Mathilde Arnaud, Véronique Cortier,
and Stéphanie Delaune

Modeling and Verifying Ad Hoc Routing Protocols

Research Report LSV-10-03

February 2010

Laboratoire Spécification et Vérification

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Modeling and Verifying Ad Hoc Routing Protocols *

Mathilde Arnaud[1,2] Véronique Cortier[1]
Stéphanie Delaune[2]

[1] LORIA, CNRS & INRIA Nancy Grand Est, France
[2] LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France

Abstract

Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their infrastructure. In such networks, a central issue, ensured by routing protocols, is to find a route from one device to another. Those 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 the protocol SRP applied to DSR.

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 of the currently proposed routing protocols for mobile ad hoc networks are assumed to be used in a friendly environment (e.g. [23, 15]). Recent research has recognized that this assumption is unrealistic and that attacks can be mounted [13, 20, 9]. Since an adversary can easily paralyse 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, secure versions of routing protocols have been developed to ensure that mobile ad hoc networks can work even in an adversarial setting [28, 13, 21]. Those routing protocols use cryptographic mechanisms such as encryption, signature, MAC, in order to prevent a malicious node to insert and delete nodes inside a path.

Formal modeling and analysis techniques are well-adapted for checking correctness of security protocols. Formal methods have for example been successfully used for authentication or key establishment security protocols and a multitude of effective frameworks have been proposed (e.g. the Paulson inductive model [22], the strand spaces model [26], the applied-pi calculus [1] or constraints systems [24] to cite only a few). While secrecy and authentication properties have been shown undecidable in the general case [12], many decision procedures have been proposed. For example, secrecy and authentication become NP-complete for a bounded number of sessions [24] and Bruno Blanchet has developed a (semi) decision procedure for security protocols encoded as Horn clauses [7]. This yielded various efficient tools for detecting flaws and proving security (e.g. ProVerif [8] or Avispa [5]).

While key-exchange protocols are well-studied in traditional networks, there are very few attempts to develop formal techniques allowing an automated analysis of secure routing protocols. Up to our knowledge,

*This work has been partially supported by the project ANR-07-SESU-002 AVOTÉ.

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 authenticity properties.

Our contributions. The first main contribution of this paper is the proposition of a calculus, inspired from CBS# [20], which allows mobile wireless networks and their security properties to be formally described and analyzed. We propose in particular a logic to express the tests performed by the nodes at each step. It allows for example checking whether a route is "locally" valid, given the information known by the node. 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. To model routing protocols in an accurate way, some features need to be taken into account. Among them:

- *Network topology*: nodes can only communicate (in a direct way) with their neighbor.
- *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.

There are also some implications for the attacker model. Indeed, in most existing formal approaches, the attacker controls the entire network. This abstraction is reasonable 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.

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 modeling where the attacker controls all the communications. We thus can model as usual all the key exchange and authenticity protocols presented e.g. in the Clark & Jacob library [10]. Since we also provide each node with a memory, our model can also capture protocols where a state global to all sessions is assumed for each agent. It is typically the case of protocols where an agent should check that a key has not been already accepted in a previous session, in order to protect the protocol against replay attacks.

Our second main contribution is to provide two NP decision procedures for analyzing routing protocols for a bounded number of sessions. For a fixed set of roles and sessions, our first decision procedure allows to discover whether there exists a network topology and a malicious behavior of some nodes that yields 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. 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 [11] 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 (Secure Routing Protocol) [21] applied on the protocol DSR (Dynamic Source Routing Protocol) [16]. This allows us to retrieve an attack presented first in [9].

Related work. Recently, some frameworks have been proposed to model wireless communication and/or routing protocols in a more accurate way. For example, Yang and Baras [27] provide a first symbolic model for routing protocols based on strand spaces, modeling the network topology but not the cryptographic primitives that can be used for securing communications. They also implement a semi-decision procedure to search for attacks. Buttyán and Vadja [9] provide a cryptographic model for routing protocol, in a cryptographic setting. They provide a security proof (by hand) for a fixed protocol. Ács [3] uses a similar cryptographic model and provides proofs of security (or insecurity) for several protocols, but his method can not be automated. Nanz and Hankin [20] propose a process calculus to model the network topology and broadcast communications. They also propose a

decision procedure but for an intruder that is already specified by the user. This allows to check security only against fixed, known in advance scenarios. 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. Schaller *et al* [25] 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. However, no generic procedure is proposed for proving security.

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 amendable to implementation, re-using existing tools (such as Avispa [5]).

A preliminary version of this work was presented at Secret'09 [6] (with informal proceedings). In this first version, our decision procedure only held for a network topology fixed in advance and for a restricted fragment of security properties. Detailed proofs of our result can be found in Appendix.

Outline. Section 2 presents our formal model for routing protocol. It is illustrated with the modeling of the SRP protocol. We then give an alternative symbolic semantics in Section 3, more amendable to automation, and we show its correctness and completeness w.r.t. the concrete semantics. Finally, we state our main decidability result in Section 4 and present a detailed sketch of its proof. Some concluding remarks can be found in Section 5.

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 *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:

t	::=		term of sort s
	$\mid$	x	variable x of sort s
	$\mid$	a	name a of sort s
	$\mid$	$f(t_1, \ldots, t_k)$	application of symbol $f \in \mathcal{F}$

where t_i is a term of some sort s_i and $ar(f) = s_1 \times \ldots \times s_k \to s$. We assume a special sort terms that subsumes all the other sorts and such that any term is of sort terms. We write $var(t)$ the set of variables occurring in a term t and $St(t)$ the syntactic subterms of t. 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* $\mathcal{F}_1 = \{\mathsf{hmac}, \langle\rangle, ::, \bot, \{\_\}\_\}$*, with the following arities:*

- $\mathsf{hmac} : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$,
- $\langle \_, \_ \rangle : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$,
- $:: : \mathsf{loc} \times \mathsf{lists} \to \mathsf{lists}$,
- $\bot :\to \mathsf{lists}$,
- $\{\_\}\_ : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$.

The sort lists *represents lists of terms of sort* loc*. The symbol* $::$ *is the list constructor.* $\bot$ *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 term* $\{m\}_k$ *represents the message* m *encrypted by the key* k*. 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* $t_1 :: (t_2 :: (t_3 :: \bot))$*.*

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 and only if all of the t_i are ground. 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 $\mathsf{mgu}(t, u) = \bot$ 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 thus is left unspecified. It is typically defined through a deduction system. For example, for 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 concatenate terms, to compute MAC when

$$\frac{T \vdash a \quad T \vdash l}{T \vdash a :: l} \qquad \frac{T \vdash a :: l}{T \vdash a} \qquad \frac{T \vdash a :: l}{T \vdash l}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash u} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash v}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v} \qquad \frac{T \vdash \{u\}_v \quad T \vdash v}{T \vdash u}$$

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

Figure 1. Example of deduction system.

he knows the key, to build lists, to encrypt and decrypt when he knows the keys. Moreover, he is able to retrieve components of a pair or a list.

2.2 Process calculus

Several calculi already exist to model security protocols (e.g. [2, 1]). However, for our purpose, a node, i.e. a process, has to perform some specific actions that can not be easily modeled in such calculi. For instance, a node stores 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. Actually, our calculus is inspired from CBS# [20], 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.

The intended behavior of each node of the network can be modeled by a *process* defined by the grammar given below. Our calculus is parametrized by a set $\mathcal{L}$ of formulas.

$P, Q ::=$	Processes
0	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

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 and then behaves like $P\sigma$ where $\sigma = \mathsf{mgu}(m, u)$. 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 $\sigma = \mathsf{mgu}(m, u)$. If no element of the form u is found, then it behaves like Q. Sometimes, for the sake of clarity, we will omit the null process. We also omit the else part when $Q = 0$.

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 perform some checks on the route they received before accepting a message. Thus we will typically consider the logic $\mathcal{L}_{\mathsf{route}}$ defined by the following grammar:

$\Phi ::=$	Formula
$\mathsf{check}(a, b)$	neighborhood of two nodes
$\mathsf{checkl}(c, l)$	local neighborhood of a node in a list
$\mathsf{route}(l)$	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

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

- $[\![\mathsf{check}(a, b)]\!]_G = 1$ iff $(a, b) \in E$,
- $[\![\mathsf{checkl}(c, l)]\!]_G = 1$ 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 $(b, c) \in E$.
- $[\![\mathsf{route}(l)]\!]_G = 1$ 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,
- $[\![\Phi_1 \wedge \Phi_2]\!]_G = [\![\Phi_1]\!]_G \wedge [\![\Phi_2]\!]_G$, $[\![\neg\Phi]\!]_G = \neg[\![\Phi]\!]_G$, and $[\![\Phi_1 \vee \Phi_2]\!]_G = [\![\Phi_1]\!]_G \vee [\![\Phi_2]\!]_G$.

2.3 Example: modeling the SRP protocol

We consider the secure routing protocol SRP introduced in [21], assuming that each node already knows his neighbors (running e.g. some neighbor discovery protocol). 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 [16]. DSR is a protocol which is used when an agent S (the source) wants to communicate with another agent D (the destination), which is not his immediate neighbor. In an ad hoc network, messages can not be sent directly to the destination, but have to travel along a path of nodes.

