}, x_a, x_r, [\![\langle \mathsf{rep}, x_D, x_S, x_{id}, x_r\rangle]\!]_{\mathsf{priv}(x_a)}\rangle \\ \Phi'_V = \mathsf{checkl}(V, x_r) \wedge \mathsf{check}(V, x_a) \\ w'' = \langle \mathsf{rep}, x_D, x_S, x_{id}, V, x_r, [\![\langle \mathsf{rep}, x_D, x_S, x_{id}, x_r\rangle]\!]_{\mathsf{priv}(V)}\rangle \end{array} \right.$$

We have found that SDMSR is subject to the same kind of attack as SRP applied to DSR. Consider the same graph G_0 as for the attack we described on SRP. Let

$$K_0 = (\lfloor P_{\mathsf{init}}(S,D)\rfloor_S \mid \lfloor P_{\mathsf{dest}}(D)\rfloor_D; \emptyset; \mathcal{I}_0)$$

The attack scenario is the following one. The source S sends a route request towards D. The request reaches the node n_I. Thus, the attacker receives the following message: $\langle \mathsf{req}, S, D, id, S :: [], [\![\langle \mathsf{req}, S, D, id\rangle]\!]_{\mathsf{priv}(S)}\rangle$. The attacker then broadcasts the following message in the name of X:

$$\langle \mathsf{req}, S, D, id, [X; W; I; S], [\![\langle \mathsf{req}, S, D, id\rangle]\!]_{\mathsf{priv}(S)}\rangle.$$

Since D is a neighbor of n_I, it will hear the transmission. In addition, the list of nodes $[X; W; I; S]$ ends with X, which is also a neighbor of D, and does not contain any loop, and signature $[\![\langle \mathsf{req}, S, D, id\rangle]\!]_{\mathsf{priv}(S)}$ is valid. Consequently, the destination D will process this request and will send the following route reply back to S:

$$\langle \mathsf{rep}, D, S, id, D, [D; X; W; I; S], [\![\langle \mathsf{rep}, D, S, id, [D; X; W; I; S]\rangle]\!]_{\mathsf{priv}(D)}\rangle.$$

The attacker will put its own signature $[\![\langle \mathsf{rep}, D, S, id, [D; X; W; I; S]\rangle]\!]_{\mathsf{priv}(n_I)}$ instead of the signature of D, and it will send the resulting message to S.

To model security in our model, we replace in P_{init} the process null by a hole and we check whether the formula $\neg\mathsf{route}(x_L)$ holds. Applying our procedure to the initial configuration K_0, we can reach the configuration

$$K_s = (\lfloor \mathsf{out}(\mathsf{error}).0\rfloor_S; \emptyset; \mathcal{I}_0 \cup \{u_1, v_2\}; \mathcal{C})$$

where

$$\mathcal{C} = \left\{ \begin{array}{l} \mathcal{I}_0 \cup \{u_1\} \Vdash v_1 \ \wedge \ \Phi_D \ \wedge \\ \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2 \ \wedge \ \Phi_S \ \wedge \ \neg\mathsf{route}(x_L) \end{array} \right\}$$

with:

$$\begin{array}{l}
u_1 = \langle \mathsf{req}, S, D, id, S :: [], [\![\langle \mathsf{req}, S, D, id\rangle]\!]_{\mathsf{priv}(S)}\rangle \\
u_2 = \langle \mathsf{rep}, D, S, id, x_A, x_L, [\![\langle \mathsf{rep}, D, S, id, x_L\rangle]\!]_{\mathsf{priv}(x_A)}\rangle \\
\Phi_D = \mathsf{check}(D, x_b) \wedge \neg\mathsf{loop}(x_b :: x_l) \\
\Phi_S = \mathsf{check}(S, x_A) \wedge \mathsf{checkl}(S, x_L) \\
v_1 = \langle \mathsf{req}, x_S, D, x_{id}, x_b :: x_l, [\![\langle \mathsf{req}, x_S, D, x_{id}\rangle]\!]_{\mathsf{priv}(x_S)}\rangle \\
v_2 = \langle \mathsf{rep}, D, x_S, x_{id}, D, D :: x_b :: x_l, [\![\langle \mathsf{rep}, D, x_S, x_{id}, D :: x_b :: x_l\rangle]\!]_{\mathsf{priv}(D)}\rangle
\end{array}$$

and the constraint system $\mathcal{C}$ has a solution

$$\theta = \{{}^{id}/_{x_{id}}, {}^{S}/_{x_S}, {}^{n_I}/_{x_A}, {}^{X}/_{x_b}, {}^{[W;n_I;S]}/_{x_l}, {}^{[D;X;W;n_I;S]}/_{x_L}\}$$

for graph G_0, so there is an $\{n_I\}$-attack on K_0 for G_0. We therefore retrieve the attack mentioned above, that we have discovered while analysing the protocol.

7.3. Other routing protocols

The model of processes we propose includes the possibility for nodes to store information in some memory. We can therefore model routing protocols based on routing tables, such as SAODV [37], SEAD [17] and ARAN [33]. However, in such protocols, the actual found route is not sent to the source node but depends on the internal states of the nodes. Security properties such as route validity can thus not be expressed using our route predicate.

We have modeled route validity in Example 5 for the protocol SRP applied to DSR. The same modeling can be applied to most source routing protocols such as Ariadne [18], endairA [11], SRDP [23], BISS [12]. However, source routing protocols may also perform recursive tests. Such tests are typically performed either by the source or the destination and aim at securing respectively the request or reply phase. These tests can not yet be included in our decision procedures.

8. Conclusion

Using our symbolic semantics, we have shown that, for general processes that can broadcast and perform some correctness checks in addition to the usual pattern matching, existence of an attack can be reduced to existence of a solution for (generalized) constraint systems. Our result holds for a bounded number of sessions and an unbounded number of node names. As an illustration, for a large class of processes without replication (that captures e.g., routing protocols SRP applied on DSR, SDMSR,...), we have proved that the existence of an attack is NP-complete. In particular, we generalize existing works on solving constraint systems to properties like the validity of a route and to protocols with broadcasting. Our result enables us to both decide the existence of a network topology that would lead to an attack, and also to automatically discover whether a particular network topology may allow malicious nodes to mount an attack.

As future work and in order to model protocols, we plan to extend our results to protocols that perform recursive tests or make use of recursive functions. Since our results reuse existing techniques such as constraint solving, we believe that our procedure could be implemented in existing tools after a few adaptations. Another extension would be to model mobility during the execution of the protocol. This would allows us to consider changes in the network topology and to analyze the security of route updates. This requires modelling of an appropriate security property.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[2] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[3] G. Ács, L. Buttyán, and I. Vajda. Provable security of on-demand distance vector routing in wireless ad hoc networks. In *Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2005)*, 2005.

[4] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *The Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2006)*, 2006.

[5] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[6] A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[7] D. Benetti, M. Merro, and L. Viganò. Model checking ad hoc network routing protocols: ARAN vs. endairA. In *Proceedings of the 8th IEEE International Conference on Software Engineering and Formal Methods (SEFM'10)*. IEEE Computer Society Press, 2010.

[8] S. Berton, H. Yin, C. Lin, and G. Min. Secure, disjoint, multipath source routing protocol(sdmsr) for mobile ad-hoc networks. In *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, GCC '06, pages 387–394, Washington, DC, USA, 2006. IEEE Computer Society.

[9] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc. Press, 2001.

[10] B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *Proc. 20th International Conference on Automated Deduction (CADE'05)*, 2005.

[11] L. Buttyán and I. Vajda. Towards Provable Security for Ad Hoc Routing Protocols. In *Proc. 2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04)*, pages 94–105, New York, NY, USA, 2004. ACM.

[12] S. Capkun and J.-P. Hubaux. Biss: Building secure routing out of an incomplete set of security associations, 2003.

[13] J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.

[14] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 11(4):496–520, 2010.

[15] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. Workshop on Formal Methods and Security Protocols*, 1999.

[16] J. C. Godskesen. Formal verification of the aran protocol using the applied pi-calculus. In *Proceedings of the Sixth International IFIP WG 1.7 Workshop on Issues in the Theory of Security*, pages 99–113, 2006.

[17] Y.-C. Hu, D. B. Johnson, and A. Perrig. Sead: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1):175–192, July 2003.

[18] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11, January 2005.

[19] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006.

[20] I. John D. Marshall. An analysis of the secure routing protocol for mobile ad hoc network route discovery: Using intuitive reasoning and formal verification to identify flaws. Technical report, Florida State University, 2003.

[21] D. Johnson, D. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for multi-hop wireless ad hoc networks. In *Ad Hoc Networking*, pages 139–172, 2001.

[22] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. The MIT press, Cambridge (MA, USA), 1991.

[23] J. Kim and G. Tsudik. Srdp: Secure route discovery for dynamic source routing in manets. *Ad Hoc Netw.*, 7(6):1097–1109, 2009.

[24] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Wireless Communications and Networking Conference*, volume 2, pages 1193–1199 Vol. 2, 2005.

[25] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

[26] S. Nanz and C. Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1):203–227, 2006.

[27] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proc. SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.

[28] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[29] C. E. Perkins and E. M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *Proc. 2nd Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pages 90–100, 1999.

[30] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, Sept. 2002.

[31] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux. Towards Provable Secure Neighbor Discovery in Wireless Networks. In *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, pages 31–42. ACM, 2008.

[32] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Comp. Soc. Press, 2001.

[33] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. In *10th IEEE International Conference of Network Protocols (ICNP)*, 2002.

[34] P. Schaller, B. Schmidt, D. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Proc. 22nd Computer Security Foundations Symposium (CSF'09)*. IEEE Comp. Soc. Press, 2009.

[35] F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[36] S. Yang and J. S. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proc. 1st ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03)*, 2003.

[37] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proc. 1st ACM workshop on Wireless SEcurity (WiSE'02)*, pages 1–10. ACM, 2002.

Appendix A. Symbolic semantics

We show in Lemma 1 that the result of a transition from a ground symbolic configuration is also a ground symbolic configuration, in particular the set of constraints obtained is a constraint system. This lemma will be useful to show that our transition system is complete (Proposition 1) and sound (Proposition 2) when considering ground configurations.

Lemma 1. *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$, and $K_s = (\mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$ be a ground symbolic configuration. If K'_s is such that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$, then K'_s is a ground symbolic configuration.*

Proof. Since K_s is a symbolic configuration, we have that $\mathcal{C}$ is a constraint system and $T \subseteq \mathcal{I}$ for every $T \Vdash u \in \mathcal{C}$. Moreover, since K_s is ground, we have that $\mathit{var}(\mathcal{I}) \cup \mathit{fv}(\mathcal{P}) \cup \mathit{var}(\mathcal{S}) \subseteq \mathit{rvar}(\mathcal{C})$. Let $K'_s = (\mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}')$ and $G = (V, E)$. To prove the result, we do a case analysis on the transition rule involved in $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$. Note that the result is straightforward for the rules Store$_s$, Par$_s$, Repl$_s$, and New$_s$. Indeed, in these cases, we have that $\mathcal{C}' = \mathcal{C}$, $\mathcal{I}' = \mathcal{I}$, and $\mathit{fv}(\mathcal{P}) \cup \mathit{var}(\mathcal{S}) = \mathit{fv}(\mathcal{P}') \cup \mathit{var}(\mathcal{S}')$. Now, we consider the remaining rules in turn.

- Rule Read-Then$_s$. We have that:

  $$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge t = u)$$

  where $\lfloor t \rfloor_n \in \mathcal{S}$.

  First we have that $\mathcal{C}'$ is a constraint system. Indeed, monotonicity is still satisfied by $\mathcal{C}'$. Moreover, we have that $\mathit{var}(t) \subseteq \mathit{var}(\mathcal{S}) \subseteq \mathit{rvar}(\mathcal{C})$ (since K_s is ground). Hence, $\mathcal{C}'$ satisfies the origination property. Since $\mathcal{I}' = \mathcal{I}$ and the deduction constraints are the same in $\mathcal{C}$ and $\mathcal{C}'$, we have that $T \subseteq \mathcal{I}'$ for every $T \Vdash u \in \mathcal{C}'$. Lastly, since K_s is ground, we have that:

  $$(\mathit{fv}(P) \smallsetminus \mathit{var}(u)) \cup \mathit{fv}(\mathcal{Q}) \subseteq \mathit{rvar}(\mathcal{C}).$$

  Consequently, we have that $\mathit{fv}(P) \cup \mathit{fv}(\mathcal{Q}) \subseteq \mathit{rvar}(\mathcal{C}) \cup \mathit{var}(u)$, and since $\mathit{rvar}(\mathcal{C}) \cup \mathit{var}(u) = \mathit{rvar}(\mathcal{C} \cup \{t = u\})$, we deduce that the resulting symbolic configuration K'_s is also ground.

- Rule Read-Else$_s$. We have that:

  $$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \mathsf{Eq})$$

  where $\mathsf{Eq} = \{\forall \mathit{var}(u) \smallsetminus \mathit{rvar}(\mathcal{C}) \,.\, t \neq u \mid \lfloor t \rfloor_n \in \mathcal{S}\}$.

  Since no deduction or unification constraint is introduced, $\mathcal{C}'$ is a constraint system. Since $\mathcal{I}' = \mathcal{I}$, we also have that $T \subseteq \mathcal{I}'$ for every $T \Vdash u \in \mathcal{C}'$. Since K_s is ground, we have that $\mathit{fv}(Q) \cup \mathit{fv}(\mathcal{Q}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C})$. Since $\mathit{rvar}(\mathcal{C}') = \mathit{rvar}(\mathcal{C})$, we have that $\mathit{fv}(Q) \cup \mathit{fv}(\mathcal{Q}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C}')$. The resulting configuration K'_s is ground.