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 hmac computed over the content of the request with a key K_{SD} shared by S and D. It 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 his neighbor in the route is indeed a neighbor of S in the network.

Consider the signature given in Example 1 and let $S, D, \mathsf{req}, \mathsf{rep}, id, K_{SD}$ be names ($S, D \in \mathcal{N}_{\mathsf{loc}}$) and x_L be a variable of sort lists. The process executed by a node S initiating the search of a route towards a node D is:

$$P_{\mathsf{init}}(S, D) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].0$$

where:

$$u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle$$
$$u_2 = \langle \mathsf{rep}, D, S, id, x_L, \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, x_L\rangle, K_{SD})\rangle$$
$$\Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg \mathsf{loop}(x_L).$$

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. 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. Below, $V \in \mathcal{N}_{\mathsf{loc}}$, x_S, x_D and x_a are variables of sort loc whereas x_r is a variable of sort lists and x_{id}, x_m are variables of sort terms. The process executed by an intermediary node V when forwarding a request is as follows:

$$P_{\mathsf{req}}(V) = \mathsf{in}\ w_1[\Phi_V].\mathsf{read}\ t\ \mathsf{then}\ 0\ \mathsf{else}\ (\mathsf{store}(t).\mathsf{out}(w_2))$$

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

When the request reaches the destination D, it checks that the request has a correct hmac and that the first node in the route is one of his neighbors. Then, the destination D constructs a route reply, in particular it computes a new hmac over the route accumulated in the request packet with K_{SD}, and sends the answer back over the network.

The process executed by the destination node D is the following:

$$P_{\mathsf{dest}}(D, S) = \mathsf{in}\ v_1[\Phi_D].\mathsf{out}(v_2).0$$

where:

$$v_1 = \langle \mathsf{req}, S, D, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{req}, S, D, x_{id}\rangle, K_{SD})\rangle$$
$$\Phi_D = \mathsf{check}(D, x_a)$$
$$v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l\rangle, K_{SD})\rangle$$

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

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

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

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 its neighbors. More specifically, we assume that the topology of the network is represented by giving an undirected graph $G = (V, E)$ with $V \subseteq \mathcal{N}_{\mathsf{loc}}$, where an edge in the graph models the fact that two nodes are neighbors. We also assume that we have a set of nodes $\mathcal{M}$ 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. However, it is well-known that the presence of several colluding malicious nodes often yields straightforward attacks [14, 18].

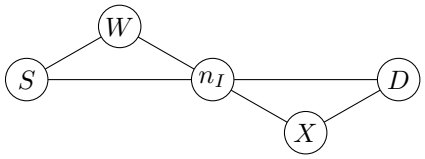
A (ground) *concrete configuration* of the network is a triplet $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ where:

- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$ where null processes, i.e. expressions of the form $\lfloor 0 \rfloor_n$ are removed. $\lfloor P \rfloor_n$ represents the (ground) process P located at node $n \in V$. 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 $n \in V$ 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.

Example 2 *Continuing our modeling of SRP, a typical initial configuration for the SRP protocol is*

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

where both the source node S and the destination node D wish to communicate. We assume that each node has an empty storage list and that the initial knowledge of the intruder is given by an infinite set of terms $\mathcal{I}_0$. A possible network configuration is modeled by the graph G_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 assume that the intermediary nodes n_I, W, and X execute the routing protocol. Actually, this is not needed to show that the protocol is flawed, and we want to keep this example as simple as possible.



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 2. They are parametrized by the underlying graph G and the set of malicious nodes $\mathcal{M}$.

The relation $\rightarrow^*_{G,\mathcal{M}}$ is the reflexive and transitive closure of $\rightarrow_{G,\mathcal{M}}$. We may write $\rightarrow$, $\rightarrow_G$, $\rightarrow_{\mathcal{M}}$ 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 case we assume that there is a single malicious node with each honest node is connected to it, we retrieve the model where the attacker is assumed to control all the communications.

Example 3 *Continuing the example developed in Section 2.3, the following sequence of transitions is enabled from the initial configuration K_0:*

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

where:

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

During this transition, S broadcasts to its neighbors a request 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 its 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 broadcasts it. Since $(X, D) \in E$, D accepts this message and the resulting configuration of the transition is

$$(\lfloor \mathsf{in}\ u_2[\Phi_S].0 \rfloor_S \cup \lfloor \mathsf{out}(v_2\sigma).0 \rfloor_D; \emptyset; \mathcal{I}_0 \cup \{u_1\})$$

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

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

Definition 1 *Let G 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 $W = \mathsf{in}\ s.\_$ 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 $G = (V, E)$ can be defined by the (non) existence of an $\{n_I\}$-attack on the configuration $(\mathcal{P} \cup \lfloor W \rfloor_n; \mathcal{S}; \mathcal{I})$ and the formula true for the graph $G' = (V \cup \{n\}, E \cup \{(n, n_I)\})$.

Example 4 *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 $P_{\mathsf{init}}(S, D)$ by a hole, and checking whether the formula $\neg \mathsf{route}(x_L)$ holds. Let $P'_{\mathsf{init}}(S, D)$ be the resulting process.*

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

$$\textsc{Comm}\quad (\{\lfloor \mathsf{in}\ u_j[\Phi_j].P_j\rfloor_{n_j} \mid \mathsf{mgu}(t,u_j)\neq\bot, [\![\Phi_j\sigma_j]\!]_G=1, (n,n_j)\in E\}\cup \lfloor \mathsf{out}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\{\lfloor P_j\sigma_j\rfloor_{n_j}\}\cup\lfloor P\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}')$$
where $\sigma_j=\mathsf{mgu}(t,u_j)$, $\mathcal{I}'=\mathcal{I}\cup\{t\}$ if $(n,n_I)\in E$ for some $n_I\in\mathcal{M}$ and $\mathcal{I}'=\mathcal{I}$ otherwise.

$$\textsc{In}\quad (\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\sigma\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I})$$
if $(n_I,n)\in E$ for some $n_I\in\mathcal{M}$, $\mathcal{I}\vdash t$, $\sigma=\mathsf{mgu}(t,u)$ and $[\![\Phi\sigma]\!]_G=1$

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

$$\textsc{Read-Then}\quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n\cup\mathcal{P};\lfloor t\rfloor_n\cup\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\sigma\rfloor_n\cup\mathcal{P};\lfloor t\rfloor_n\cup\mathcal{S};\mathcal{I})$$
where $\sigma=\mathsf{mgu}(t,u)$

$$\textsc{Read-Else}\quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor Q\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I})$$
if for all t such that $\lfloor t\rfloor_n\in\mathcal{S}$, $\mathsf{mgu}(t,u)=\bot$

$$\textsc{If-Else}\quad (\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \quad \text{if } [\![\Phi]\!]_G=1$$

$$\textsc{If-Then}\quad (\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor Q\rfloor_n\cup\mathcal{P};\mathcal{S};\mathcal{I}) \quad \text{if } [\![\Phi]\!]_G=0$$

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

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

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

Figure 2. Concrete transition system.

where $P=\mathsf{if}\ \neg\mathsf{route}(x_L)\ \mathsf{then}\ \mathsf{out}(\mathsf{error})$. *Then, we recover the attack mentioned in [9] with the topology* G_0 *given in Example 2, and from the initial configuration:*

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

Indeed, we have that:

$$\begin{aligned}K_0' &\to^* (\lfloor \mathsf{in}\ u_2[\Phi_S].P\rfloor_S\cup\lfloor \mathsf{out}(m').0\rfloor_D;\emptyset;\mathcal{I})\\ &\to (\lfloor \mathsf{in}\ u_2[\Phi_S].P\rfloor_S\cup\lfloor 0\rfloor_D;\emptyset;\mathcal{I}')\\ &\to (\lfloor \mathsf{if}\neg\mathsf{route}([X;W;S])\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\rfloor_S;\emptyset;\mathcal{I}')\\ &\to (\lfloor \mathsf{out}(\mathsf{error}).0\rfloor_S;\emptyset;\mathcal{I}')\end{aligned}$$

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

3 Symbolic semantics

It is difficult to directly reason with the transition system defined in Figure 2 since it is infinitely branching. Indeed, a potentially infinite number of distinct messages can be sent at each step by the intruder node. 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. [4]).

3.1 Constraint systems

As in [19, 11, 24], 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.

Definition 2 *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 non empty set of terms, and* X *is a set of variables. Moreover, we assume that the constraints in* $\mathcal{C}$ *can be ordered* $C_1,\ldots,C_n$ *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. Actually, the set $rvar(\mathcal{C})$ represents all the free variables. This means that all the variables are introduced in the right-hand-side of a unification constraint or a deduction constraint.

Note that our disequality constraints are rather general since they do not simply allow 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 $dom(\theta) = rvar(\mathcal{C})$ such that:

- $t\theta = u\theta$ for all $t = u \in \mathcal{C}$;
- $T\theta \vdash u\theta$ for all $T \Vdash u \in \mathcal{C}$;
- for all $(\forall X.\, t \neq u) \in \mathcal{C}$, then $t\theta$ and $u\theta$ are not unifiable (even renaming the variables of X with fresh variables); and
- $[\![\Phi\theta]\!]_G = 1$ for every formula $\Phi \in \mathcal{C}$.

Example 5 *Let* $\mathcal{C} = \{\mathcal{I}_0 \cup \{u_1\} \Vdash v_1 \wedge \Phi_D \wedge \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2 \wedge \Phi_S \wedge \neg\mathsf{route}(x_L)\}$ *with:*

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

We have that $\mathcal{C}$ *is a constraint system, and* $\sigma = \{^{id}/_{x_{id}}, ^{X}/_{x_a}, ^{[W;S]}/_{x_l}, ^{[X;W;S]}/_{x_L}\}$ *is a solution of the constraint system* $\mathcal{C}$ *for graph* G_0.

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 $n \in V$;
- $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in V$ 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 $T \Vdash u \in \mathcal{C}$.

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

Symbolic transitions are defined in Figure 3, they mimic concrete ones. In particular, for the second 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_j, and the set L that reject the message t because the condition ϕ is not fulfilled.

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 is formally proved in Appendix A.

Example 6 *For example, executing the same transitions as in Example 4 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 5.*

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. 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* θ *a solution 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.

Each concrete transition can be matched by a symbolic one.

$$\textsc{Comm}_s \quad (\lfloor \mathsf{out}(t).P \rfloor_n \cup \{\lfloor \mathsf{in}\ u_i[\Phi_i].P'_i \rfloor_{n_i} \mid i \in I\} \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \{\lfloor \mathsf{in}\ u_k[\Phi_k].P'_k \rfloor_{n_k} \mid k \in K \cup L\} \cup \{\lfloor P'_j \rfloor_{n_j} \mid j \in J\} \cup \mathcal{P}; \mathcal{S}; \mathcal{I}'; \mathcal{C}')$$

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 $i \in I$,
- $\mathcal{C}' = \mathcal{C} \wedge \{t = u_j \wedge \Phi_j \mid j \in J\} \wedge \{\forall (var(u_k) \setminus rvar(\mathcal{C})) \,.\, t \neq u_k \mid k \in K\} \wedge \{t = u_l\alpha_l \wedge \neg\Phi_l\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u_l) \setminus rvar(\mathcal{C})$ 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.

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

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

$$\textsc{Read-Then}_s \quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge t = u) \quad \text{where } \lfloor t \rfloor_n \in \mathcal{S}$$

$$\textsc{Read-Else}_s \quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor Q \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge \{\forall X \,.\, t \neq u \mid \lfloor t \rfloor_n \in \mathcal{S}\}) \text{ where } X = var(u) \setminus rvar(\mathcal{C})$$

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

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

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

$$\textsc{Repl}_s \quad \lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \rightarrow^s_{G,\mathcal{M}} \lfloor P\alpha \rfloor_n \cup \lfloor !P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}$$

where α is a renaming of the bound variables of P that are not in $rvar(\mathcal{C})$.

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

Figure 3. Symbolic transition system.

Lemma 1 (Completeness) *Let G 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 $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$.*

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, each symbolic transition can be instantiated in a concrete one.

Lemma 2 (Soundness) *Let G 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.*

We deduce that checking for a concrete attack can be reduced to checking for a symbolic one.

Proposition 1 *Let G 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.

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 7 *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 5), so there is an $\{n_I\}$-attack on K_0 for G_0.

4 Decidability result

In the remaining of the paper, we assume the fixed signature $(\mathcal{S}_1, \mathcal{F}_1)$ (defined in Example 1) for list, concatenation, mac and encryption. We also assume its associated deduction system $\vdash$ defined in Figure 1.

Simple properties like secrecy are undecidable when considering an unbounded number of role executions, even for classical protocols [12]. 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 as its 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 $\mathcal{I}'$ a finite set of terms (the intruder is given all the node names in addition to its usual initial knowledge).

Our second main contribution is to show that accessibility properties are decidable for finite processes of our process algebra, which models secure routing protocols, for a bounded number of sessions. We actually provide two decision procedures, according to whether the network is *a priori* given or not. In case the network topology is not fixed in advance, our procedure allows to automatically discover 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 an hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists a graph G 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 an hole, G be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. 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 guaranty 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 [24]. The (NP) decision procedures proposed for proving Theorems 1 and 2 involve several steps, with many common ingredients.

Step 1. Applying Proposition 1, it is sufficient to decide whether there exists a sequence of symbolic transitions (and a graph G if G is not fixed)

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

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 remains 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. It has been shown in [11] 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 variables only. We have extended this result to our generalized notion of constraint systems, *i.e.* with disequality tests and formula of $\mathcal{L}_{\mathsf{route}}$, and for an intruder knowledge with an infinite number of names.

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 [11] since we are left with (non solved) disequality constraints and formulas. The key step consists in showing that we can bound (polynomially) the size of the lists in a minimal attack.

The two last steps are developed in the two following (sub-)sections.

4.1 Solving constraint systems with an infinite number of names

A constraint system $\mathcal{C}$ can be split in four disjoint sets $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ 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. Considering a most general unifier θ of the unification constraints in $\mathcal{C}_1$, we are left to decide the existence of a solution for $(\mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4)\theta$. We say that a constraint system $\mathcal{C}$ is a *deduction constraint system* if all 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. First, we have that $\mathcal{C}_2\theta$ is a deduction constraint system. The goal of this section is to show that we can assume that $\mathcal{C}_2\theta$ is in *solved form*.

It has been shown in [11] that the existence of a solution of a deduction constraint system can be reduced to the existence of a solution of a *solved* deduction constraint system by applying (a variant of) the transformation rules presented in Figure 4.

All the rules are indexed by a substitution. When there is no index then the identity substitution is implicitly assumed. We write $\mathcal{C} \rightsquigarrow_\sigma^n \mathcal{C}'$ 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 $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. We write $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ if $\mathcal{C} \rightsquigarrow_\sigma^n \mathcal{C}'$ for some $n \geq 1$, or if $\mathcal{C}' = \mathcal{C}$ and σ is the identity substitution.

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. It has been shown in [11] that a deduction constraint system admits a solution if, and only if, the transformation rules yield (in nondeterministically polynomial time) a solved constraint.

However, the result of [11] 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. Moreover, we have adapted the simplification rules to our signature with mac and lists.

Definition 4 *Let $\mathcal{C}$ be a deduction constraint system where all left hand sides of constraints are finite, and $\mathcal{I}_0$ be a (possibly infinite) set of names. 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 $\overline{\mathcal{C}}^{\mathcal{I}_0}$.

We show that when solving a special constraint system $(\mathcal{C}, \mathcal{I}_0)$, it is sufficient to apply the transformation rules to $\mathcal{C}$, following a well-chosen strategy in order to get a polynomial bound on the length of simplification sequences. We consider the following strategy $\mathcal{S}$:

- apply eagerly R_4 and postpone R_1 as much as possible
- apply the substitution rules eagerly (as soon as they are enabled). This implies that all substitution rules are applied at once, since the rules $\mathsf{R}_1, \mathsf{R}_4, \mathsf{R}_f$ cannot enable a substitution.
- when R_4 and substitution rules are not enabled, apply R_f to the constraint whose right hand side is maximal (in size).

For ordinary constraint systems, the strategy $\mathcal{S}$ is complete and yields derivations of polynomial length (see Section 4.7 in [11]). It remains to show that the procedure also works for special constraint systems.

Theorem 3 *Let $(\mathcal{C}_0, \mathcal{I}_0)$ be a special constraint system, Φ a set of formulas and disequality constraints, and θ be a substitution.*

1. *(Correctness) 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 $\sigma\theta$ 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 $\theta = \sigma\theta'$, $\mathcal{C}_0 \rightsquigarrow_\sigma^* \mathcal{C}'$ by a derivation in $\mathcal{S}$, and θ' is a solution for $(\mathcal{C}', \mathcal{I}_0)$ and $\Phi\sigma$.*
3. *(Termination) If $\mathcal{C}_0 \rightsquigarrow_\sigma^n \mathcal{C}'$ 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 $\mathcal{C}_0$.*

The proof of Theorem 3 is mainly an adaptation of the result in [11] and relies on the following lemma, which intuitively states that adding an infinite set of disjoint names does not provide an additional deduction power to the intruder.