- Rule IF-THEN$_s$. We have that:

$$(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \Phi)$$

  It is easy to see that $\mathcal{C}'$ is still a constraint system. Moreover, since $\mathcal{I}' = \mathcal{I}$ and $\mathcal{C}' \smallsetminus \mathcal{C}$ does not contain any deduction contraint, we have that $T \subseteq \mathcal{I}'$ for every $T \Vdash u \in \mathcal{C}'$. Since K_s is ground, we have that $\mathit{fv}(P) \cup \mathit{fv}(\mathcal{Q}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C})$. Since $\mathit{rvar}(\mathcal{C}') = \mathit{rvar}(\mathcal{C})$, we have that $\mathit{fv}(P) \cup \mathit{fv}(\mathcal{Q}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C}')$. Thus, the configuration K'_s is ground.

- Rule IF-ELSE$_s$. Similar to the previous case.

- Rule IN$_s$. We have that:

$$(\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \mathcal{I} \Vdash u \wedge \Phi)$$

  where $(n_I, n) \in E$ for some $n_i \in \mathcal{M}$.

  Since $\mathcal{C}$ is a constraint system and $T \subseteq \mathcal{I}$ for any $T \Vdash u \in \mathcal{C}$, we deduce that $\mathcal{C}'$ satisfies the monotonicity property. Since $\mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C})$ (because K_s is ground), $\mathcal{C}'$ satisfies the origination property. Clearly, we have that $T \subseteq \mathcal{I}'$ for any $T \Vdash u \in \mathcal{C}'$. Lastly, since K_s is ground, we have that: $\mathit{fv}(P) \cup \mathit{fv}(\mathcal{Q}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C}) \cup \mathit{var}(u)$. Since $\mathit{rvar}(\mathcal{C}') = \mathit{rvar}(\mathcal{C}) \cup \mathit{var}(u)$, we easily deduce that the symbolic configuration K'_s is ground.

- Rule COMM$_s$. We have that:

$$\begin{aligned}(\lfloor \mathsf{out}(t).P\rfloor_n \cup \mathcal{P}_I \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) &\rightarrow^s_{G,\mathcal{M}} \\ &(\lfloor P\rfloor_n \cup \mathcal{P}_J \cup \mathcal{P}_{K,L} \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}'; \mathcal{C} \wedge \mathcal{C}_J \wedge \mathcal{C}_K \wedge \mathcal{C}_L)\end{aligned}$$

  where:

  - $\mathcal{P}_I = \{\lfloor \mathsf{in}\ u_i[\Phi_i].P_i\rfloor_{n_i} \mid i \in I\}$,
  - $\mathcal{P}_J = \{\lfloor P_j\rfloor_{n_j} \mid j \in J\}$
  - $\mathcal{P}_{K,L} = \{\lfloor \mathsf{in}\ u_k[\Phi_k].P_k\rfloor_{n_k} \mid k \in K \cup L\}$
  - $\mathcal{C}_J = \{t = u_j \wedge \Phi_j \mid j \in J\}$,
  - $\mathcal{C}_K = \{\forall \mathit{var}(u_k) \smallsetminus \mathit{rvar}(\mathcal{C}) \,.\, t \neq u_k \mid k \in K\}$,
  - $\mathcal{C}_L = \{t = u_l\alpha_l \wedge \neg\Phi_l\alpha_l \mid l \in L\}$.

  $\lfloor P'\rfloor_{n'} \in \mathcal{Q}$ implies that $(n, n') \notin E$ or P' is not of the form $\mathsf{in}\ u'[\Phi'].Q'$, $I = J \uplus K \uplus L$, $(n_i, n) \in E$ for any $i \in I$, α_l is a renaming of $\mathit{var}(u_l) \smallsetminus \mathit{rvar}(\mathcal{C})$ by fresh variables, and if $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$ then $\mathcal{I}' = \mathcal{I} \cup \{t\}$ else $\mathcal{I}' = \mathcal{I}$.

  Clearly, $\mathcal{C}'$ satisfies the monotocity property. Moreover, we have that $T \subseteq \mathcal{I}'$ for any $T \Vdash u \in \mathcal{C}'$. To show that $\mathcal{C}'$ satisfies the origination property, we have to prove that $\mathit{var}(t) \subseteq \mathit{rvar}(\mathcal{C})$. This is indeed the case since K_s is ground and $\mathit{var}(t) \subseteq \mathit{fv}(\mathcal{P})$. Lastly, we have to show that K'_s is ground. Since K_s is ground, we have that:

1. $fv(\mathcal{P}_I) \subseteq rvar(\mathcal{C})$
2. $fv(P) \cup fv(\mathcal{Q}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C})$
3. $var(t) \subseteq rvar(\mathcal{C})$.

From 1, we deduce that $fv(P_J) \cup fv(\mathcal{P}_{K,L}) \subseteq rvar(\mathcal{C}) \cup \bigcup_{j \in J} var(u_j)$.

Moreover, we have that $rvar(\mathcal{C}) \cup \bigcup_{j \in J} var(u_j) \subseteq rvar(\mathcal{C}')$.

Hence, we have that:

– $fv(P) \cup fv(\mathcal{Q}) \cup fv(\mathcal{P}_J) \cup fv(\mathcal{P}_{K,L}) \subseteq rvar(\mathcal{C}')$,
– $var(\mathcal{S}') = var(\mathcal{S}) \subseteq rvar(\mathcal{C}) \subseteq rvar(\mathcal{C}')$,
– $var(\mathcal{I}') \subseteq var(\mathcal{I}) \cup var(t) \subseteq rvar(\mathcal{C}) \subseteq rvar(\mathcal{C}')$.

We easily conclude that K'_s is a ground symbolic configuration. □

We now show that a concrete transition corresponds to a symbolic transition.

Proposition 1 (completeness). *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ be a ground symbolic configuration and θ be a solution of $\mathcal{C}$. Let K_c be the θ-concretization of K_s. Let K'_c be a concrete configuration such that $K_c \rightarrow_{G,\mathcal{M}} K'_c$. Then there exists a ground symbolic configuration K'_s and a substitution θ' such that:*

- *K'_c is the θ'-concretization of K'_s, and*
- *$K_s \rightarrow^s_{G,\mathcal{M}} K'_s$.*

Proof. Let $K_c = (\mathcal{P}; \mathcal{S}; \mathcal{I})$. We distinguish cases depending on which transition is applied to K_c. We show that there exists a symbolic configuration K'_s such that K'_c is the θ'-concretization of K'_s and $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$. Thanks to Lemma 1, we easily deduce that K'_s is ground.

- Rule PAR. We have that:

$$K_c = (\lfloor P_1 | P_2 \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P_1 \rfloor_n \cup \lfloor P_2 \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) = K'_c$$

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor P^s_1 | P^s_2 \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$, $P^s_1\theta = P_1$, and $P^s_2\theta = P_2$. Let $K'_s = (\lfloor P^s_1 \rfloor_n \cup \lfloor P^s_2 \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the PAR$_s$ rule), θ is a solution of $\mathcal{C}$ and K'_c is the θ-concretization of K'_s.

- Rule REPL. We have that:

$$K_c = (\lfloor !P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P\alpha \rfloor_n \cup \lfloor !P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) = K'_c$$

  where α is a fresh renaming of the bound variables in P. Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s =$

$(\lfloor !P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$ and $P_s\theta = P$. Note that α is also a renaming of the variables in $bv(P_s) \smallsetminus rvar(\mathcal{C})$. Let $K'_s = (\lfloor P_s\alpha \rfloor_n \cup \lfloor !P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the $\textsc{Repl}_s$ rule) and θ is a solution of $\mathcal{C}$. It remains to show that K'_c is the θ-concretization of K'_s. Since the variables introduced by α are fresh, we have that $\mathsf{img}(\alpha) \cap dom(\theta) = \emptyset$, and since $P_s\theta = P$, we have that $dom(\alpha) \cap dom(\theta) = \emptyset$. Hence we have that $(P_s\alpha)\theta = (P_s\theta)\alpha = P\alpha$. This allows us to conclude.

- Rule $\textsc{New}$. We have that:

  $$K_c = (\lfloor \mathsf{new}\ m.P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P\{^{m'}/_{m}\} \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) = K'_c$$

  where m' is a fresh name.

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{new}\ m.P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$, and $P_s\theta = P$. Let $K'_s = (\lfloor P_s\{^{m'}/_{m}\} \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the $\textsc{New}_s$ rule), θ is a solution of $\mathcal{C}$ and K'_c is the θ-concretization of K'_s.

- Rule $\textsc{Store}$. We have that:

  $$K_c = (\lfloor \mathsf{store}(t).P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{Q}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I}) = K'_c$$

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{store}(t_s).P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$, $P_s\theta = P$ and $t_s\theta = t$. Let $K'_s = (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the $\textsc{Store}_s$ rule), θ is a solution of $\mathcal{C}$ and K'_c is the θ-concretization of K'_s.

- Rule $\textsc{Read-Then}$. We have that:

  $$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \mathcal{Q}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P\sigma \rfloor_n \cup \mathcal{Q}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I})$$

  where $\sigma = \mathsf{mgu}(t, u)$.

  Note that t is ground since K_c is a ground concrete configuration, and thus we have that $u\sigma = t$. Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{read}\ u_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\mathcal{Q}_s\theta = \mathcal{Q}$, $u_s\theta = u$, $t_s\theta = t$, $P_s\theta = P$, $Q_s\theta = Q$, $\mathcal{S}_s\theta = \mathcal{S}$ and $\mathcal{I}_s\theta = \mathcal{I}$. Let $K'_s = (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C} \wedge t_s = u_s)$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the $\textsc{Read-Then}_s$ rule). Let $\theta' = \theta \cup \sigma$. To show that θ' is a solution of $\mathcal{C}$, it remains to prove that $(t_s\theta)\sigma = (u_s\theta)\sigma$. Actually, we have that $(t_s\theta)\sigma = t\sigma = t = u\sigma = (u_s\theta)\sigma$. Lastly, we have that $P_s\theta' = (P_s\theta)\sigma = P\sigma$. Since K_s is a ground symbolic configuration, we have that $fv(\mathcal{Q}_s) \cup var(\mathcal{S}_s) \cup var(\mathcal{I}_s) \subseteq rvar(\mathcal{C}) = dom(\theta)$. Thus $\mathcal{Q}_s\theta' = \mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta' = \mathcal{S}_s\theta = \mathcal{S}$, and $\mathcal{I}_s\theta' = \mathcal{I}_s\theta = \mathcal{I}$. Hence, we have that K'_c is the θ'-concretization of K'_s.

- Rule Read-Else. We have that:

  $$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I})$$

  and for all $\lfloor t \rfloor_n \in \mathcal{S}$ we have that $\mathsf{mgu}(t,u) = \bot$.

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{read}\ u_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $u_s\theta = u$, $P_s\theta = P$, $Q_s\theta = Q$, $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, and $\mathcal{I}_s\theta = \mathcal{I}$.

  Let $K_s' = (\lfloor Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}')$ where $\mathcal{C}' = \mathcal{C} \wedge \{\forall var(u_s) \smallsetminus rvar(\mathcal{C}).t_s \neq u_s \mid \lfloor t_s \rfloor_n \in \mathcal{S}\}$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K_s'$ (with the Read-Else$_s$ rule). Now, let us show that θ is a solution of $\mathcal{C}'$. Let $\forall var(u_s) \smallsetminus rvar(\mathcal{C}).t_s \neq u_s$ be a disequation in $\mathcal{C}' \smallsetminus \mathcal{C}$. We have that $u_s\theta = u$, $t_s\theta = t$ for some term t such that $\lfloor t \rfloor_n \in \mathcal{S}$, and $\mathsf{mgu}(t,u) = \bot$. Thus, θ is also a solution of this constraint, and more generally θ is a solution of $\mathcal{C}'$. Now, it is easy to see that K_c' is the θ-concretization of K_s'.

- Rule If-Then. We have that:

  $$K_c = (\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) = K_c'$$

  and $[\![\Phi]\!]_G = 1$.

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{if}\ \Phi_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $\Phi_s\theta = \Phi$, $P_s\theta = P$, $Q_s\theta = Q$, $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, and $\mathcal{I}_s\theta = \mathcal{I}$.

  Let $K_s' = (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C} \wedge \Phi_s)$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K_s'$ (with the If-Then$_s$ rule). By hypothesis, we have that θ is a solution of $\mathcal{C}$, and as $[\![\Phi_s\theta]\!]_G = [\![\Phi]\!]_G$ is true, we easily deduce that θ is a solution of $\mathcal{C}' = \mathcal{C} \wedge \Phi_s$. Lastly, it is easy to see that K_c' is the θ-concretization of K_s'.

- Rule If-Else. Similar to the previous case.

- Rule In. We have that:

  $$K_c = (\lfloor \mathsf{in}\ u[\Phi].P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P\sigma \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) = K_c'$$

  with $(n_I, n) \in E$ for some $n_I \in \mathcal{M}$, $\sigma = \mathsf{mgu}(t,u)$, $\mathcal{I} \vdash t$ and $[\![\Phi\sigma]\!]_G = 1$.

  Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{in}\ u_s[\Phi_s].P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $u_s\theta = u$, $\Phi_s\theta = \Phi$, $P_s\theta = P$, $\mathcal{Q}_s\theta = \mathcal{Q}$, $\mathcal{S}_s\theta = \mathcal{S}$, and $\mathcal{I}_s\theta = \mathcal{I}$. Let $K_s' = (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}')$ where $\mathcal{C}' = \mathcal{C} \wedge \mathcal{I}_s \Vdash u_s \wedge \Phi_s$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K_s'$ (with the In$_s$ rule). Let $\theta' = \theta \cup \sigma$. By hypothesis, we have that θ is a solution of $\mathcal{C}$. To show that θ' is a solution of $\mathcal{C}'$, it remains to establish that:

  - $(\mathcal{I}_s\theta)\sigma \vdash (u_s\theta)\sigma$: We have that $(\mathcal{I}_s\theta)\sigma = \mathcal{I}\sigma = \mathcal{I}$ since $var(\mathcal{I}) = \emptyset$, and $(u_s\theta)\sigma = u\sigma = t$. Since by hypothesis, we have that $\mathcal{I} \vdash t$, we easily conclude.