Lemma 3 *Let T be a set of terms that contains at least one constant, u a term and E a set of constants such that $St(T \cup \{u\}) \cap E = \emptyset$. If $T \cup E \vdash u$, then we have that $T \vdash u$.*

$$
\begin{array}{llll}
\mathsf{R}_1 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow & \mathcal{C} \qquad \text{if } T \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u \\
\mathsf{R}_2 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow_\sigma & \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathsf{mgu}(t,v), t \in St(T), v \in St(u), t \neq v, t, v \text{ not variables} \\
\mathsf{R}_3 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow_\sigma & \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathsf{mgu}(t_1,t_2), t_1, t_2 \in St(T), t_1 \neq t_2, t_1, t_2 \text{ not variables} \\
\mathsf{R}_4 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow & \bot \qquad \text{if } var(T,u) = \emptyset \text{ and } T \not\vdash u \\
\mathsf{R}_f & \mathcal{C} \wedge T \Vdash f(u,v) & \rightsquigarrow & \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v \qquad \text{for } f \in \{\langle\rangle, ::, \mathsf{hmac}, \mathsf{enc}\}
\end{array}
$$

Figure 4. Simplification rules

4.2 Bounding the size of minimal attacks

Applying the technique described in the previous section, we are left to decide the existence of a solution for a constraint system $\mathcal{C} = \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_2$ is a solved deduction constraint system, $\mathcal{C}_3$ contains disequality constraints, and $\mathcal{C}_4$ contains formulas of $\mathcal{L}_{\mathsf{route}}$. We can show that the size of the constraint system $\mathcal{C}$ obtained after the successive transformations remains polynomially bounded in the size of the initial configuration.

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 4 *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 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.*

We then show that it is possible to find a solution in which lists are polynomially bounded. We need to prove two separate lemmas, according to whether the network topology is fixed or not. In case the network topology is not fixed, we show that we can bound the size of an attack, possibly by changing the graph.

Lemma 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, then there exists a graph G' 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 $\mathcal{C}, \Phi_1$ and Φ_2.*

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.

Lemma 6 *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 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 $\mathcal{C}, \Phi_1, \Phi_2$ and G.*

The proofs of Lemma 5 and 6 use the fact that disequality constraints can be satisfied using fresh node name and that the predicates of the logic $\mathcal{L}_{\mathsf{route}}$ check only a finite number of nodes. Combining the lemmas, we get that it is possible to bound the size of a (minimal) solution. For Theorem 1, we conclude by further noticing that nodes that do not occur explicitly in a solution σ can be removed from the graph.

5 Conclusion

Using our symbolic semantic, we have shown that, for general processes that can broadcast and perform some correctness check in addition to the usual pattern matching, existence of attacks can be reduced to existence of a solution for (generalized) constraint systems. As an illustration, for a large class of processes without replication that captures routing protocol like SRP applied on DSR, 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 in particular to automatically discover whether a particular network topology may allow malicious nodes to mount an attack.

As future work, we plan to extend our results to other cryptographic primitives (e.g. signatures and public keys) in order to model more protocols. 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. We also plan to consider how to model changes in the network topology to analyze the security of route updates.

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. *Secure Routing in Multi-hop Wireless Networks*. PhD thesis, Budapest University of Technology and Economics, 2009.

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

[5] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. 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.

[6] M. Arnaud, V. Cortier, and S. Delaune. Modeling and verifying ad hoc routing protocol. In *Proc. 4th International Workshop on Security and Rewriting Techniques (SecReT'09)*, pages 33–46, Port Jefferson, NY, USA, 2009.

[7] 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.

[8] 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.

[9] 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.

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

[11] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 2010. To appear.

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

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

[14] 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.

[15] 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.

[16] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

A Proofs of Section 3

We show in Lemma 7 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 (Lemma 1) and sound (Lemma 3) when considering ground configurations.

Lemma 7 *Let G 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 $var(\mathcal{I}) \cup fv(\mathcal{P}) \cup var(\mathcal{S}) \subseteq rvar(\mathcal{C})$.

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 transition rules $\textsc{Store}_s$, $\textsc{Par}_s$, $\textsc{Repl}_s$, and $\textsc{New}_s$. Indeed, in these cases, we have that $\mathcal{C}' = \mathcal{C}$, $\mathcal{I}' = \mathcal{I}$, and $fv(\mathcal{P}) \cup var(\mathcal{S}) = fv(\mathcal{P}') \cup var(\mathcal{S}')$.

Now, we consider the remaining rules in turn.

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

  $$K_s = (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P \rfloor_n \cup \mathcal{Q}; \mathcal{S}; \mathcal{I}; \mathcal{C} \wedge t = u) = K'_s$$

  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 $var(t) \subseteq var(\mathcal{S}) \subseteq rvar(\mathcal{C})$ (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:

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

  Consequently, we have that:

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

  Since $rvar(\mathcal{C}) \cup var(u) = rvar(\mathcal{C} \cup \{t = u\})$, we deduce that the resulting symbolic configuration K'_s is also ground.

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

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

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

  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:

  $$fv(Q) \cup fv(\mathcal{Q}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C}).$$

  Since $rvar(\mathcal{C}') = rvar(\mathcal{C})$, we have that:

  $$fv(Q) \cup fv(\mathcal{Q}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C}').$$

  The resulting configuration K'_s is ground.

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

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

  It is easy to see that $\mathcal{C}'$ is still a constraint system. Moreover, since $\mathcal{I}' = \mathcal{I}$ and $\mathcal{C}' \setminus \mathcal{C}$ 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:

  $$fv(P) \cup fv(\mathcal{Q}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C}).$$

  Since $rvar(\mathcal{C}') = rvar(\mathcal{C})$, we have that:

  $$fv(P) \cup fv(\mathcal{Q}) \cup var(\mathcal{S}) \cup var(\mathcal{I}) \subseteq rvar(\mathcal{C}').$$

  Thus, the configuration K'_s is ground.

- Rule $\textsc{If-Else}_s$. Similar to the previous case.

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

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

  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. Moreover, since $var(\mathcal{I}) \subseteq rvar(\mathcal{C})$ (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:

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

  Since $rvar(\mathcal{C}') = rvar(\mathcal{C}) \cup var(u)$, we easily deduce that the symbolic configuration K'_s is ground.

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

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

  where:

  - $\mathcal{P}_I = \{\lfloor \mathsf{in}\ u_i[\Phi_i].P_i \rfloor_{n_i} \mid i \in I\}$,
  - $\mathcal{P}_J = \{\lfloor P_j \rfloor_{n_j} \mid j \in J\}$

– $\mathcal{P}_{K,L} = \{\lfloor \mathsf{in}\ u_k[\Phi_k].P_k \rfloor_{n_k} \mid k \in K \cup L\}$
– $\mathcal{C}_J = \{t = u_j \wedge \Phi_j \mid j \in J\}$,
– $\mathcal{C}_K = \{\forall var(u_k) \setminus rvar(\mathcal{C}) \,.\, t \neq u_k \mid k \in K\}$,
– $\mathcal{C}_L = \{t = u_l\alpha_l \wedge \neg\Phi_l\alpha_l \mid l \in L\}$.

$\lfloor P' \rfloor_{n'} \in \mathcal{Q}$ 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 $var(u_l) \setminus rvar(\mathcal{C})$ 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 $var(t) \subseteq rvar(\mathcal{C})$. This is indeed the case since K_s is ground and $var(t) \subseteq fv(\mathcal{P})$. 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:

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

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

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

Lemma 1 (Completeness) *Let G 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 $K_s \to^s_{G,\mathcal{M}} K'_s$.*

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 $K_s \to^s_{G,\mathcal{M}} K'_s$. Thanks to Lemma 7, we easily deduce that K'_s is ground.

- Rule Par. We have that:

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

  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 $K_s \to^s_{G,\mathcal{M}} K'_s$ (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:

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

  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 $bv(P_s) \setminus rvar(\mathcal{C})$. 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 $K_s \to^s_{G,\mathcal{M}} K'_s$ (with the 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 New. We have that:

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

  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 $K_s \to^s_{G,\mathcal{M}} K'_s$ (with the New$_s$ rule), θ is a solution of $\mathcal{C}$ and K'_c is the θ-concretization of K'_s.

- Rule Store. We have that:

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

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 StoreS rule), θ is a solution of $\mathcal{C}$ and K'_c is the θ-concretization of K'_s.

- Rule Read-Then. We have that:

$$K_c = (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q \rfloor_n \mathcal{Q}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I}) \rightarrow_{G,\mathcal{M}} (\lfloor P\sigma \rfloor_n \cup \mathcal{Q}; \lfloor t \rfloor_n \cup \mathcal{S}; \mathcal{I}) = K'_c$$

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 Read-Thens 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 $(t_s\theta)\sigma = t\sigma = t = u\sigma$. 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:

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

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 $\mathcal{C}' = \mathcal{C} \wedge \{\forall var(u_s) \setminus rvar(\mathcal{C}).t_s \neq u_s \mid \lfloor t_s \rfloor_n \in \mathcal{S}\}$.

We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$ (with the Read-Elses rule). Now, let us show that θ is a solution of $\mathcal{C}'$. Let $\forall var(u_s) \setminus rvar(\mathcal{C}).t_s \neq u_s$ be a disequation in $\mathcal{C}' \setminus \mathcal{C}$. 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-Thens 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 Ins 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

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

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 $\mathcal{I}' = \mathcal{I}$.

Since K_s is a symbolic configuration whose θ-concretization is K_c, we have that:

$$K_s = (\lfloor \mathsf{out}(t_s).P_s \rfloor_n \cup \{\lfloor \mathsf{in}\ u_j^s[\Phi_j^s].P_j^s \rfloor_{n_j} | j \in J\} \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

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 $u_j^s\theta = u_j$, $\Phi_j^s\theta = \Phi_j$, and $P_j^s\theta = P_j$.

Let

– $\mathcal{P}^s_{K,L} = \{\lfloor \mathsf{in}\ u_k^s[\Phi_k^s].P_k^s \rfloor_{n_k} \in \mathcal{Q}_s \mid (n_k, n) \in E\}$,
– $\mathcal{Q}'_s$ be such that $\mathcal{Q}_s = \mathcal{P}^s_{K,L} \cup \mathcal{Q}'_s$,
– $K = \{k \mid \lfloor \mathsf{in}\ u_k^s[\Phi_k^s].P_k^s \rfloor_{n_k} \in \mathcal{P}^s_{K,L}$ and $\mathsf{mgu}(t_s\theta, u_k^s\theta) = \bot\}$,
– $L = \{l \mid \lfloor \mathsf{in}\ u_l^s[\Phi_l^s].P_l^s \rfloor_{n_k} \in \mathcal{P}^s_{K,L}$ and $\sigma'_l = \mathsf{mgu}(t_s\theta, u_l^s\theta)$ exists, and $\neg[\![(\Phi_l^s\theta)\sigma'_l]\!]_G = 1\}$.

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:

– $\mathcal{P}^s_J = \{\lfloor P_j^s \rfloor_{n_j} \mid j \in J\}$,
– $\mathcal{C}' = \mathcal{C} \wedge \mathcal{C}_J \wedge \mathcal{C}_K \wedge \mathcal{C}_L$,
– $\mathcal{C}_J = \{t_s = u_j^s \mid j \in J\}$,
– $\mathcal{C}_K = \{\forall var(u_k) \setminus rvar(\mathcal{C}) \,.\, t_s \neq u_k^s \mid k \in K\}$,
– $\mathcal{C}_L = \{t_s = u_l^s\alpha_l \wedge \neg\Phi_l^s\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u_l^s) \setminus rvar(\mathcal{C})$ 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 $K_s \to^s_G K'_s$. 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 $\sigma_l = \mathsf{mgu}(t, (u_l^s\theta)\alpha_l)$ for any $l \in L$ ($\sigma'_l = \alpha_l\sigma_l$). 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. $t_s\theta' = u_j^s\theta'$ 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:
$$(u_j^s\theta') = (u_j^s\theta)\sigma = (u_j^s\theta)\sigma_j = u_j\sigma_j = t$$
– θ' is a solution of $\mathcal{C}_K$, i.e. θ' satisfies $\forall var(u_k) \setminus rvar(\mathcal{C}) \,.\, t_s \neq u_k^s$ for any $k \in K$. This is true since $dom(\theta) = rvar(\mathcal{C})$, and $\mathsf{mgu}(t_s\theta, u_k^s\theta) = \bot$ for any $k \in K$.
– θ' is a solution of $\mathcal{C}_L$, i.e. $t_s\theta' = (u_l^s\alpha_l)\theta'$ and $[\![(\Phi_l^s\alpha_l)\theta']\!]_G = 0$ for any $l \in L$. We have that $t_s\theta' = (t_s\theta)\sigma = t\sigma = t$ and $(u_l^s\alpha_l)\theta' = ((u_l^s\alpha_l)\theta)\sigma_l = ((u_l^s\theta)\alpha_l)\sigma_l = t$ (by definition of σ_l). Moreover we have that $(\Phi_l^s\alpha_l)\theta' = ((\Phi_l^s\alpha_l)\theta)\sigma_l = ((\Phi_l^s\theta)\alpha_l)\sigma_l = (\Phi_l^s\theta)\sigma'_l$. Hence, we have that $[\![(\Phi_l^s\alpha_l)\theta']\!]_G = 0$ for any $l \in L$.

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

– $P_s\theta' = (P_s\theta)\sigma = P_s\theta = P$,
– $P_j^s\theta' = (P_j^s\theta)\sigma = (P_j^s\theta)\sigma_j = P_j\sigma_j$ 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. □

Lemma 3 (Soundness) *Let G 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 $K_s \to^s_{G,\mathcal{M}} K'_s$. 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 $K_c \to_{G,\mathcal{M}} K'_c$, 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 $rvar(\mathcal{C})$, 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 $K_c \to_{G,\mathcal{M}} K'_c$. We distinguish several cases, depending on the rule involved in the transition $K_s \to^s_{G,\mathcal{M}} K'_s$.

- Rule $\textsc{Par}_s$. We have that:
$$K_s = (\lfloor P_1^s | P_2^s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) \to^s_{G,\mathcal{M}} (\lfloor P_1^s \rfloor_n \cup \lfloor P_2^s \rfloor_n \cup \mathcal{Q}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}) = K'_s$$
Since K'_c (resp. K_c) is the θ'-concretization (resp. θ-concretization) of K'_s (resp. K_s), we have that:
– $K'_c = (\lfloor P_1^s\theta' \rfloor_n \cup \lfloor P_2^s\theta' \rfloor_n \cup \mathcal{Q}_s\theta'; \mathcal{S}_s\theta'; \mathcal{I}_s\theta')$,
– $K_c = (\lfloor P_1^s\theta | P_2^s\theta \rfloor_n \cup \mathcal{Q}_s\theta; \mathcal{S}_s\theta; \mathcal{I}_s\theta)$.

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

- Rule REPL$_s$. We have that:

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

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 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 NEW$_s$. We have that:

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

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 NEW rule).

- Rule STORE$_s$. We have that:

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

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

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

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

- Rule READ-THEN$_s$. We have that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$, i.e.:

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

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 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 READ-THEN rule).

- Rule READ-ELSE$_s$. We have that:

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

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

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 $rvar(\mathcal{C}') = rvar(\mathcal{C})$, 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:

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

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 $rvar(\mathcal{C}') = rvar(\mathcal{C})$, 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 $K_c \to_{G,\mathcal{M}} K'_c$ by the If-Then rule.

- Rule If-Else$_s$. This case is similar to the previous one.

- Rule In$_s$. We have that:

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

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 $dom(\sigma) = var(u_s\theta)$. 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 $fv(\mathcal{Q}_s) \cup var(\mathcal{S}_s) \cup var(\mathcal{I}_s) \subseteq dom(\theta)$, 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 $K_c \to_{G,\mathcal{M}} K'_c$ by the In rule.

- Rule Comm$_s$. We have that $K_s \to^s_{G,\mathcal{M}} K'_s$, i.e.:

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

where:

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

– $I = J \uplus K \uplus L$,

– $\mathcal{P}^s_J = \{\lfloor P^s_j \rfloor_{n_j} \mid j \in J\}$,

– $\mathcal{P}^s_{K,L} = \{\lfloor \mathsf{in}\ u_k[\Phi^s_k].P^s_k \rfloor_{n_k} \mid k \in K \uplus L\}$,

– $\mathcal{C}' = \mathcal{C} \wedge \mathcal{C}_J \wedge \mathcal{C}_K \wedge \mathcal{C}_L$,

– $\mathcal{C}_J = \{t_s = u^s_j \wedge \Phi^s_j \mid j \in J\}$,

– $\mathcal{C}_K = \{\forall y \in var(u^s_k) \setminus rvar(\mathcal{C}) \,.\, t_s \neq u^s_k \mid k \in K\}$,

– $\mathcal{C}_L = \{t_s = u^s_l\alpha_l \wedge \neg\Phi^s_l\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u^s_l) \setminus rvar(\mathcal{C})$ 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 $K_c \to_{G,\mathcal{M}} K'_c$. 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 $K_c \to_G K'_c$, 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 $var(u^s_i)$, whereas for any $i \in L$, σ_i is the restriction of σ to the variables $var(u^s_i\alpha_i)$. Lastly, $dom(\sigma_i) = \emptyset$ 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 $rvar(\mathcal{C})$, 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 $\Phi^s_i\theta' = (\Phi^s_i\theta)\sigma = (\Phi^s_i\theta)\sigma_i = (\Phi^s_i\theta)\overline{\sigma}_i$. 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 $\lfloor \mathsf{in}\ u^s_i[\Phi^s_i].P^s_i \rfloor_{n_i} \in \mathcal{P}^s_I$, $(n, n_i) \in E$, $\overline{\sigma}_i = \mathsf{mgu}(t\theta, u^s_i\theta)$ exists, and $[\![(\Phi^s_i\theta)\overline{\sigma}_i]\!]_G = 1$. 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 $\forall y \in var(u_i^s) \setminus rvar(\mathcal{C}) \,.\, t_s \neq u_i^s$. 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. □

The symbolic transition system is complete and sound with respect to the concrete transition system.

Proposition 1 *Let G 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,*

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

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

Proof. 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 Lemma 1 recursively, we deduce that there exists a ground symbolic configuration K'_s and a substitution θ' such that:

- $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}; \emptyset) \rightarrow_G^{s*} K'_s$, 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 $\mathcal{C}' \wedge u = \mathsf{error}$. This allows us to conclude.