- $[\![(\Phi_s\theta)\sigma]\!]_G = 1$. Actually, we have that $(\Phi_s\theta)\sigma = \Phi\sigma$. Since, by hypothesis, we have that $[\![\Phi\sigma]\!]_G = 1$, we easily conclude.

Hence, we have that θ' is a solution of $\mathcal{C}'$. It is easy to see that K'_c is the θ'-concretization of K'_s.

- Rule Comm. We have that

$$K_c = (\lfloor \mathsf{out}(t).P \rfloor_n \cup \{\lfloor \mathsf{in}\ u_j[\Phi_j].P_j \rfloor_{n_j} \mid j \in J\} \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}) \\ \to_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \lfloor P_j\sigma_j \rfloor_{n_j}\} \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}') = K'_c$$

where:

- $\sigma_j = \mathsf{mgu}(t, u_j)$, $(n, n_j) \in E$, and $[\![\Phi_j\sigma_j]\!]_G = 1$ for any $j \in J$,
- if $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$ then $\mathcal{I}' = \mathcal{I} \cup \{t\}$ else $\mathcal{I}' = \mathcal{I}$.

Moreover, we know that $\lfloor P' \rfloor_{n'} \in \mathcal{Q}$ implies that:

- $(n, n') \notin E$, or
- P' is not of the form $\mathsf{in}\ u'[\Phi'].Q'$, or
- $P' = \mathsf{in}\ u'[\Phi'].Q'$ and $(\mathsf{mgu}(t, u') = \perp$ or $[\![\Phi'\mathsf{mgu}(t, u')]\!]_G = 0)$.

Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that $K_s = (\lfloor \mathsf{out}(t_s).P_s \rfloor_n \cup \{\lfloor \mathsf{in}\ u^s_j[\Phi^s_j].P^s_j \rfloor_{n_j} | j \in J\} \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ with $t_s\theta = t$, $P_s\theta = P$, $\mathcal{I}_s\theta = \mathcal{I}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{Q}_s\theta = \mathcal{Q}$ and for any $j \in J$, we have that $u^s_j\theta = u_j$, $\Phi^s_j\theta = \Phi_j$, and $P^s_j\theta = P_j$. Let

- $\mathcal{P}^s_{K,L} = \{\lfloor \mathsf{in}\ u^s_k[\Phi^s_k].P^s_k \rfloor_{n_k} \in \mathcal{Q}_s \mid (n_k, n) \in E\}$,
- $\mathcal{Q}'_s$ be such that $\mathcal{Q}_s = \mathcal{P}^s_{K,L} \uplus \mathcal{Q}'_s$,
- $K = \{k \mid \lfloor \mathsf{in}\ u^s_k[\Phi^s_k].P^s_k \rfloor_{n_k} \in \mathcal{P}^s_{K,L}$ and $\mathsf{mgu}(t_s\theta, u^s_k\theta) = \perp\}$,
- $L = \{l \mid \lfloor \mathsf{in}\ u^s_l[\Phi^s_l].P^s_l \rfloor_{n_l} \in \mathcal{P}^s_{K,L}$ and $\sigma'_l = \mathsf{mgu}(t_s\theta, u^s_l\theta)$ exists, and $\neg[\![(\Phi^s_l\theta)\sigma'_l]\!]_G = 1\}$.

We have that $\mathcal{P}^s_{K,L} = \{\lfloor \mathsf{in}\ u^s_k[\Phi^s_k].P^s_k \rfloor_{n_k} \in \mathcal{P}^s_{K,L} \mid k \in K \cup L\}$.
Let $K'_s = (\lfloor P_s \rfloor_n \cup \mathcal{P}^s_J \cup \mathcal{P}^s_{K,L} \cup \mathcal{Q}'_s; \mathcal{S}_s; \mathcal{I}'_s; \mathcal{C}')$ where:

- $\mathcal{P}^s_J = \{\lfloor P^s_j \rfloor_{n_j} \mid j \in J\}$,
- $\mathcal{C}' = \mathcal{C} \wedge \mathcal{C}_J \wedge \mathcal{C}_K \wedge \mathcal{C}_L$,
- $\mathcal{C}_J = \{t_s = u^s_j \wedge \Phi^s_j \mid j \in J\}$,
- $\mathcal{C}_K = \{\forall var(u_k) \smallsetminus rvar(\mathcal{C}) \,.\, t_s \neq u^s_k \mid k \in K\}$,
- $\mathcal{C}_L = \{t_s = u^s_l\alpha_l \wedge \neg\Phi^s_l\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u^s_l) \smallsetminus rvar(\mathcal{C})$ by fresh variables,
- $\mathcal{I}'_s = \mathcal{I}_s \cup \{t\}$ if $(n, n_I) \in E$ and $\mathcal{I}'_s = \mathcal{I}_s$ otherwise.

Clearly, we have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$. To conclude, it remains to show that there exists a substitution θ' that is a solution of $\mathcal{C}'$ and such that K'_c is the θ'-concretization of K'_s.

Let $\sigma_J = \bigcup_{j \in J} \sigma_j$. Let $\sigma_L = \bigcup_{l \in L} \sigma_l$ where $\sigma_l = \mathsf{mgu}(t, (u^s_l\theta)\alpha_l)$ for any $l \in L$ (Note that $\sigma'_l = \sigma_l \circ \alpha_l$). Let $\theta' = \theta \cup \sigma$ where $\sigma = \sigma_J \cup \sigma_L$. By hypothesis, we have that θ is a solution of $\mathcal{C}$. To show that θ' is a solution of $\mathcal{C}' = \mathcal{C} \wedge \mathcal{C}_J \wedge \mathcal{C}_K \wedge \mathcal{C}_L$, it remains to establish that:

- θ' is a solution of $\mathcal{C}_J$, i.e., $t_s\theta' = u^s_j\theta'$ for any $j \in J$. We have that $t_s\theta' = (t_s\theta)\sigma = t\sigma = t$ (since t is ground). Moreover, for any $j \in J$, we have that $(u^s_j\theta') = (u^s_j\theta)\sigma = (u^s_j\theta)\sigma_j = u_j\sigma_j = t$.
- θ' is a solution of $\mathcal{C}_K$, i.e., θ' satisfies $\forall \mathit{var}(u_k) \smallsetminus \mathit{rvar}(\mathcal{C}) \,.\, t_s \neq u^s_k$ for any $k \in K$. This is true since $\theta' = \theta \cup \sigma$ and $\mathsf{mgu}(t_s\theta, u^s_k\theta) = \bot$ for any $k \in K$.
- θ' is a solution of $\mathcal{C}_L$, i.e., $t_s\theta' = (u^s_l\alpha_l)\theta'$ and $[\![(\Phi^s_l\alpha_l)\theta']\!]_G = 0$ for any $l \in L$. We have that $t_s\theta' = (t_s\theta)\sigma = t\sigma = t$ and $(u^s_l\alpha_l)\theta' = ((u^s_l\alpha_l)\theta)\sigma_l = ((u^s_l\theta)\alpha_l)\sigma_l = t$ (by definition of σ_l). Moreover we have that $(\Phi^s_l\alpha_l)\theta' = ((\Phi^s_l\alpha_l)\theta)\sigma_l = ((\Phi^s_l\theta)\alpha_l)\sigma_l = (\Phi^s_l\theta)\sigma'_l$. Hence, we have that $[\![(\Phi^s_l\alpha_l)\theta']\!]_G = 0$ for any $l \in L$.

Lastly, it remains to verify that K'_c is the θ'-concretization of K'_s. Indeed, we have that:

- $P_s\theta' = (P_s\theta)\sigma = P_s\theta = P$,
- $P^s_j\theta' = (P^s_j\theta)\sigma = (P^s_j\theta)\sigma_j = P_j\sigma_j$ for any $j \in J$,
- $(\mathcal{P}^s_{K,L} \cup \mathcal{Q}'_s)\theta' = \mathcal{Q}_s\theta' = (\mathcal{Q}_s\theta)\sigma = \mathcal{Q}_s\theta = \mathcal{Q}$,
- $\mathcal{S}_s\theta' = (\mathcal{S}_s\theta)\sigma = \mathcal{S}_s\theta = \mathcal{S}$,
- $\mathcal{I}'_s\theta' = (\mathcal{I}'_s\theta)\sigma = \mathcal{I}'_s\theta = \mathcal{I}'$.

This allows us to conclude. □

Proposition 2 (soundness). *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ and $K'_s = (\mathcal{P}'_s; \mathcal{S}'_s; \mathcal{I}'_s; \mathcal{C}')$ be two ground symbolic configurations such that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$. Let θ' be a solution of $\mathcal{C}'$ and θ be the restriction of θ' to $\mathit{rvar}(\mathcal{C})$. Let K_c be the θ-concretization of K_s. There exists a ground concrete configuration K'_c such that:*

- *$K_c \rightarrow_{G,\mathcal{M}} K'_c$, and*
- *K'_c is the θ'-concretization of K'_s.*

Proof. As taking a transition can only add constraints to the constraint system $\mathcal{C}$ and since θ' is a solution of $\mathcal{C}'$, it also satisfies the constraints in $\mathcal{C}$. Furthermore, θ is the restriction of θ' to $\mathit{rvar}(\mathcal{C})$, so θ is a solution of $\mathcal{C}$. Let K_c be the θ-concretization of K_s and K'_c be the θ'-concretization of K'_s. It remains to show that $K_c \rightarrow_{G,\mathcal{M}} K'_c$. We distinguish several cases, depending on the rule involved in the transition $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$.

- Rule $\textsc{Par}_s$. We have that:

$$(\lfloor P_1^s | P_2^s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_1^s \rfloor_n \cup \lfloor P_2^s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K'_c = (\lfloor P_1^s\theta' \rfloor_n \cup \lfloor P_2^s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$,
  - $K_c = (\lfloor P_1^s\theta | P_2^s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

  Since $\mathcal{C}' = \mathcal{C}$, we have that $\theta' = \theta$, and thus $K_c \rightarrow_{G,\mathcal{M}} K'_c$ (by the $\textsc{Par}$ rule).

- Rule $\textsc{Repl}_s$. We have that:

$$(\lfloor !P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s\alpha_s \rfloor_n \cup \lfloor !P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

  where α_s is a renaming of the bound variables of P_s that are not in $rvar(\mathcal{C})$. Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K'_c = (\lfloor (P_s\alpha_s)\theta' \rfloor_n \cup \lfloor !P_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$,
  - $K_c = (\lfloor !P_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

  Since $\mathcal{C}' = \mathcal{C}$, we have that $\theta' = \theta$. To show that $K_c \rightarrow_{G,\mathcal{M}} K'_c$ (by the $\textsc{Repl}$ rule), it remains to prove that:

  - $(P_s\theta)\alpha_s = (P_s\alpha_s)\theta$. This equality comes from the fact $dom(\theta) \cap dom(\alpha_s) = \emptyset$.
  - α_s is a renaming of $bv(P_s\theta)$. This is due to the fact that α_s is renaming of the bound variables of P_s that are not in $rvar(\mathcal{C})$ and $dom(\theta) = rvar(\mathcal{C})$.

- Rule $\textsc{New}_s$. We have that:

$$(\lfloor \mathsf{new}\ m.P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s\{^{m'}/_m\} \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

  where m' is a fresh name.

  As in the previous cases, we have that K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s). Moreover, since $\mathcal{C}' = \mathcal{C}$, we have that $\theta' = \theta$. Hence, we have that:

  - $K'_c = (\lfloor ((P_s\{^{m'}/_m\})\theta) \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
  - $K_c = (\lfloor \mathsf{new}\ m.P_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

  As in the previous case, since m' is a fresh name and $(P_s\theta)\{^{m'}/_m\} = (P_s\{^{m'}/_m\})\theta$, we have that $K_c \rightarrow_{G,\mathcal{M}} K'_c$ (by the $\textsc{New}$ rule).

- Rule $\textsc{Store}_s$. We have that:

$$(\lfloor \mathsf{store}(t_s).P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K_c = (\lfloor \mathsf{store}(t_s\theta).(P_s\theta) \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
  - $K'_c = (\lfloor P_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \lfloor t_s\theta' \rfloor_n \cup \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$.

  Since $\mathcal{C}' = \mathcal{C}$, we have that $\theta' = \theta$, and thus $K_c \rightarrow_{G,\mathcal{M}} K'_c$ (by the $\textsc{Store}$ rule).

- Rule $\textsc{Read-Then}_s$. We have that:

$$\begin{array}{l}(\lfloor \mathsf{read}\ u_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \\ \qquad \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \lfloor t_s \rfloor_n \cup \mathcal{S}_s; \mathcal{I}_s; \mathcal{C} \wedge t_s = u_s)\end{array}$$

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K'_c = (\lfloor P_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \lfloor t_s\theta' \rfloor_n \cup \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$,
  - $K_c = (\lfloor \mathsf{read}\ u_s\theta\ \mathsf{then}\ P_s\theta\ \mathsf{else}\ Q_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \lfloor t_s\theta \rfloor_n \cup \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

  Since θ' is a solution of $\mathcal{C}'$, we have $t_s\theta' = u_s\theta'$. Moreover, since θ is the restriction of θ' to $rvar(\mathcal{C})$, we have that $\theta' = \theta \cup \sigma$ for some substitution σ. We have that $(t_s\theta)\sigma = (u_s\theta)\sigma$. Since $t_s\theta$ is a ground term, actually we have that $\sigma = \mathsf{mgu}(t_s\theta, u_s\theta)$. Hence, we have that:

$$K_c \rightarrow_{G,\mathcal{M}} (\lfloor (P_s\theta)\sigma \rfloor_n \cup \mathcal{Q}_s\theta; \lfloor t_s\theta \rfloor_n \cup \mathcal{S}_s\theta; \mathcal{I}_s\theta)$$

  by the $\textsc{Read-Then}$ rule. Since K_s is a ground symbolic configuration, we know that $var(\mathcal{I}_s) \cup fv(\mathcal{Q}_s) \cup var(\lfloor t_s \rfloor_n \cup \mathcal{S}_s) \subseteq dom(\theta)$, and thus, $K_c \rightarrow_{G,\mathcal{M}} K'_c$ (by the $\textsc{Read-Then}$ rule).