Conversely, assume that

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

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 7), 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 7, we know that the symbolic configurations involved in this derivation are ground.

Hence, by applying recursively Lemma 3, 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. □

B Proofs of Section 4.1

It has been shown in [11] 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 variables only.

We extend in Theorem 3 this result to our notion of constraint system and for an intruder knowledge with an infinite number of names. We use the simplification rules in Figure 4.

B.1 Some Preliminaries

Let $\mathcal{C}$ be a constraint system and $\mathcal{I}$ be a possible infinite set of names. We want to apply the simplification rules to $\overline{\mathcal{C}}^I$. We show that it is enough to apply the simplification rules to $\mathcal{C}$ and add $\mathcal{I}$ in the left-hand side of each constraint of $\mathcal{C}$. This will give us an NP decision procedure.

Lemma 8 *Let T be a set of terms, u be a term and E be a set of constants such that $St(T \cup \{u\}) \cap E = \emptyset$. We consider an inference system where no constant of E appears in the rules, and side conditions are preserved by arbitrary substitution of the constants of E. If $T \cup E \vdash u$, and T contains at least one constant, then $T \vdash u$.*

Proof. Let π be a proof of $T \cup E \vdash u$, and a be a constant in T. Let δ be the operation replacing all constants of E by a. The operation δ is extended to terms, proofs and substitutions in the obvious way. We show that $\pi\delta$ is a proof of $T \vdash u\delta$. We reason by induction on π.

- If π is the application of the axiom rule, then $u \in T \cup E$. Either $u \in T$, or $u \in E$. In the latter case, $u\delta = a$, and $a \in T$. Otherwise, $u \in St(T)$, thus $u\delta = u$. Consequently, in both cases, $u\delta \in T$, so $\pi\delta$ is a proof of $T \vdash u\delta$.

- Suppose that the last inference rule R applied in π is of the form

$$\frac{T \cup E \vdash v_1 \quad \dots \quad T \cup E \vdash v_n}{T \cup E \vdash v}\ s$$

  There exists θ such that $u = v\theta$, and $[\![s\theta]\!] = 1$. We have that π_i is proof of $T \cup E \vdash v_i\theta = u_i$ for $1 \leq i \leq n$. We can apply the induction hypothesis on each proof π_i. Hence, we have that $\pi_i\delta$ is a proof of $T \vdash u_i\delta$. Now we apply the inference rule R to $T \vdash u_1\delta$, $\dots$, $T \vdash u_n\delta$. For this, we have to show that there exists a substitution θ' such that $v_i\theta' = (v_i\theta)\delta$ for each $i \in \{1, \dots, n\}$. As the inference rules do not contain any constant of E, we have that $(v_i\theta)\delta = v_i(\theta\delta)$ and thus such a substitution θ' exists. It is sufficient to choose $\theta' = \theta\delta$. To conclude, we only have to check the side condition, i.e. to check that $[\![s\theta']\!] = 1$. We consider an inference system where side conditions are preserved by substituting the constants of E. We easily deduce that $[\![s\theta']\!] = 1$ from the fact that $[\![s\theta]\!] = 1$. Hence, we can apply the rule R, obtaining a proof $\pi\delta$ of $T \vdash v\theta'$, i.e. a proof of $T \vdash u\delta$ since $v\theta' = v(\theta\delta) = (v\theta)\delta = u\delta$.

By induction, we have shown that if $T \cup E \vdash u$, then $T \vdash u\delta$. If $St(u) \cap E = \emptyset$, then $u\delta = u$. Consequently, if $T \cup E \vdash u$ with $St(u) \cap E = \emptyset$, then $T \vdash u$. □

Remark: Lemma 8 holds in particular for the inference system given in Figure 1. Hence, Lemma 3 is a corollary of Lemma 8.

Lemma 9 *Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system. If $\mathcal{C} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \mathcal{C}'$ then $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \overline{\mathcal{C}'}^{\mathcal{I}}$, and $(\mathcal{C}', \mathcal{I})$ is a special constraint system.*

Proof. We reason by case study over R, the simplification rule used in $\mathcal{C} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \mathcal{C}'$.

- If $\mathcal{C}'$ is obtained by applying R_1, then $\mathcal{C} = \mathcal{C}' \wedge T \Vdash u$ 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,

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

  Hence we have that

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

  So we can apply the simplification rule R_1 to $\overline{\mathcal{C}}^{\mathcal{I}}$, and $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}_1}{\rightsquigarrow}_\sigma \overline{\mathcal{C}'}^{\mathcal{I}}$.

  Furthermore, $St(\mathcal{C}') \subseteq St(\mathcal{C})$, hence $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$, since $(\mathcal{C}, \mathcal{I})$ is a special constraint system. Thus, $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

- If $\mathcal{C}'$ is obtained by applying R_2, then $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$, $\mathcal{C}' = \mathcal{C}_0\sigma \wedge T\sigma \Vdash u\sigma$, where $\sigma = \mathsf{mgu}(t, v)$, $t \in St(T)$, $v \in St(u)$, $t \neq v$, t, v not variables.

  In that case, $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$, we can thus apply simplification rule R_2 to $\overline{\mathcal{C}}^{\mathcal{I}}$, and $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}_2}{\rightsquigarrow}_\sigma \overline{\mathcal{C}_0}^{\mathcal{I}}\sigma \wedge (T \cup \mathcal{I})\sigma \Vdash u\sigma = \overline{\mathcal{C}'}^{\mathcal{I}}$

  Furthermore, to show that $(\mathcal{C}', \mathcal{I})$ is a special constraint system, it remains to show that $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$. As $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, for every $x \in dom(\sigma)$, $St(x\sigma) \cap \mathcal{I} = \emptyset$. So $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$, i.e. $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$.

- $\mathcal{C}'$ is obtained by applying R_3. This case is similar to the previous one.

- If $\mathcal{C}'$ is obtained by applying some rule R_f, then $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash f(u, v)$, and $\mathcal{C}' = \mathcal{C}_0 \wedge T \Vdash u \wedge T \Vdash v$.

  In that case, $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash f(u, v)$, we can apply rule R_f, and $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}_f}{\rightsquigarrow}_\sigma \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u \wedge T \cup \mathcal{I} \Vdash v$.

  Furthermore, $St(\mathcal{C}') \subseteq St(\mathcal{C})$, hence $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$, since $(\mathcal{C}, \mathcal{I})$ is a special constraint system. Thus, $(\mathcal{C}', \mathcal{I})$ is a special constraint system.

- If $\mathcal{C}'$ is obtained by applying rule R_4, then $\mathcal{C}' = \bot$, $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$, $var(T, u) = \emptyset$ and $T \nVdash u$.

  Consequently, $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$. Furthermore, $var(T \cup \mathcal{I}, u) = var(\mathcal{I}) \cup var(T, u) = \emptyset$ because $\mathcal{I}$ is a set of names. Thanks to Lemma 3, we also have that $T \cup \mathcal{I} \nVdash u$, because $St(T, u) \cap \mathcal{I} = \emptyset$ (as $(\mathcal{C}, \mathcal{I})$ is a special constraint system). So we can apply Rule R_4: $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}_4}{\rightsquigarrow} \bot = \overline{\bot}^{\mathcal{I}}$ and $(\bot, \mathcal{I})$ is a special constraint system.

□

Lemma 10 *Let $(\mathcal{C},\mathcal{I})$ be a special constraint system such that $\overline{\mathcal{C}}^{\mathcal{I}} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \mathcal{C}'_s$, then there exists $\mathcal{C}'$ such that $\mathcal{C}'_s = \overline{\mathcal{C}'}^{\mathcal{I}}$ and $\mathcal{C} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \mathcal{C}'$, furthermore $(\mathcal{C}',\mathcal{I})$ is a special constraint system.*

Proof. We reason by case study over the simplification rule R used in $\overline{\mathcal{C}}^{I} \overset{\mathsf{R}}{\rightsquigarrow}_\sigma \mathcal{C}'_s$.

- If $\mathcal{C}'_s$ is obtained by applying R_1, then $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}'_s \wedge T \Vdash u$ and $T \cup \{x \mid (T' \Vdash x) \in \overline{\mathcal{C}}^{\mathcal{I}}, T' \subsetneq T\} \vdash u$.

  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,

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

  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 3, we obtain that

  $$T' \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u.$$

  Consequently, we can apply rule R_1 to $\mathcal{C}$: $\mathcal{C} \overset{\mathsf{R}_1}{\rightsquigarrow} \mathcal{C}'$. Furthermore, $St(\mathcal{C}') \subseteq St(\mathcal{C})$, hence $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$, since $(\mathcal{C},\mathcal{I})$ is a special constraint system. Thus, $(\mathcal{C}',\mathcal{I})$ is a special constraint system.