- Rule $\textsc{Read-Else}_s$. We have that:

$$\begin{array}{l}K_s = (\lfloor \mathsf{read}\ u_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \\ \qquad\qquad \rightarrow^s_{G,\mathcal{M}} (\lfloor Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C} \wedge \mathsf{Eq}) = K'_s\end{array}$$

  where $\mathsf{Eq} = \{\forall var(u_s) \smallsetminus rvar(\mathcal{C}) \,.\, t_s \neq u_s \mid \lfloor t_s \rfloor_n \in \mathcal{S}_s\}$.

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K_c = (\lfloor \mathsf{read}\ u_s\theta\ \mathsf{then}\ P_s\theta\ \mathsf{else}\ Q_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
  - $K'_c = (\lfloor Q_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$.

Since $\mathit{rvar}(\mathcal{C}') = \mathit{rvar}(\mathcal{C})$, we have that $\theta' = \theta$. Moreover, since θ is a solution of $\mathcal{C}'$, we have that $u_s\theta$ is not unifiable with $t_s\theta$ for any $\lfloor t_s \rfloor_n \in \mathcal{S}_s$. In other words, $\mathsf{mgu}(u_s\theta, t) = \bot$ for any t such that $\lfloor t \rfloor_n \in \mathcal{S}_s\theta$. Hence, we have that $K_c \rightarrow_{G,\mathcal{M}} K'_c$ by the READ-ELSE rule.

- Rule IF-THEN$_s$. We have that:

$$\begin{aligned} K_s = (\lfloor \mathsf{if}\ \Phi_s\ \mathsf{then}\ P_s\ \mathsf{else}\ Q_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \\ \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C} \wedge \Phi_s) = K'_s \end{aligned}$$

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K_c = (\lfloor \mathsf{if}\ \Phi_s\theta\ \mathsf{then}\ P_s\theta\ \mathsf{else}\ Q_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
  - $K'_c = (\lfloor P_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

  Since $\mathit{rvar}(\mathcal{C}') = \mathit{rvar}(\mathcal{C})$, we have that $\theta' = \theta$. Moreover, since θ is a solution of $\mathcal{C}'$, we have that $[\![\Phi_s\theta]\!] = 1$. Hence, we have that $K_c \rightarrow_{G,\mathcal{M}} K'_c$ by the IF-THEN rule.

- Rule IF-ELSE$_s$. This case is similar to the previous one.

- Rule IN$_s$. We have that:

$$(\lfloor \mathsf{in}\ u_s[\Phi_s].P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}')$$

  where $\mathcal{C}' = \mathcal{C} \wedge \mathcal{I}_s \Vdash u_s \wedge \Phi_s$ and $(n_I, n) \in E$ for some $n_I \in \mathcal{M}$.

  Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

  - $K_c = (\lfloor \mathsf{in}\ u_s\theta[\Phi_s\theta].P_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
  - $K'_c = (\lfloor P_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$.

  Since θ' is a solution of $\mathcal{C}'$, we have that $\mathcal{I}_s\theta' \vdash u_s\theta'$ and $[\![\Phi_s\theta']\!]_G = 1$. Moreover, we know that there exists a substitution σ such that $\theta' = \theta \cup \sigma$ with $\mathit{dom}(\sigma) = \mathit{var}(u_s\theta)$. We have that $u_s\theta'$ is a ground term, and thus $\mathsf{mgu}(u_s\theta', u_s\theta) = \sigma$. Lastly, since K_s is a ground symbolic configuration, we have that $\mathit{fv}(\mathcal{Q}_s) \cup \mathit{var}(\mathcal{S}_s) \cup \mathit{var}(\mathcal{I}_s) \subseteq \mathit{dom}(\theta)$, and thus $\mathcal{Q}_s\theta' = \mathcal{Q}_s\theta$, $\mathcal{S}_s\theta' = \mathcal{S}_s\theta$, and $\mathcal{I}_s\theta' = \mathcal{I}_s\theta$. Hence, we have that $K_c \rightarrow_{G,\mathcal{M}} K'_c$ by the IN rule.

- Rule COMM$_s$. We have that:

$$(\mathcal{P}^s_I \cup \lfloor \mathsf{out}(t_s).P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\mathcal{P}^s_J \cup \mathcal{P}^s_{K,L} \cup \lfloor P_s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}'_s; \mathcal{C}')$$

  where:

  - $\mathcal{P}^s_I = \{\lfloor \mathsf{in}\ u^s_i[\Phi^s_i].P^s_i \rfloor_{n_i} \mid (n_i, n) \in E, i \in I\}$,

- $I = J \uplus K \uplus L$,
- $\mathcal{P}^s_J = \{\lfloor P^s_j \rfloor_{n_j} |\ j \in J\}$,
- $\mathcal{P}^s_{K,L} = \{\lfloor \mathsf{in}\ u_k[\Phi^s_k].P^s_k \rfloor_{n_k} |\ k \in K \uplus L\}$,
- $\mathcal{C}' = \mathcal{C}\ \wedge\ \mathcal{C}_J\ \wedge\ \mathcal{C}_K\ \wedge\ \mathcal{C}_L$,
- $\mathcal{C}_J = \{t_s = u^s_j\ \wedge\ \Phi^s_j \mid j \in J\}$,
- $\mathcal{C}_K = \{\forall y \in \mathit{var}(u^s_k) \smallsetminus \mathit{rvar}(\mathcal{C})\,.\, t_s \neq u^s_k \mid k \in K\}$,
- $\mathcal{C}_L = \{t_s = u^s_l \alpha_l \wedge \neg\Phi^s_l \alpha_l \mid l \in L\}$ where α_l is a renaming of $\mathit{var}(u^s_l) \smallsetminus \mathit{rvar}(\mathcal{C})$ by fresh variables.
- $\mathcal{I}'_s = \mathcal{I}_s \cup \{t_s\}$ if $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$ and $\mathcal{I}'_s = \mathcal{I}_s$ otherwise.

Moreover, $\lfloor Q_s \rfloor_{n'} \in \mathcal{Q}_s$ implies that $(n, n') \notin E$ or Q_s is not of the form $\mathsf{in}\ u'_s[\Phi'_s].Q'_s$. We have also that $(n_i, n) \in E$ for every $i \in I$.

Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:

- $K_c = (\mathcal{P}^s_I\theta \cup \lfloor \mathsf{out}(t_s\theta).P_s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$,
- $K'_c = (\mathcal{P}^s_J\theta' \cup \mathcal{P}^s_{K,L}\theta' \cup \lfloor P_s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}'_s\theta')$.

To conclude, it remains to show that $K_c \rightarrow_{G,\mathcal{M}} K'_c$. First, by using the fact that K_s is a ground symbolic configuration, we have that:

- $\mathcal{S}_s\theta' = \mathcal{S}_s\theta$,
- if $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$ then $\mathcal{I}'_s\theta' = \mathcal{I}_s\theta' \cup \{t_s\theta'\} = \mathcal{I}_s\theta \cup \{t_s\theta\}$. Otherwise, we have that $\mathcal{I}'_s\theta' = \mathcal{I}_s\theta$.
- $P_s\theta' = P_s\theta$, $\mathcal{Q}_s\theta' = \mathcal{Q}_s\theta$, and $\mathcal{P}^s_{K,L}\theta' = \mathcal{P}^s_{K,L}\theta$ (thanks to the renaming α_l).

Note also that the processes in $\mathcal{Q}_s\theta$ are not of the right form to evolve by receiving a message from the node n. Thus, to show that $K_c \rightarrow_G K'_c$, it remains to prove that $J = J'$ where

$$J' = \left\{\ i \ \middle|\ \begin{array}{l} \lfloor \mathsf{in}\ u^s_i[\Phi^s_i].P^s_i \rfloor_{n_i} \in \mathcal{P}^s_I, \\ \overline{\sigma}_j = \mathsf{mgu}(t\theta, u^s_i\theta) \text{ exists },\\ (n, n_i) \in E,\ [\![(\Phi^s_i\theta)\overline{\sigma}_i]\!]_G = 1 \end{array} \right\}.$$

We prove the two inclusions separately. Let σ be the substitution such that $\theta' = \theta \cup \sigma$. For any $i \in J$, we denote by σ_i the restriction of σ to the variables $\mathit{var}(u^s_i)$, whereas for any $i \in L$, σ_i is the restriction of σ to the variables $\mathit{var}(u^s_i\alpha_i)$. Lastly, $\mathit{dom}(\sigma_i) = \emptyset$ when $i \in K$. Hence, we have that $\sigma = \bigcup_{i \in I} \sigma_i$.

First, we show that $J \subseteq J'$. Let $i \in J$. We know that $\lfloor \mathsf{in}\ u^s_i[\Phi^s_i].P^s_i \rfloor_{n_i} \in \mathcal{P}^s_I$, $(n, n_i) \in E$, and since θ' is a solution of $\mathcal{C}'$, we have that $t\theta' = u^s_i\theta'$ and $[\![\Phi^s_i\theta']\!]_G = 1$. Since $t\theta' = u^s_i\theta'$ and θ is the restriction of θ on the variables $\mathit{rvar}(\mathcal{C})$, we deduce that $\overline{\sigma}_i = \mathsf{mgu}(t\theta, u^s_i\theta)$ exists. Since $t\theta$ is a ground term, we have that $\overline{\sigma}_i = \sigma_i$. Lastly, we have that $\Phi_i^s\theta' = (\Phi_i^s\theta)\sigma = (\Phi_i^s\theta)\sigma_i = (\Phi_i^s\theta)\overline{\sigma}_i$. This allows us to conclude that $i \in J'$.

Now, we show that $J' \subseteq J$. Let $i \in J'$. By definition of J', we know that $\lfloor \mathsf{in}\ u_i^s[\Phi_i^s].P_i^s \rfloor_{n_i} \in \mathcal{P}_I^s$, $(n, n_i) \in E$, $\overline{\sigma}_i = \mathsf{mgu}(t\theta, u_i^s\theta)$ exists, and $[\![(\Phi_i^s\theta)\overline{\sigma}_i]\!]_G = 1$. Hence, we have that $i \in I$. In order to conclude that $i \in J$, it is sufficient to show that $i \notin K$ and $i \notin L$.

1. $i \notin K$. By contradiction, assume that $i \in K$. Since θ' is a solution of $\mathcal{C}'$, we have that θ' satisfies the constraint $\forall y \in var(u_i^s) \smallsetminus rvar(\mathcal{C}) \,.\, t_s \neq u_i^s$. This implies that $t_s\theta$ and $u_i^s\theta$ are not unifiable This is impossible since we know that $\overline{\sigma}_i = \mathsf{mgu}(t\theta, u_i^s\theta)$ exists. Contradiction. Hence, we deduce that $i \notin K$.
2. $i \notin L$. By contradiction, assume that $i \in L$. Since θ' is a solution of $\mathcal{C}'$, we have that $t = (u_i^s\alpha_i)\theta'$ and $[\![(\Phi_i^s\alpha_i)\theta']\!]_G = 0$. Actually, we have that:
$$(u_i^s\alpha_i)\theta' = ((u_i^s\alpha_i)\theta)\sigma_i = ((u_i^s\theta)\alpha_i)\sigma_i.$$
Hence, we have that $\overline{\sigma}_i = \alpha_i\sigma_i$. We have also that:
$$(\Phi_i^s\alpha_i)\theta' = ((\Phi_i^s\alpha_i)\theta)\sigma_i = ((\Phi_i^s\theta)\alpha_i)\sigma_i.$$
We deduce that $[\![(\Phi_i^s\theta)\overline{\sigma}_i]\!]_G = 0$. Contradiction. Hence, we have that $i \notin L$.

This allows us to conclude. □

Appendix B. Turning constraint systems into solved forms

Lemma 2. *Let T be a set of terms that contains at least one term of each sort, u_0 be a term, and $\mathcal{I}$ be a set of names $St(T \cup \{u_0\}) \cap \mathcal{I} = \emptyset$. If $T \cup \mathcal{I} \vdash u_0$, then we have that $T \vdash u_0$.*

Proof. Let π be a proof of $T \cup \mathcal{I} \vdash u_0$. Let δ be the operation replacing each name in $\mathcal{I}$ by a term in T of the same sort. The operation δ is extended to terms and substitutions in the obvious way. We show by induction on π that there exists a proof π' of $(T \cup \mathcal{I})\delta \vdash u_0\delta$. Since $St(T \cup \{u_0\}) \cap \mathcal{I} = \emptyset$, and $\mathcal{I}\delta \subseteq T$, this allows us to conclude that $T \vdash u_0$.

Base case: π is reduced to the application of an axiom rule. In such a case, we have that $u_0 \in T \cup \mathcal{I}$ and thus there exists a proof π' of $(T \cup \mathcal{I})\delta \vdash u_0\delta$ that is also reduced to an axiom rule.