- If $\mathcal{C}'_s$ is obtained by applying R_2, then $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$, $\mathcal{C}'_s = \overline{\mathcal{C}_0}^{\mathcal{I}}\sigma \wedge T\sigma \cup \mathcal{I} \Vdash u\sigma$, where $\sigma = \mathsf{mgu}(t,v)$, $t \in St(T \cup \mathcal{I})$, $v \in St(u)$, $t \neq v$, t, v not variables.

  In that case, $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$. We only need to show that $t \in St(T)$ to apply simplification rule R_2 to $\mathcal{C}$ with substitution σ. It is sufficient to show that $t \notin St(\mathcal{I}) = \mathcal{I}$. By contradiction, suppose that $t \in \mathcal{I}$. We know that $t\sigma = v\sigma$. If $t \in \mathcal{I}$, $t\sigma = t \in \mathcal{I}$. But $v \in St(\mathcal{C})$, so $St(v) \cap \mathcal{I} = \emptyset$. If $v\sigma \in \mathcal{I}$, as $\mathcal{I}$ is a set of names, this implies that v is a variable, which yields a contradiction.

  Consequently, $t \in St(T)$, and we can apply simplification rule R_2 to $\mathcal{C}$:
  $\mathcal{C} \overset{\mathsf{R}_2}{\rightsquigarrow}_\sigma \mathcal{C}_0\sigma \wedge T\sigma \Vdash u\sigma = \mathcal{C}'$

  Furthermore, to show that $(\mathcal{C}',\mathcal{I})$ is a special constraint system, it remains to show that $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$. As $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, for every $x \in dom(\sigma)$, $St(x\sigma) \cap \mathcal{I} = \emptyset$. So $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$, i.e. $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$.

- If $\mathcal{C}'_s$ is obtained by applying R_3, then $\overline{\mathcal{C}}^{\mathcal{I}} = \overline{\mathcal{C}_0}^{\mathcal{I}} \wedge T \cup \mathcal{I} \Vdash u$, $\mathcal{C}'_s = \overline{\mathcal{C}_0}^{\mathcal{I}}\sigma \wedge T\sigma \cup \mathcal{I} \Vdash u\sigma$, where $\sigma = \mathsf{mgu}(t_1,t_2)$, $t_1, t_2 \in St(T \cup \mathcal{I})$, $t_1 \neq t_2$, t_1, t_2 not variables.

  In that case, $\mathcal{C} = \mathcal{C}_0 \wedge T \Vdash u$. We only need to show that $t_1, t_2 \in St(T)$ to apply simplification rule R_3 to $\mathcal{C}$ with substitution σ. It is sufficient to show that $t_1, t_2 \notin St(\mathcal{I}) = \mathcal{I}$. By contradiction, suppose that $t_1 \in \mathcal{I}$ (the same reasoning holds for $t_2 \in \mathcal{I}$ by symmetry). We know that $t_1\sigma = t_2\sigma$. If $t_2 \in \mathcal{I}$, $t_2\sigma = t_2 \in \mathcal{I}$. So $t_1 = t_2$, which is false. So $t_2 \notin \mathcal{I}$. In that case, as $t_2\sigma \in \mathcal{I}$, as $\mathcal{I}$ is a set of names, this implies that t_2 is a variable, which yields a contradiction.

  Consequently, $t_1, t_2 \in St(T)$, and we can apply simplification rule R_3 to $\mathcal{C}$:
  $\mathcal{C} \overset{\mathsf{R}_3}{\rightsquigarrow}_\sigma \mathcal{C}_0\sigma \wedge T\sigma \Vdash u\sigma = \mathcal{C}'$

  Furthermore, to show that $(\mathcal{C}',\mathcal{I})$ is a special constraint system, it is sufficient to show that $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$. As $St(\mathcal{C}) \cap \mathcal{I} = \emptyset$, for every $x \in dom(\sigma)$, $St(x\sigma) \cap \mathcal{I} = \emptyset$. So $St(\mathcal{C}\sigma) \cap \mathcal{I} = \emptyset$, i.e. $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$.

- If $\mathcal{C}'_s$ is obtained by applying some rule R_f, then $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}_0 \wedge T \Vdash f(u,v)$, and $\mathcal{C}'_s = \mathcal{C}_0 \wedge T \Vdash u \wedge T \Vdash v$.

  By definition of $\overline{\mathcal{C}}^{\mathcal{I}}$, there exists $\mathcal{C}_1, T'$ such that $\mathcal{C}_0 = \overline{\mathcal{C}_1}^{\mathcal{I}}$, $T = T' \cup \mathcal{I}$ and $\mathcal{C} = \mathcal{C}_1 \wedge T' \Vdash f(u,v)$. Consequently, we can apply the same simplification rule R_f :
  $\mathcal{C} \overset{\mathsf{R}_f}{\rightsquigarrow}_\sigma \mathcal{C}_1 \wedge T' \Vdash u \wedge T' \Vdash v = \mathcal{C}'$. Furthermore, $\overline{\mathcal{C}'}^{\mathcal{I}} = \overline{\mathcal{C}_1}^{\mathcal{I}} \wedge T' \cup \mathcal{I} \Vdash u \wedge T' \cup \mathcal{I} \Vdash v = \mathcal{C}_0 \wedge T \Vdash u \wedge T \Vdash v = \mathcal{C}'_s$
  Furthermore, $St(\mathcal{C}') \subseteq St(\mathcal{C})$, hence $St(\mathcal{C}') \cap \mathcal{I} = \emptyset$, since $(\mathcal{C},\mathcal{I})$ is a special constraint system. Thus, $(\mathcal{C}',\mathcal{I})$ is a special constraint system.

- If $\mathcal{C}'_s$ is obtained by applying rule R_4, then $\mathcal{C}'_s = \bot$, $\overline{\mathcal{C}}^{\mathcal{I}} = \mathcal{C}_0 \wedge T \Vdash u$, $var(T,u) = \emptyset$ and $T \nvdash u$.

  By definition of $\overline{\mathcal{C}}^{\mathcal{I}}$, there exists $\mathcal{C}_1, T'$ such that $\mathcal{C}_0 = \overline{\mathcal{C}_1}^{\mathcal{I}}$, $T = T' \cup \mathcal{I}$ and $\mathcal{C} = \mathcal{C}_1 \wedge T' \Vdash u$. $var(T',u) = var(T,u) = \emptyset$, and, as $T = T' \cup \mathcal{I}$ and $T \nvdash u$, we have that $T' \nvdash u$. Consequently, we can apply to $\mathcal{C}$ the simplification rule R_4: $\mathcal{C} \overset{\mathsf{R}_4}{\rightsquigarrow} \bot$. We have that $(\bot,\mathcal{I})$ is a special constraint system, and $\overline{\bot}^{\mathcal{I}} = \bot$.

B.2 Proof of Theorem 3

The following 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 3 *Let $(\mathcal{C}_0, \mathcal{I}_0)$ be a special constraint system, Φ a set of formulas and disequality constraints, and θ be a substitution.*

1. *(Correctness) 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 $\sigma\theta$ 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 $\theta = \sigma\theta'$, $\mathcal{C}_0 \rightsquigarrow_\sigma^* \mathcal{C}'$ by a derivation in $\mathcal{S}$, and θ' is a solution for $(\mathcal{C}', \mathcal{I}_0)$ and $\Phi\sigma$.*
3. *(Termination) If $\mathcal{C}_0 \rightsquigarrow_\sigma^n \mathcal{C}'$ 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 $\mathcal{C}_0$.*

Proof. We show the three items separately.
Correctness. If $\mathcal{C}_0 \rightsquigarrow_\sigma^* \mathcal{C}'$, then by applying Lemma 9 inductively, we get that $\overline{\mathcal{C}_0}^{\mathcal{I}_0} \rightsquigarrow_\sigma^* \overline{\mathcal{C}'}^{\mathcal{I}_0}$. Thanks to Theorem 4.3 in [11], we deduce that $\sigma\theta$ is an attack for Φ and $\overline{\mathcal{C}_0}^{\mathcal{I}_0}$.

Completeness. θ is a solution for $\overline{\mathcal{C}_0}^{\mathcal{I}_0}$ and Φ. By applying Theorem 4.3 in [11], there exists a constraint system $\mathcal{C}_s$ in solved form and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\overline{\mathcal{C}_0}^{\mathcal{I}_0} \rightsquigarrow_\sigma^* \mathcal{C}_s$ by a derivation in $\mathcal{S}$, and θ' is a solution for $\mathcal{C}'$ and $\Phi\sigma$. By applying Lemma 10 recursively, we get that there exists $\mathcal{C}'$ such that $\mathcal{C}_s = \overline{\mathcal{C}'}^{\mathcal{I}_0}$, and $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$. 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 $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ 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 $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ is in $\mathcal{S}$.

Termination. The strategy yields derivations of polynomial length (see Section 4.7 in [11]).

C Decidability

We restrict ourselves to finite ground processes without replication.