Induction step: π ends with an instance of another inference rule. Suppose that the last inference rule R applied in π is of the form
$$\frac{T \cup \mathcal{I} \vdash v_1 \quad \dots \quad T \cup \mathcal{I} \vdash v_n}{T \cup \mathcal{I} \vdash v_0}$$

Let $\pi_1, \ldots, \pi_n$ be the direct subproofs of π. For each $i \in \{1, \ldots, n\}$, we have that the root of π_i is labeled with $T \cup \mathcal{I} \vdash u_i$ for some term u_i. Moreover, we know that there exists a substitution θ such that $u_i = v_i\theta$ for each $i \in \{0, 1, \ldots, n\}$. Thanks to our induction hypothesis, we have that there exist $\pi'_1, \ldots, \pi'_n$ such that π'_i is a proof of $(T \cup \mathcal{I})\delta \vdash u_i\delta$. Actually, by inspection of our inference system, it is easy to see that $u_i\delta = (v_i\theta)\delta = v_i(\theta\delta)$ for each $i \in \{0, 1, \ldots, n\}$. Hence, we can apply the rule R on $\pi'_1, \ldots, \pi'_n$. Let π' be the resulting proof tree. Since $v_0\theta' = v_0(\theta\delta) = (v_0\theta)\delta = u_0\delta$. We have that π' is a proof of $(T \cup \mathcal{I})\delta \vdash u_0\delta$. □

Proposition 3. *Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system.*

1. *If $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ then $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \overline{\mathcal{C}'}^{\mathcal{I}}$ by applying the same simplification rule, and $(\mathcal{C}', \mathcal{I})$ is a special constraint system.*
2. *If $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \mathcal{C}'_s$, then there exists $\mathcal{C}'$ such that $\mathcal{C}'_s = \overline{\mathcal{C}'}^{\mathcal{I}}$ and $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ by applying the same simplification rule. Furthermore, we have that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.*

Proof. We show the two items separately.

1). We reason by case study over the simplification rule used in $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$.

Rule R_1. In such a case, we have that $\mathcal{C} = \mathcal{C}' \wedge T \Vdash u$ and $T \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u$. Consequently, $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}'}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$. Furthermore, we have that

$$\{x \mid (T' \Vdash x) \in \overline{\mathcal{C}}^{\mathcal{I}}, T' \subsetneq T \cup \mathcal{I}\} = \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\}.$$

Hence we have that $T \cup \mathcal{I} \cup \{x \mid (T' \Vdash x) \in \overline{\mathcal{C}}^{\mathcal{I}}, T' \subsetneq T \cup \mathcal{I}\} \vdash u$, and thus $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \overline{\mathcal{C}'}^{\mathcal{I}}$ with R_1. Lastly, since $St(\mathcal{C}') \subseteq St(\mathcal{C})$, we easily deduce that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_2. In such a case, we have that $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$, $\mathcal{C}' = \mathcal{C}_0\sigma \wedge T\sigma \Vdash u\sigma$ for some $\mathcal{C}_0$, T, u, and σ where $\sigma = \mathsf{mgu}(t, v)$ with $t \in St(T)$, $v \in St(u)$, $t \neq v$, and t, v not variables. Hence, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$ and $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \overline{\mathcal{C}_0}^{\mathcal{I}}\sigma \wedge (T \cup \mathcal{I})\sigma \Vdash u\sigma = \overline{\mathcal{C}'}^{\mathcal{I}}$ using R_2. Lastly, as $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, for every $x \in dom(\sigma)$, we have that $St(x\sigma) \cap \mathcal{I} = \emptyset$. Hence, we deduce that $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$, and so $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_3 *and* R'_3. These two cases are similar to the previous one.

Rule R_f. In such a case, we have that $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash \mathsf{f}(u, v)$, and $\mathcal{C}' = \mathcal{C}_0 \wedge T \Vdash u \wedge T \Vdash v$ for some $\mathcal{C}_0$, T, f, u, and v. Hence, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash \mathsf{f}(u, v)$, and thus $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u \wedge T \cup \mathcal{I} \Vdash v$ using R_f. Lastly, as $St(\mathcal{C}') \subseteq St(\mathcal{C})$, we easily deduce that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_4. In such a case, we have that $\mathcal{C}' = \bot$, and $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$, for some $\mathcal{C}_0$, T and u such that $var(T \cup \{u\}) = \emptyset$ and $T \not\vdash u$. Hence, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$ and $var(T \cup \mathcal{I} \cup \{u\}) = \emptyset$. Thanks to Lemma 2 and relying on the fact that $(\mathcal{C}, \mathcal{I})$ is a special constraint system, we also have that

$T \cup \mathcal{I} \not\vdash u$. Thus, we have that $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow \bot = \overline{\bot}^{\mathcal{I}}$ using R_4 and $(\bot, \mathcal{I})$ is a special constraint system.

2) We reason by case study over the simplification rule used in $\overline{\mathcal{C}}^{\mathcal{I}} \rightsquigarrow_\sigma \mathcal{C}'_s$.

Rule R_1. In such a case, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}'_s \wedge T \Vdash u$ and $T \cup \{x \mid (T' \Vdash x) \in \overline{\mathcal{C}}^{\mathcal{I}}, T' \subsetneq T\} \vdash u$ for some $\mathcal{C}'_s$, T, and u. By definition of $\overline{\mathcal{C}}^{\mathcal{I}}$, we deduce that there exists $\mathcal{C}'$ such that $\mathcal{C} = \mathcal{C}' \wedge T' \Vdash u$, with $\overline{\mathcal{C}'}^{\mathcal{I}} = \mathcal{C}'_s$, and $T = T' \cup \mathcal{I}$. Furthermore, we have that

$$\{x \mid (T' \Vdash x) \in \overline{\mathcal{C}}^{\mathcal{I}}, T' \subsetneq T \cup \mathcal{I}\} = \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\}.$$

Consequently, we have that $T' \cup \mathcal{I} \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u$. As $(\mathcal{C}, \mathcal{I})$ is a special constraint system, we have that $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, so we can apply Lemma 2, we obtain that $T' \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u$. We obtain that $\mathcal{C} \rightsquigarrow \mathcal{C}'$ using R_1. Lastly, we have that $St(\mathcal{C}') \subseteq St(\mathcal{C})$, and thus we easily deduce that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_2. In such a case, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}^0_s \wedge T_s \Vdash u$, $\mathcal{C}'_s = \mathcal{C}^0_s\sigma \wedge T_s\sigma \Vdash u\sigma$ for some $\mathcal{C}^0_s$, T_s, u, and σ where $\sigma = \mathsf{mgu}(t, v)$, $t \in St(T_s)$, $v \in St(u)$, $t \neq v$, t, v not variables. In such a case, we know that there exist $\mathcal{C}^0$ and T such that $\mathcal{C} = \mathcal{C}^0 \wedge T \Vdash u$, $T_s = T \cup \mathcal{I}$, and $\mathcal{C}^0_s = \overline{\mathcal{C}^0}^{\mathcal{I}}$. In order to apply R_2 to $\mathcal{C}$, it remains to show that $t \in St(T)$. Since, we already know that $t \in St(T_s)$, we have to show that $t \notin St(\mathcal{I}) = \mathcal{I}$. By contradiction, suppose that $t \in \mathcal{I}$. In such a case, in order to have $t\sigma = v\sigma$, v has to be a variable. This yields to a contradiction. Hence, we have that $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}^0\sigma \wedge T\sigma \Vdash u\sigma \overset{\mathsf{def}}{=} \mathcal{C}'$ and $\overline{\mathcal{C}'}^{\mathcal{I}} = \mathcal{C}'_s$ Lastly, as $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, for every $x \in dom(\sigma)$, we have that $St(x\sigma) \cap \mathcal{I} = \emptyset$. This allows us to conclude that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_3*,* R'_3. These two cases are similar to the previous one.

Rule R_f. In such a case, we have that $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}^0_s \wedge T_s \Vdash \mathsf{f}(u, v)$, and $\mathcal{C}'_s = \mathcal{C}^0_s \wedge T_s \Vdash u \wedge T_s \Vdash v$ for some $\mathcal{C}^0_s$, T_s, f, u, and v. By definition of $\overline{\mathcal{C}}^{\mathcal{I}}$, there exists $\mathcal{C}^0$, and T such that $\mathcal{C} = \mathcal{C}^0 \wedge T \Vdash \mathsf{f}(u, v)$, $\mathcal{C}^0_s = \overline{\mathcal{C}^0}^{\mathcal{I}}$, and $T_s = T \cup \mathcal{I}$. Consequently, using R_f, we have that

$$\mathcal{C} \rightsquigarrow \mathcal{C}^0 \wedge T \Vdash u \wedge T \Vdash v \overset{\mathsf{def}}{=} \mathcal{C}'.$$

Furthermore, we have that $\overline{\mathcal{C}'}^{\mathcal{I}} = \mathcal{C}'_s$. Lastly, we have that $St(\mathcal{C}') \subseteq St(\mathcal{C})$. This allows us to deduce that $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

Rule R_4. In such a case, we have that $\mathcal{C}'_s = \bot$, and $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}^0_s \wedge T_s \Vdash u$, for some $\mathcal{C}^0_s$, T_s, and u such that $var(T_s \cup \{u\}) = \emptyset$ and $T_s \not\vdash u$. By definition of $\overline{\mathcal{C}}^{\mathcal{I}}$, there exists $\mathcal{C}^0$, and T such that $\mathcal{C} = \mathcal{C}^0 \wedge T \Vdash u$, $\mathcal{C}^0_s = \overline{\mathcal{C}^0}^{\mathcal{I}}$, and $T_s = T \cup \mathcal{I}$. Consequently, we have that $var(T \cup \{u\}) = \emptyset$ and $T \not\vdash u$. Hence, we can apply R_4, and we obtain that $\mathcal{C} \rightsquigarrow \bot \overset{\mathsf{def}}{=} \mathcal{C}'$. Lastly, it is easy to see that $(\mathcal{C}', \mathcal{I})$ is a special constraint system, and $\overline{\mathcal{C}'}^{\mathcal{I}} = \mathcal{C}'_s$. $\square$

DELETE	$P \cup \{s = s\} \quad \Longrightarrow \quad P$
DEC.	$P \cup \{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)\} \Longrightarrow P \cup \{s_1 = t_1, \ldots, s_n = t_n\}$
CONF.	$P \cup \{f(s_1, \ldots, s_n) = g(t_1, \ldots, t_k)\} \Longrightarrow \bot \quad$ if $f \neq g$
COAL.	$P \cup \{x = y\} \Longrightarrow P\{x \mapsto y\} \cup \{x = y\}$ if $x, y \in var(P)$ and $x \neq y$
CHECK	$P \cup \{x_1 = s_1[x_2], \ldots, x_n = s_n[x_1]\} \Longrightarrow \bot$ if $s_i \notin \mathcal{X}$ for some $i \in [1 \ldots n]$
MERGE	$P \cup \{x = s, x = t\} \Longrightarrow P \cup \{x = s, s = t\}$if $0 < \|s\| \leq \|t\|$

Figure B.8: Rules for DAG syntactic unification

Lemma 4. *Let T be a set of terms and P be a set of equations between terms in $St(T)$ with $\sigma = \mathsf{mgu}(P)$. We have that $St(T\sigma) \subseteq St(T)\sigma$.*

Proof. We use the rules for DAG syntactic unification given in Figure B.8. Applying these rules on P results in a set of equations $P' = \{x_1 = t_1, \ldots, x_n = t_n\}$ in DAG solved form (see [22]). By definition of a DAG solved form, we have that:

- $x_i \neq x_j$ for all $1 \leq i < j \leq n$,
- $x_i \notin var(t_j)$ for all $1 \leq i < j \leq n$.

Let $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. By inspection of the rules in Figure B.8, we can show by induction on the length of the derivation from P to P' that $St(P')\sigma \subseteq St(P)\sigma$. Since $St(P) \subseteq St(T)$, we easily deduce that $St(t_i)\sigma \subseteq St(T)\sigma$ for every $1 \leq i \leq n$.

Let $u \in St(T\sigma)$, we show that there exists $t \in St(T)$ such that $u = t\sigma$. Either there exists v a subterm of T such that $u = v\sigma$, and we conclude, or there exists $x_i \in dom(\sigma)$ such that u is a subterm of $x_i\sigma$. In that case, let $i_0 = \mathsf{max}\{i \mid u \in St(x_i\sigma)\}$.

- Either $u \in St(t_{i_0})\sigma \subseteq St(T)\sigma$, and we conclude.
- Or $u \in St(x\sigma)$ for some $x \in var(t_{i_0}) \cap dom(\sigma)$. By definition of a DAG solved form, we have that $var(t_{i_0}) \cap dom(\sigma) \subseteq \{x_{i_0+1}, \ldots, x_n\}$. Hence, we have that $u \in St(x_j\sigma)$ for some $j > i_0$. This yields to a contradiction. □

Appendix C. Bounding the size of minimal solutions for solved forms

In this appendix, we give a full proof of Proposition 4 (see Appendix C.2). We also explain how to adapt the proof of Proposition 4 to prove Proposition 5 (see Appendix C.3).

Appendix C.1. A preliminary result

Let S be a set, we denote by $\#S$ the cardinal of S. Let u be a term. We denote by $|u|_d$ the maximal depth of a variable in u. The lemma below is useful to bound the depth of variables after application of a substitution.

Lemma 5. *Let T be a set of terms, P be a set of equations between terms in T and $\sigma = \mathsf{mgu}(P)$. For every variable $x \in St(T)$, we have that:*

$$|x\sigma|_d \leq \#dom(\sigma) \cdot \max\{|t|_d \mid t \in T\}.$$

Proof. We use the rules for DAG syntactic unification given in Figure B.8. Applying these rules on P results in a set of equations representing a most general unifier of P in DAG solved form (see [22]): $\sigma = \{x_1 = t_1, \ldots, x_n = t_n\}$. By definition of a DAG solved form, we have that:

- $x_i \neq x_j$ for all $1 \leq i < j \leq n$,
- $x_i \notin var(t_j)$ for all $1 \leq i < j \leq n$.

Hence, we have that $|x\sigma|_d < |t_1|_d + \ldots + |t_n|_d$. Furthermore, by inspection of the rules, we can see that each t_i is a subterm (modulo a non-bijective renaming of the variables) of T. For every $1 \leq i \leq n$, we have that $|t_i|_d \leq \max\{|t|_d \mid t \in T\}$. Since $n = \#dom(\sigma)$, we deduce that $|x\sigma|_d < \#dom(\sigma) \cdot \max\{|t|_d \mid t \in T\}$. $\square$

Appendix C.2. Case of a fixed topology

We first consider the case where the topology is fixed.

Definition 8 (extracted list). *An* extracted list from *a list $l = [a_1; \ldots ; a_n]$ is a list $[a_{i_1}; \ldots ; a_{i_k}]$ such that $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n$ with $0 \leq k \leq n$.*

We prove that we can find a solution in which lists are polynomially bounded. In the case where the network topology is fixed, the bound depends on the size of the graph, i.e., its number of edges. Let l be a list, we denote by $|l|_\ell$ the length of l.

Proposition 4. *Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, then there exists a solution σ of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G that is polynomially bounded in the size of Φ_1, Φ_2 and E.*

Proof. We write $\Phi_2 = \bigwedge_n \forall Y_n . u_n \neq v_n$, and

$$\Phi_1 = \bigwedge_i \pm_i \mathsf{check}(a_i, b_i) \wedge \bigwedge_j \bigwedge_k^{p_j} \pm_{j_k} \mathsf{checkl}(c_{j_k}, l_j) \wedge \bigwedge_l \pm_l \mathsf{route}(r_l) \wedge \bigwedge_h \pm_h \mathsf{loop}(p_h)$$

with $\pm \in \{+,-\}$, a_i, b_i, c_{j_k} are of sort loc, l_j, r_l, p_h are terms of sort lists, u_n, v_n are terms and Y_n are sets of variables.

In the following, we denote:

- N the maximal depth of a variable in the disequality constraints,
- k the maximal number of variables in a disequality constraint,
- C the number of constraints $\pm\mathsf{checkl}$ in Φ_1,
- L the number of constraints loop in Φ_1,
- R the number of constraints $\neg\mathsf{route}$ in Φ_1, and
- $M = \mathsf{max}(kN + 3C + L + R + 3, \#E)$.

We show that, if there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G, then there exists a substitution σ such that σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, and

- for all variables x of sort lists, $|x\sigma|_\ell \leq M$, and
- $x\sigma \in \mathcal{I} \cup \mathcal{N}_{\mathsf{loc}}$ otherwise.

First, we have that $x\sigma \in \mathcal{N}_{\mathsf{loc}}$ when x is a variable of sort loc. Moreover, thanks to Lemma 3, we can assume that $x\sigma \in \mathcal{I}$ when x is a variable that is neither of sort loc nor of type lists. Now, among these solutions, consider a smallest solution σ of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, where the size of a solution σ is given by $|\sigma| = |x_1\sigma|_\ell + \ldots + |x_n\sigma|_\ell$ where $x_1, \ldots, x_n$ are the variables of sort lists that occur in $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$.

If $|x\sigma|_\ell \leq M$ for all variables x of sort lists, then we easily conclude. Otherwise, there exists a variable x_ℓ of sort lists such that the length of $x_\ell\sigma$ is greater than M. We are going to show that we can build σ' from σ, solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, smaller than σ. More specifically, we build σ' such that for all $x \neq x_\ell, x\sigma' = x\sigma$, and $|x_\ell\sigma'|_\ell \leq M < |x_\ell\sigma|_\ell$.

We build $x_\ell\sigma'$ by marking the names we want to keep in the list in the following manner:

$$x_\ell\sigma = \begin{array}{|c|c|c|c|c|c|}\hline a_1 & a_2 & \ldots & a_{kN} & \ldots & a_P \\ \hline\end{array}$$

We mark the first kN names in the list:

$$\begin{array}{|c|c|c|c|c|}\hline \overline{a_1} & \overline{a_2} & \ldots & \overline{a_{kN}} & \ldots \\ \hline\end{array}$$

We then mark the other names we want to keep in the list in the following way:

Case of a checkl *that occurs positively.*

If there exists c_{j_k} such that $\mathsf{checkl}(c_{j_k}, l_j)$ is a constraint that occurs positively in Φ_1, i.e., $\pm_{j_k} = +$, and $x_\ell \in \mathit{var}(l_j)$. Assume that $l_j = d_1 :: \ldots :: d_p :: x_\ell$. As σ is a solution for Φ_1, in particular we know that $c = c_{j_k}\sigma$ appears exactly once in $l_j\sigma$, and for any l' sublist of $l_j\sigma$,

- if $l' = a :: c :: l_1$, then $(a, c) \in E$.
- if $l' = c :: b :: l_1$, then $(b, c) \in E$.

Since c appears exactly once in $l_j\sigma$, either there exists n such that $c = d_n\sigma$, or there exists m such that $c = a_m$. In the first case and if $n = p$, we mark a_1. In the second case, we mark a_m, a_{m-1}(if $m > 1$) and a_{m+1}(if $m < P$). Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the checkl condition for graph G.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \dots & \overline{a_{m-1}} & \overline{a_m} & \overline{a_{m+1}} & \dots & a_P \\ \hline\end{array}$$

Case of a checkl *that occurs negatively.*

If there exists c_{j_k} such that $\mathsf{checkl}(c_{j_k}, l_j)$ is a constraint that occurs negatively in Φ_1, i.e., $\pm_{j_k} = -$, and $x_\ell \in var(l_j)$. Assume that $l_j = b_1 :: \ldots :: b_p :: x_\ell$. As σ is a solution for Φ_1, we can have three different cases depending on $c = c_{j_k}\sigma$:

- c does not appear in $l_j\sigma$: for every n, m, $b_n\sigma \neq c$ and $a_m \neq c$. In that case, we mark nothing.
- c appears at least twice in $l_j\sigma$. In that case, we choose two occurrences of c and we mark them when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \dots & \overline{c} & \dots & \overline{c} & \dots & a_P \\ \hline\end{array}$$

- c appears once in $l_j\sigma$, but one of his neighbors in the list is not a neighbor of it in the graph. For example, $c = a_i$ and $(a_i, a_{i+1}) \notin E$. We mark c and this false neighbor when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|}\hline a_1 & \dots & \overline{a_i} & \overline{a_{i+1}} & \dots & a_M \\ \hline\end{array}$$

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the $\neg$checkl condition for graph G.

Case of a loop *that occurs positively.*

If there exists h such that $\mathsf{loop}(p_h)$ is a constraint that occurs positively in Φ_1, i.e., $\pm_h = +$, and $x_\ell \in var(p_h)$. Assume $p_h = b_1 :: \ldots :: b_p :: x_\ell$. Then there exists a name c repeated in $p_h\sigma$. We mark two occurrences of such a c, when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \dots & \overline{c} & \dots & \overline{c} & \dots & a_P \\ \hline\end{array}$$

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the loop condition for graph G. Indeed, the condition does not depend on the graph.

Case of a loop *that occurs negatively.*

If there exists h such that $\mathsf{loop}(p_h)$ occurs negatively in Φ_1, i.e., $\pm_h = -$, and $x_\ell \in var(p_h)$. Assume that $p_h = b_1 :: \ldots :: b_p :: x_\ell$. Removing nodes from the list preserves this condition, so any extracted list of $x_\ell\sigma$ satisfies the $\neg\mathsf{loop}$ condition. Moreover, as a variation of a list is built with a fresh constant, any variation of a list extracted from $x_\ell\sigma$ satisfies the condition.

Case of a route *that occurs negatively.*

If there exists r_l such that $\mathsf{route}(r_l)$ occurs negatively in Φ_1, i.e., $\pm_l = -$, and $x_\ell \in var(r_l)$. Assume that $r_l = b_1 :: \ldots :: b_p :: x_\ell$. As σ is a solution for Φ_1, we can have two different cases:

- There exists a name c repeated in $r_l\sigma$. Then we mark two occurrences of such a c, when they appear in $x_\ell\sigma$.
- There exists a sublist l of $r_l\sigma$ such that $l = c :: d :: l_1$ and $(c, d) \notin E$. We mark c and d if they appear in $x_\ell\sigma$.

a_1	$\ldots$	$\overline{c}$	$\overline{d}$	$\ldots$	a_P

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the $\neg\mathsf{route}$ condition for G.

Case of a route *that occurs positively.*

If there exists r_l such that $\mathsf{route}(r_l)$ occurs positively in Φ_1, i.e., $\pm_l = +$, and $x_\ell \in var(r_l)$. Assume that $r_l = b_1 :: \ldots :: b_p :: x_\ell$. Write $r_l\sigma = c_1 :: \ldots :: c_n$. As σ is a solution for Φ_1 in G, for every $0 < i < n$, $(c_i, c_{i+1}) \in E$ and for every $i \neq j$, $c_i \neq c_j$. Consequently, $|r_l\sigma|_\ell \leq \#E$, and as $|x_\ell\sigma|_\ell \leq |r_l\sigma|_\ell$, we have that $|x_\ell\sigma|_\ell \leq \#E$. But our hypothesis tells us that $|x_\ell\sigma|_\ell > M \geq \#E$. So there is no positive route condition on x_ℓ.

We count the number of marked names. We have marked the first kN names in the list. For each constraint $\pm\mathsf{checkl}$, we mark at most 3 names in the list. Suppose there are several constraints $\neg\mathsf{route}(l)$ with x_ℓ sublist of l. Either $\neg\mathsf{route}(x_\ell\sigma)$ holds, and we can mark two names in $x_\ell\sigma$ which will make all the $\neg\mathsf{route}$ constraints true; or the constraint is satisfied by marking one name for each constraint. Thus, we need only mark $\mathsf{max}(R, 2)$ names when $R \geq 1$ and 0 otherwise. Thus, in any case, it is sufficient to mark $R + 1$ names in $x_\ell\sigma$. Similarly, it is sufficient to mark $L + 1$ names in $x_\ell\sigma$ to satisfy the loop constraints. The number of names marked in the list is at most

$$kN + 3C + (R + 1) + (L + 1) \leq M.$$

Consider l_1 extracted from $x_\ell\sigma$ by keeping only the marked names in $x_\ell\sigma$ and the first unmarked name. Such an unmarked name exists, because $|x_\ell\sigma|_\ell \geq M$.

Let l_2 be the variation of l_1 replacing the first unmarked name with a fresh constant a_ℓ. For each condition considered above, l_2 satisfies it, as it is a variation of a list extracted from $x_\ell\sigma$ containing the marked names.

Let σ_0 be the substitution such that $x\sigma_0 = x\sigma$ for every $x \in dom(\sigma) \smallsetminus \{x_\ell\}$, and $x\sigma = x$ otherwise. Let $\sigma' = \sigma_0 \cup \{x_\ell \mapsto l_2\}$. By hypothesis, σ is a solution of Φ_1 for G, so by construction, σ' is a solution of Φ_1 for G. Now, it remains for us to show that σ' is a solution of $(\mathcal{C}, \mathcal{I})$ and Φ_2.

Deduction constraints. Consider a deduction constraint $T_i \vdash x_i$ in $\mathcal{C}$. Either x_i is of sort loc or lists, which means that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma'$, thus $T_i\sigma' \cup \mathcal{I} \subseteq T_0 \cup \mathcal{I} \vdash x_i\sigma'$. Or x_i is not of sort loc or lists, so in particular $x_i \in dom(\sigma) \smallsetminus x_\ell$, and $x_i\sigma' = x_i\sigma \in \mathcal{I} \cup \mathcal{N}_{\mathsf{loc}}$, so again $T_i\sigma' \cup \mathcal{I} \subseteq T_0 \cup \mathcal{I} \vdash x_i\sigma'$. Hence, in both cases, we have that $T_i\sigma' \cup \mathcal{I} \vdash x_i\sigma'$. Consequently, σ' is a solution of $(\mathcal{C}, \mathcal{I})$.

Disequality constraints. Consider a disequality constraint $\forall Y.u \neq v \in \Phi_2$. We assume w.l.o.g. that $dom(\sigma) \cap Y = \emptyset$. We have to show that $u\sigma'$ and $v\sigma'$ are not unifiable. We distinguish two cases. Either $u\sigma_0$ and $v\sigma_0$ are not unifiable, but in such a case, we easily deduce that $u\sigma'$ and $v\sigma'$ are not unifiable too. This allows us to conclude. Otherwise, let $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$.

If $dom(\mu) \subseteq Y$, let $\tau = \{x_\ell \mapsto x_\ell\sigma\} \circ \mu$. We have that:

$$\begin{aligned}(u\sigma)\tau &= ((u\sigma_0)\{x_\ell \mapsto x_\ell\sigma\})\tau = (u\sigma_0\mu)\{x_\ell \mapsto x_\ell\sigma\}\\ (v\sigma)\tau &= ((v\sigma_0)\{x_\ell \mapsto x_\ell\sigma\})\tau = (v\sigma_0\mu)\{x_\ell \mapsto x_\ell\sigma\}.\end{aligned}$$

Hence, we deduce that $u\sigma$ and $v\sigma$ are unifiable, and we obtain a contradiction since σ satisfies the constraint $\forall Y.u \neq v$. Hence, this case is impossible.

Otherwise, there exists a term t such that $\mu(x_\ell) = t$, and $var(t) \subseteq Y$. We apply Lemma 5 to the set $T = \{u\sigma_0, v\sigma_0\}$, and the set of equations $P = \{u\sigma_0 = v\sigma_0\}$. We have that $\mu = \mathsf{mgu}(P)$. Since σ_0 is ground, we get that:

$$\begin{aligned}|t|_d &\leq \#dom(\mu).\mathsf{max}(|u\sigma_0|_d, |v\sigma_0|_d)\\ &\leq \#dom(\mu).\mathsf{max}(|u|_d, |v|_d)\\ &\leq kN\end{aligned}$$

We reason by case over t:

- If t is not of sort lists, as σ' is well-sorted, $u\sigma'$ and $v\sigma'$ are not unifiable.

- Suppose $t = [a_1; \ldots; a_n]$, with $a_1, \ldots, a_n$ terms of sort loc. We write $t = t_1@t_2$ with t_2 ground term of maximal size, where @ denotes the concatenation of lists. We have shown that $|t_1|_d = |t|_d \leq kN$.

  We know that $x_\ell\sigma' = [b_1; \ldots; b_p]$ and there exists $k' > kN$ such that $b_{k'} = a_\ell$ and a_ℓ is a name of $\mathcal{I}$ which does not appear anywhere else in the constraints. Consequently, $a_{k'} \neq a_\ell$, and so $x_\ell\sigma' \neq t\theta$ for any substitution θ.

  Now, assume by contradiction that $u\sigma'$ and $v\sigma'$ are unifiable. This means that there exists τ such that $(u\sigma')\tau = (v\sigma')\tau$. Hence, we have that $\tau \circ$

$\{x_\ell \mapsto x_\ell\sigma'\}$ is an unifier of $u\sigma_0$ and $v\sigma_0$. By hypothesis, we have that $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$. Hence, we deduce that there exists θ' such that $\tau \circ \{x_\ell \mapsto x_\ell\sigma'\} = \theta' \circ \mu$. We have that:

- $\tau \circ \{x_\ell \mapsto x_\ell\sigma'\}(x_\ell) = x_\ell\sigma'$, and
- $\theta' \circ \mu(x_\ell) = t\theta'$.

This leads to a contradiction.

- Suppose $t = a_1 :: \ldots :: a_n :: y_\ell$, with $y_\ell \in Y$ variable of sort lists. We know that $|t|_d \leq kN$, thus we must have $n < kN$. We reason by contradiction. Assume that there exists θ' such that $(u\sigma')\theta' = (v\sigma')\theta'$. In the remaining of the proof, we show that $u\sigma$ and $v\sigma$ are unifiable.

  By hypothesis, we have that $\theta' \circ \{x_\ell \mapsto x_\ell\sigma'\}$ is an unifier of $u\sigma_0$ and $v\sigma_0$. Since $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$, we deduce that there exists ρ' such that:

$$\rho' \circ \mu \;=\; \theta' \circ \{x_\ell \mapsto x_\ell\sigma'\}.$$

  We have that $x_\ell\sigma' = (x_\ell\mu)\rho' = t\rho'$. By hypothesis, we know that the size of $x_\ell\sigma$ is greater than $M \geq kN > n$. Let l_t be the list obtaining from $x_\ell\sigma$ by removing its n first elements. Let ρ_0 be a substitution such that $x\rho_0 = x\rho'$ for every $x \in \mathit{dom}(\rho) \smallsetminus \{y_\ell\}$, and $y\rho_0 = y$ otherwise. Let $\rho = \rho_0 \circ \{y_\ell \mapsto l_t\}$. In order to conclude, it remains to show that $\rho \circ \mu$ is an unifier of $u\sigma$ and $v\sigma$.

  We have that $x_\ell\sigma' = (x_\ell\mu)\rho' = t\rho' = a_i\rho' :: \ldots a_n\rho' :: y_\ell\rho'$. Moreover, we know that $x_\ell\sigma$ and $x_\ell\sigma'$ have the same first kN elements by construction, and $n < kN$. Relying on this fact to establish the last equality, we have that:

$$\begin{array}{rcl} (x_\ell\mu)\rho & = & t\rho \\ & = & (a_1 :: \ldots :: a_n :: y_\ell)\rho \\ & = & a_1\rho :: \ldots :: a_n\rho :: l_t \\ & = & a_1\rho' :: \ldots :: a_n\rho' :: l_t \\ & = & x_\ell\sigma. \end{array}$$

  Hence, we have that $((u\sigma)\mu)\rho = ((u\sigma_0)\mu)\rho$, and $((v\sigma)\mu)\rho = ((v\sigma_0)\mu)\rho$. We easily conclude that $u\sigma$ and $v\sigma$ are unifiable since we know that $(u\sigma_0)\mu = (v\sigma_0)\mu$.

In all possible cases, σ' satisfies the disequality constraint.

As a conclusion, σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$, smaller than σ, which leads to a contradiction. $\square$

Appendix C.3. Case of an a priori *unknown topology*

In case the topology is not fixed, we show that we can bound the size of an attack, possibly by changing the graph. The proof follows the same lines as the proof of Proposition 4. However, we can not consider the size of the graph to bound the size of the lists. This is used in the proof of Proposition 4 to deal with the case of route that occur positively in the formula. In Proposition 5, we rely on the fact that we can change the graph to solve this problem.

Lemma 6. *Let $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a graph, $(\mathcal{C}, \mathcal{I})$ be a special constraint system, Φ_1 be a formula of $\mathcal{L}_{\mathsf{route}}$, and Φ_2 be a set of disequality constraints. Let σ be a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G and $\mathcal{N}'_{\mathsf{loc}} = \mathcal{N}_{\mathsf{loc}} \cap names(\mathcal{C}, \Phi_1, \Phi_2, \sigma)$. Let $G' = (\mathcal{N}_{\mathsf{loc}}, E')$ be a graph that coincides with G on $\mathcal{N}'_{\mathsf{loc}}$, i.e., such that $E = \{(n_1, n_2) \mid (n_1, n_2) \in E' \text{ and } n_1, n_2 \in \mathcal{N}'_{\mathsf{loc}}\}$. Then σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G'.*

Proof. We show that σ satisfies each constraint in $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ when the underlying graph is G'. First, not that σ trivially satisfies the deduction constraints, the disequality constraints and the loop constraints.

In order to conclude, we have to check that this result also holds for the remaining constraints in Φ_1.

- $[\![\mathsf{check}(a\sigma, b\sigma)]\!]_G = 1$ if, and only if, $(a\sigma, b\sigma) \in E$. We have that $[\![\mathsf{check}(a\sigma, b\sigma)]\!]_G = 1$ if, and only if, $[\![\mathsf{check}(a\sigma, b\sigma)]\!]_{G'} = 1$.
- $[\![\mathsf{checkl}(c\sigma, l\sigma)]\!]_G = 1$ if, and only if, $l\sigma$ is of sort lists, $c\sigma$ appears exactly once in $l\sigma$, and for any l' sub-list of $l\sigma$,
  - if $l' = a :: c\sigma :: l_1$, then $(a, c\sigma) \in E$.
  - if $l' = c\sigma :: b :: l_1$, then $(b, c\sigma) \in E$.

  As in the previous case, we easily conclude that $[\![\mathsf{checkl}(c\sigma, l\sigma)]\!]_G = 1$ if, and only if, $[\![\mathsf{checkl}(c\sigma, l\sigma)]\!]_{G'} = 1$.
- $[\![\mathsf{route}(l\sigma)]\!]_G = 1$ if, and only if, $l\sigma$ is of sort lists, $l\sigma = [a_1; \ldots; a_n]$, for every $1 \leq i < n$, $(a_i, a_{i+1}) \in E$, and for every $1 \leq i, j \leq n, i \neq j$ implies that $a_i \neq a_j$. As in the previous case, $(a_i, a_{i+1}) \in E$ if, and only if, $(a_i, a_{i+1}) \in E'$. Hence, $[\![\mathsf{route}(l\sigma)]\!]_G = 1$ if, and only if, $[\![\mathsf{route}(l\sigma)]\!]_{G'} = 1$.

Hence, σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G'. □

Proposition 5. *Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for the graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$, then there exists a graph $G' = (\mathcal{N}_{\mathsf{loc}}, E')$ and a substitution σ such that σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G', and σ is polynomially bounded in the size of Φ_1 and Φ_2. Moreover, we have that G' coincides with G on $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$, i.e., $E = \{(n_1, n_2) \in E' \mid n_1, n_2 \in V\}$.*

Proof. We adapt the proof of Proposition 4 by showing that there exists a solution σ such that for every variable x of sort lists, we have that $|x\sigma|_\ell \leq M = 2 \times (kN + 3C + L + R + 2)$ where k, N, C, L, and R are defined as in Proposition 4.

Let σ be a solution of $(\mathcal{C},\mathcal{I})\wedge\Phi_1\wedge\Phi_2$ for graph G and assume that there exists a variable x_ℓ of sort lists such that $|x_\ell\sigma|_\ell > M$. Let V_{ubi} be a set of $M/2$ fresh nodes, i.e., names in $\mathcal{N}_{\mathsf{loc}}$ that do not occur in $\mathcal{C}$, Φ_1, Φ_2, and E. Consider G' the ubiquitous graph associated to G and V_{ubi}. Note that by definition of G', we have that G' coincides with G on $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$. We show that we can build σ', a solution of $(\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G', such that for $x \neq x_\ell$, $x\sigma' = x\sigma$, and $|x_\ell\sigma'|_\ell \leq M$.

We build σ' in a similar way as in the previous proof. We mark $x_\ell\sigma$ as in the previous proof. The number of names marked in the list is at most:

$$kN + 3C + (R + 1) + (L + 1) \leq M/2.$$

Consider l_1 extracted from $x_\ell\sigma$ by leaving exactly one unmarked name between sequences of marked names. Note that, we have no more than $M/2$ unmarked names in l_1. Let l_2 be the ubiquitous variation of l_1 according to V_{ubi}. The fact that we consider a ubiquitous variation allows one to satisfy the constraint route that occurs positively. Note that, we have no more than $M/2$ ubiquitous names in l_2, so $|l_2|_\ell \leq M$.

Let σ_0 be the substitution such that $x\sigma_0 = x\sigma$ for every $x \in dom(\sigma) \smallsetminus \{x_\ell\}$, and $x\sigma = x$ otherwise. Let $\sigma' = \sigma_0 \cup \{x_\ell \mapsto l_2\}$. By construction, we have that the substitution σ' satisfies Φ_1. We show that σ' is a solution of $(\mathcal{C},\mathcal{I})$ and Φ_2 for G' as in Proposition 4. □

Appendix D. Decidability

Theorem 1. *Let $K = (\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ be an initial concrete configuration with a hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ such that there is an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Let $K_s = (\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0];\mathcal{S};\mathcal{I};\emptyset)$. K_s is a ground symbolic configuration whose concretization is $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0];\mathcal{S};\mathcal{I})$. Let V_K be the set of names of sort loc that occur in $\mathcal{P}$ and $\mathcal{M}$. Our decision procedure works as follows:

Step 1. We partially guess the graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$. Actually, we guess whether $(n_1, n_2) \in E$ for every $n_1, n_2 \in V_K$.
Let $G_K = (\mathcal{N}_{\mathsf{loc}}, E_K)$ where $E_K = \{(n_1, n_2) \mid (n_1, n_2) \in E \text{ and } n_1, n_2 \in V_K\}$.

Step 2. We guess a path of execution of the symbolic transition rules w.r.t. the graph G_K.

$$K_s \rightarrow^{s*}_{G_K,\mathcal{M}} (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}';\mathcal{S}';\mathcal{I}';\mathcal{C}).$$

Step 3. Let $\mathcal{C}' = \mathcal{C} \wedge \{u = \mathsf{error}\} = \mathcal{C}'_0 \wedge \Phi'_1 \wedge \mathsf{Eq}'$ where $\mathcal{C}'_0$ is a finite set of deduction constraints, Eq' is a finite set of unification constraints, and Φ'_1 contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. Let σ be an mgu of the equality constraints in Eq'. Let $\Phi_1 = \Phi'_1\sigma$.

Step 4. Since K is an initial configuration, we have that $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \uplus \mathcal{I}_f$ for some finite set of terms $\mathcal{I}_f$. Let $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \smallsetminus \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$. We have that $\mathcal{C}'_0\sigma = \overline{\mathcal{D}}^{\mathcal{I}_0}$ for some special constraint system $(\mathcal{D}, \mathcal{I}_0)$.

Step 5. We guess a sequence of transformation rules from $\mathcal{D}$ to $\mathcal{D}'$ where $\mathcal{D}'$ is a constraint system in solved form. We have that:

$$\mathcal{D} \rightsquigarrow^*_{\sigma'} \mathcal{D}' \text{ with } \mathcal{D}' \text{ in solved form.}$$

Step 6. We compute the conjunctive normal form of the formula $\Phi_1\sigma'$. Hence, $\Phi_1\sigma'$ is equivalent to

$$\bigwedge_k \phi_1^k \vee \dots \vee \phi_{i_k}^k.$$

We choose non-deterministically $\phi_{\alpha_k}^k$ for every k. Let $\Phi_2 = \bigwedge_k \phi_{\alpha_k}^k$.

Step 7. Let S be the DAG size of $\mathcal{P}$, $\mathcal{S}$, Φ, $\mathcal{M}$, and $\mathcal{I}_f$. Let $\mathcal{I}'_0$ be a finite subset of $\mathcal{I}_0$ of size $2\mathsf{S}^2 \times (\mathsf{S}^4 + 5\mathsf{S}^2 + 2)$. Guess the values of variables which are not of sort lists in $\mathcal{I}'_0 \cup \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$. Guess the values of variables of sort lists among lists of nodes in $\mathcal{I}'_0 \cup \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$ of length at most $2 \times (\mathsf{S}^4 + 5\mathsf{S}^2 + 2)$. This gives us a substitution σ and we guess a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ such that $E \subseteq \{(n_1, n_2) \mid n_1, n_2 \in \mathcal{I}'_0 \cup \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)\}$ and that coincides with G_K on V_K, i.e:

$$E_K = \{(n_1, n_2) \in E \mid n_1, n_2 \in V_K\}.$$

Lastly, we check whether σ is a solution of $(\mathcal{D}', \mathcal{I}_0) \wedge \Phi_2$ for the graph G.

Proof. We now explain each step of our algorithm.

Step 1. We have that $\#V_K < \#\mathit{names}(\mathcal{P}, \mathcal{M})$. Hence, we can guess G_K whose size is polynomially bounded.

Step 2. For every graph $G' = (\mathcal{N}_{\mathsf{loc}}, E')$ with $E_K = \{(n_1, n_2) \in E' \mid n_1, n_2 \in V_K\}$, we have that:

$$(\mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^{s*}_{G_K, \mathcal{M}} (\mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}') \text{ iff } (\mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^{s*}_{G', \mathcal{M}} (\mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}').$$

So we can guess the transitions knowing only E_K. Now, thanks to Theorem 3 we deduce that there is an $\mathcal{M}$-attack on K and Φ for graph G if, and only if, there is a derivation

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}; \emptyset) \rightarrow^{s*}_{G_K, \mathcal{M}} (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

and the constraint system $\mathcal{C}' = \mathcal{C} \wedge \{u = \mathsf{error}\}$ has a solution for graph G.

Actually, we can guess such a path. Indeed, the number of derivations starting from configuration K_s is bounded. Actually, the length of possible paths is bounded by the size of the protocol: as there is no replication in the initial configuration, each transition leads to a smaller process. Moreover, the number of configurations reachable with one symbolic transition is bounded as well: we can first guess which process is going to evolve and which is the corresponding transition. There is only one possible resulting configuration once this is chosen, except for the communication transition, where we also have to guess which neighbors will receive the message, and for the read transition, where we have to choose which term to read.

Step 3. Let σ be an mgu of the equality constraints in $\mathcal{C}'$. If σ does not exist, then $\mathcal{C}'$ has no solution. Else, $\mathcal{C}'$ has a solution if, and only if, $\mathcal{C}'\sigma$ has a solution.

Step 4. First, $\mathcal{C}'_0\sigma$ is a constraint system. In particular, it satisfies the origination property since application of a substitution preserves this property. We have that $\mathcal{C}'_0\sigma = \overline{\mathcal{D}}^{\mathcal{I}_0}$ where $(\mathcal{D}, \mathcal{I}_0)$ is a special constraint system.

Step 5. We apply Theorem 4. Thus, there exists a solution θ of $(\mathcal{D}, \mathcal{I}_0)$ and Φ_1 for graph G if, and only if, there exists a special constraint system $(\mathcal{D}', \mathcal{I}_0)$ in solved form and some substitutions σ', and θ' such that $\theta = \theta' \circ \sigma'$, $\mathcal{D} \rightsquigarrow^*_{\sigma'} \mathcal{D}'$ and θ' is a solution for $\mathcal{D}'$ and $\Phi_1\sigma'$ for graph G.

Step 6. This step is straightforward. The formula $\Phi_1\sigma'$ contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. Consequently, $\Phi_2 = \bigwedge_k \phi^k_{\alpha_k}$, obtained from $\Phi_1\sigma'$, can be written:

$$\begin{aligned}\Phi_2 = \bigwedge_i \forall Y_i . u_i \neq v_i \ \wedge \ \bigwedge_j \pm_j \, \mathsf{check}(a_j, b_j) \ \wedge \\ \bigwedge_k \bigwedge_i \pm_{i_k} \mathsf{checkl}(c_{i_k}, l_k) \ \wedge \ \bigwedge_h \pm_h \mathsf{loop}(p_h) \ \wedge \ \bigwedge_l \pm_l \mathsf{route}(r_l)\end{aligned}$$

Finally, we are left to decide whether there exists a solution to a solved special constraint system $(\mathcal{D}', \mathcal{I}_0)$ and a formula Φ_2 as described above.

Step 7. First, we show that for any term $t \in St(\mathcal{D}', \Phi_2)$, there exists t' in $St(\mathcal{C}', \Phi'_1)$ such that $t = (t'\sigma)\sigma'$. Thanks to Theorem 4, we have that

$$St(\mathcal{D}') \subseteq St(\mathcal{D}\sigma') \subseteq St(\mathcal{D})\sigma'.$$

Moreover, we have that

$$St(\Phi_2) \subseteq St(\Phi_1\sigma') \subseteq St(\Phi_1)\sigma' \cup \bigcup_{x \in var(\mathcal{D})} St(x\sigma') \subseteq St(\Phi_1)\sigma' \cup St(\mathcal{D})\sigma'.$$

The last inclusion is a direct consequence of the inclusion $St(\mathcal{D}\sigma') \subseteq St(\mathcal{D})\sigma'$. Hence, we have that: $St(\mathcal{D}', \Phi_2) \subseteq St(\Phi_1)\sigma' \cup St(\mathcal{D})\sigma' \subseteq St(\mathcal{C}'\sigma)\sigma'$. By relying on Lemma 4, we obtain that $St(\mathcal{C}'\sigma) \subseteq St(\mathcal{C}')\sigma$. Since $names(\mathcal{D}', \Phi_2) \cap \mathcal{I}_0 = \emptyset$, we deduce that

$$\begin{array}{rcl} St(\mathcal{D}', \Phi_2) & \subseteq & St(\mathcal{C}')\sigma\sigma' \smallsetminus \mathcal{I}_0 \\ & \subseteq & (St(\mathcal{C}')\sigma\sigma' \smallsetminus \mathcal{N}_{\mathsf{loc}}) \cup names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f) \end{array}$$

Let S be the DAG size of $\mathcal{P}$, $\mathcal{S}$, Φ, $\mathcal{M}$, and $\mathcal{I}_f$. By inspection of the symbolic transition rules, we see that at each step, the constraint system can grow at most of size S (because of the communication rule). Hence, we have that $\#St(\mathcal{D}', \Phi_2) \leq \mathsf{S}^2$.

Let N be the maximal depth of variables in the terms of all disequality constraints in Φ_2, and k the maximal total number of variables in a disequality constraint. We have that $kN \leq \mathsf{D}^2$ where D is the DAG size of the largest disequality constraint that occurs in $\mathcal{D}'$. Since $\mathsf{D} \leq \#St(\mathcal{D}', \Phi_2)$, we deduce that $kN \leq \mathsf{D}^2 \leq \mathsf{S}^4$.

Let L be the number of occurrences of a loop predicate in Φ_2, R be the number of occurrences of a route predicate in Φ_2, and C be the number of occurrences of a checkl predicate in Φ_2. We have that:

$$L \leq \mathsf{S}^2,\ R \leq \mathsf{S}^2, \text{and } C \leq \mathsf{S}^2.$$

Now, we have to show that if there exists a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ such that $E_K = \{(n_1, n_2) \in E \mid n_1, n_2 \in V_K\}$ and on which there is an attack, then there exists a graph as described in Step 7 for which there is an attack and the substitution witnessing the fact that there exists an attack is also as described in Step 7 of our algorithm.

- Thanks to Lemma 3, we know that there is a solution where the variables which are not of sort loc or lists are substituted by names in $\mathcal{I}_0$ (independently of the underlying graph).
- Thanks to Proposition 5, we know that if there is a graph $G = (\mathcal{N}_{\mathsf{loc}}, E)$ leading to a solution, there exists a substitution σ where the size of the instantiated variables of sort lists is bounded by $M = 2 \times (kN + 3C + R + L + 2)$ and there exists a graph $G' = (\mathcal{N}_{\mathsf{loc}}, E')$ that coincides with G on $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$.

  We have that: $M \leq 2 \times (\mathsf{S}^4 + 5\mathsf{S}^2 + 2)$. Hence, the number of distinct names of sort loc in σ is bounded by $\#var(\mathcal{D}', \Phi_2) \times M \leq 2\mathsf{S}^2 \times (\mathsf{S}^4 + 5\mathsf{S}^2 + 2)$. We consider a set $\mathcal{I}'_0$ having this size. So, there is a solution σ for G' such that $names(\sigma) \subseteq \mathcal{I}'_0 \cup names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$.
- Thanks to Lemma 6, we know that if σ is a solution for graph $G' = (\mathcal{N}_{\mathsf{loc}}, E')$, then σ is also a solution for any graph $G'' = (\mathcal{N}_{\mathsf{loc}}, E'')$ that coincides with G' on $\mathcal{N}'_{\mathsf{loc}}$ where $\mathcal{N}'_{\mathsf{loc}}$ represents the names in $\mathcal{N}_{\mathsf{loc}}$ that occur in $\mathcal{D}'$, Φ_2, and σ. Note that $\mathcal{N}'_{\mathsf{loc}} \subseteq \mathcal{I}'_0 \cup names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$

Let $G'' = (\mathcal{N}_{\mathsf{loc}}, E'')$ be the graph such that

$$E'' = \{(n_1, n_2) \in E' \mid n_1, n_2 \in \mathcal{I}'_0 \cup \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)\}.$$

We have that σ is a solution for the graph G'' and the graph G'' is as described in Step 7. $\square$

We will now explain how to decide the existence of an attack given a fixed graph G.

Theorem 2. *Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with a hole, $G = (\mathcal{N}_{\mathsf{loc}}, E)$ be a finite graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a formula. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Let $K_s = (\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}; \emptyset)$. First, K_s is a ground symbolic configuration whose concretization is $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I})$. We write $G = (\mathcal{N}_{\mathsf{loc}}, E)$. Let $V = \{n \mid \exists n' \text{ such that } (n, n') \in E\}$. Our decision procedure works as follows:

Step 1. We guess a path of execution of the symbolic transition rules w.r.t. graph G.

$$K_s \rightarrow^{s*}_{G,\mathcal{M}} (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}).$$

Step 2. Let $\mathcal{C}' \wedge \{u = \mathsf{error}\} = \mathcal{C}'_0 \wedge \Phi'_1 \wedge \mathsf{Eq}'$ where $\mathcal{C}'_0$ is a finite set of deduction constraints, Eq' is a finite set of unification constraints, and Φ'_1 contains disequality constraints and formula of $\mathcal{L}_{\mathsf{route}}$. Let σ be an mgu of the equality constraints in Eq'. Let $\Phi_1 = \Phi'_1\sigma$.

Step 3. Since K is an initial configuation, we have that $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}_f$ for some finite set of terms $\mathcal{I}_f$. Let $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \smallsetminus \mathit{names}(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f)$. We have that $\mathcal{C}'_0\sigma = \overline{\mathcal{D}}^{\mathcal{I}_0}$ for some special constraint system $(\mathcal{D}, \mathcal{I}_0)$.

Step 4. We guess a sequence of transformation rules from $\mathcal{D}$ to $\mathcal{D}'$ where $\mathcal{D}'$ is a constraint system in solved form. We have that:

$$\mathcal{D} \rightsquigarrow^*_{\sigma'} \mathcal{D}' \text{ with } \mathcal{D}' \text{ in solved form.}$$

Step 5. We compute the conjunctive normal form of formula $\Phi_1\sigma'$. Hence, $\Phi_1\sigma'$ is equivalent to

$$\bigwedge_k \phi^k_1 \vee \cdots \vee \phi^k_{i_k}.$$

We choose non-deterministically $\phi^k_{\alpha_k}$ for every k. Let $\Phi_2 = \bigwedge_k \phi^k_{\alpha_k}$.

Step 6. Let S be the DAG size of $\mathcal{P}$, $\mathcal{S}$, Φ, $\mathcal{M}$, and $\mathcal{I}_f$. Let $\mathcal{I}_0'$ be a finite subset of $\mathcal{I}_0$ of size $\mathsf{S}^2 \times \mathsf{max}(\mathsf{S}^4 + 5\mathsf{S}^2 + 3, \#E)$. Guess the values of variables of sort lists among lists of nodes in $\mathcal{I}_0' \cup names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f) \cup V$ of length at most $\mathsf{max}(\mathsf{S}^4 + 5\mathsf{S}^2 + 3, \#E)$. Guess the values of the other variables, i.e. those that are not of sort lists, in $\mathcal{I}_0' \cup names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{M}, \mathcal{I}_f) \cup V$. This gives us a substitution σ. Lastly, we check whether σ is a solution of $(\mathcal{D}', \mathcal{I}_0) \wedge \Phi_2$ for graph G.

Proof. The first five steps are the same as Steps 2 to 6 in Theorem 1. Thus, it remains to justify Step 6 of the procedure described above. As shown in the proof of Step 7 in Theorem 1, we have that:

- $N \leq \mathsf{S}^2$ where N is the maximal depth of variables in the terms of all disequality constraints in Φ_2;
- $k \leq \mathsf{S}^2$ where k is the maximal total number of variables in a disequality constraint in Φ_2;
- $L \leq \mathsf{S}^2$ where L is the number of occurrences of a loop predicate in Φ_2;
- $C \leq \mathsf{S}^2$ where C is the number of occurrences of a checkl predicate in Φ_2;
- $R \leq \mathsf{S}^2$ where R is the number of occurrences of a route predicate in Φ_2.

Now, we want to show that if there exists an attack for graph G, then there is an attack captured by a substitution as described in Step 6 of our algorithm.

- Thanks to Lemma 3, we know that there is a solution where the variables which are not of sort lists are substituted by names in $\mathcal{I}_0$.
- Thanks to Proposition 4, we know that if there is a solution, there exists one, say σ, such that $|x\sigma| \leq M$ for any x of type lists where:

$$M = \mathsf{max}(kN + 3C + L + R + 3, \#E).$$

Actually, we have that $M \leq \mathsf{max}(\mathsf{S}^4 + 5\mathsf{S}^2 + 3, \#E|)$.
Hence, the number of distinct names of sort loc in σ is bounded by

$$\#var(\mathcal{D}', \Phi_2) \times M \leq \mathsf{S}^2 \times \mathsf{max}(\mathsf{S}^4 + 5\mathsf{S}^2 + 3, \#E).$$

We consider a set $\mathcal{I}_0'$ having this size. This allows us to conclude. $\square$