C.1 Preliminary results

Lemma 11 *Let $\forall Y.u \neq v$ be a disequality constraint. Let σ be a substitution with $dom(\sigma) \subseteq Y$ and such that $u\sigma = v\sigma$. Then for every ground substitution θ with $dom(\theta) \cap Y = \emptyset$, we have that $u\theta$ and $v\theta$ are unifiable. In other words, the constraint is not satisfiable.*

Proof. Let θ be a ground substitution such that $dom(\theta) \cap Y = \emptyset$. In order to conclude, we have to show that $u\theta$ and $v\theta$ are unifiable. Let $\tau = \theta \circ \sigma$. We have that:

- $(u\theta)\tau = ((u\theta)\sigma)\theta = (u\sigma)\theta$, and
- $(v\theta)\tau = ((v\theta)\sigma)\theta = (v\sigma)\theta$.

By hypothesis, we have that $u\sigma = v\sigma$. Hence, we easily conclude that $(u\theta)\tau = (v\theta)\tau$, i.e. $u\theta$ and $v\theta$ are unifiable. □

Let u be a term. We denote by $|u|$ the size of u (i.e. number of non variable symbols in u) and by $|u|_d$ the maximal depth of a variable in u. Let S be a set, we denote by $\#S$ the cardinal of S.

Lemma 12 *Let T be a set of terms and P be a set of equations between terms in T with $\sigma = \mathsf{mgu}(P)$. For every variable x, 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 5. Applying these rules on P results in a most general unifier of P in dag solved form (see [17]):

$$\sigma = \{x_1 = t_1, \ldots, x_n = t_n\}.$$

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. Hence, 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 $|x\sigma|_d < \#dom(\sigma) \cdot \max\{|t|_d \mid t \in T\}$. □

Lemma 13 *Let P and P' be two sets of equations and σ, σ' be two substitutions such that $\sigma = \mathsf{mgu}(P)$ and $\sigma' = \mathsf{mgu}(P'\sigma)$. We have that $\sigma' \circ \sigma = \mathsf{mgu}(P \cup P')$.*

Proof. First, it is easy to check that $\sigma' \circ \sigma$ is indeed an unifier of P and P'.

Now, let μ be an unifier of $P \cup P'$. In order to conclude that $\sigma' \circ \sigma$ is a most general unifier, we show that there exists μ' such that $\sigma' \circ \sigma = \mu' \circ \mu$.

$$\textsc{Delete}\quad P \cup \{s = s\} \implies P$$

$$\textsc{Dec.}\quad P \cup \{f(s_1,\dots,s_n) = f(t_1,\dots,t_n)\} \implies P \cup \{s_1 = t_1,\dots,s_n = t_n\}$$

$$\textsc{Conf.}\quad P \cup \{f(s_1,\dots,s_n) = g(t_1,\dots,t_k)\} \implies \bot \quad \text{if } f \neq g$$

$$\textsc{Coal.}\quad P \cup \{x = y\} \implies P\{x \mapsto y\} \cup \{x = y\} \quad \text{if } x, y \in var(P) \text{ and } x \neq y$$

$$\textsc{Check}\quad P \cup \{x_1 = s_1[x_2],\dots,x_n = s_n[x_1]\} \implies \bot \quad \text{if } s_i \notin \mathcal{X} \text{ for some } i \in [1\dots n]$$

$$\textsc{Merge}\quad P \cup \{x = s, x = t\} \implies P \cup \{x = s, s = t\} \quad \text{if } 0 < |s| \leq |t|$$

Figure 5. Rules for dag syntactic unification

- Since $\sigma = \mathsf{mgu}(P)$, there exists τ such that $\mu = \tau \circ \sigma$. Note also that τ is a unifier of $P'\sigma$
- Since $\sigma' = \mathsf{mgu}(P'\sigma)$, there exists τ'' such that $\tau = \tau'' \circ \sigma'$.

We have that $\mu = \tau \circ \sigma = (\tau'' \circ \sigma') \circ \sigma = \tau'' \circ (\sigma' \circ \sigma)$. This allows us to conclude that $\sigma' \circ \sigma$ is a most general unifier. □

Lemma 14 *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 $St(T\sigma) \subseteq \{t\sigma \mid t \in St(T)\}$.*

Proof. We use the rules for dag syntactic unification given in Figure 5. Applying these rules on P results in a set of equations $P' = \{x_1 = t_1,\dots,x_n = t_n\}$ in dag solved form (see [17]). 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 $\sigma = \{x_1 \mapsto t_1,\dots,x_n \mapsto t_n\}$. By inspection of the rules in Figure 5, 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 $var(t_{i_0}) \cap dom(\sigma) \subseteq \{x_{i_0+1},\dots,x_n\}$. Hence, we have that $u \in St(x_j\sigma)$ for some $j > i_0$. This yields to a contradiction. □

Lemma 4 *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 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

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

where $x_1,\dots,x_n$ are distinct variables and $var((\mathcal{C},\mathcal{I}) \wedge \Phi_1 \wedge \Phi_2) = \{x_1,\dots,x_n\} = rvar(\mathcal{C})$. We show the result by induction on $\mu(\sigma) = \#\{x \in rvar(\mathcal{C}) \mid x \text{ is not of sort } \mathsf{loc} \text{ or } \mathsf{lists} \text{ and } x\sigma \notin \mathcal{I}\}$.

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 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 $X = dom(\sigma) \setminus \{x_{i_0}\}$. Clearly, we have that $\mu(\sigma') \leq \mu(\sigma)$. 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 $x_i \notin \mathcal{X}_t$, 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. If $u\sigma'$ and $v\sigma'$ are not unifiable, we easily conclude. Otherwise, the means that $u\tau$ and $v\tau$ are unifiable. Let $\theta = \mathsf{mgu}(u\tau, v\tau)$. We distinguish 3 cases:

Case 1: $x \notin dom(\theta)$, i.e. $dom(\theta) \subseteq Y$. In such a case, we have that:

$$(u\sigma)\theta = ((u\tau)\{x \mapsto x\sigma\})\theta = (u\tau\theta)\{x \mapsto x\sigma\}$$
$$(v\sigma)\theta = ((v\tau)\{x \mapsto x\sigma\})\theta = (v\tau\theta)\{x \mapsto x\sigma\}.$$

Hence, we deduce that $u\sigma$ and $v\sigma$ are unifiable, and we obtain a contradiction.

Case 2: $x \in dom(\theta)$ and $x\theta = y \in Y$. Again, we have that:

$$(u\sigma)\theta = ((u\tau)\{x \mapsto x\sigma\})\theta = (u\tau\theta)\{y \mapsto x\sigma\}$$
$$(v\sigma)\theta = ((v\tau)\{x \mapsto x\sigma\})\theta = (v\tau\theta)\{y \mapsto x\sigma\}.$$

Hence, we deduce that $u\sigma$ and $v\sigma$ are unifiable, and we obtain a contradiction.

Case 3: $x \in dom(\theta)$ and $x\theta \notin Y$. Assume by contradiction that there exists a substitution θ' such that $u\sigma'\theta' = v\sigma'\theta'$ (i.e. σ' does not satisfy $\forall Y.u \neq v$). In such a case, we have that:

$$(u\sigma')\theta' = ((u\tau)\{x \mapsto a\})\theta' = ((u\tau)\theta')\{x \mapsto a\}$$
$$(v\sigma')\theta' = ((v\tau)\{x \mapsto a\})\theta' = ((v\tau)\theta')\{x \mapsto a\}$$

Since a is fresh, we deduce that $(u\tau)\theta' = (v\tau)\theta'$, and thus we have that $((u\tau)\theta')\{x \mapsto x\sigma\} = ((v\tau)\theta')\{x \mapsto x\sigma\}$, 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 $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$. □

Lemma 15 *Let G 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 G' be a graph 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 $\mathcal{C}$, Φ_1, Φ_2, and σ. 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. Let $G = (V, E)$ and $G' = (V', E')$.

- $[\![\mathsf{check}(a\sigma, b\sigma)]\!]_G = 1$ if, and only if, $(a\sigma, b\sigma) \in E$ with $a\sigma$ and $b\sigma$ of sort loc. Hence, 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, $c\sigma$ is of sort loc, $l\sigma$ 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'. □

Definition 5 *An* extracted list from $l = [a_1, \ldots, a_n]$ *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 denote by $names(t)$ the set of names occurring in a term t. This notation is extended as expected to sets of terms, substitutions, ...

Lemma 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, then there exists a graph G' 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 $\mathcal{C}, \Phi_1$ and Φ_2.*

Proof. We write $G = (V_G, E_G)$, $\Phi_2 = \bigwedge_n \forall Y_n.u_n \neq v_n$,

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

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

Let N be the maximal depth of a variable in the disequality constraints, k be the maximal number of variables in a disequality constraint, C be the number of constraints $\pm\mathsf{checkl}$ in Φ_1, L be the number of constraints loop in Φ_1, R be the number of constraints $\neg\mathsf{route}$ in Φ_1, and $M = 2 \times (kN + 3C + L + R + 2)$.

We are going to show that there exists a solution σ where for all variables x of sort lists, $|x\sigma| \leq M$.

Consider σ a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G. If for all variables x of sort lists, $x\sigma$ is of length at most M, the result is true.

Else, there exists a variable x_ℓ of sort lists such that $|x_\ell\sigma| > M$. We extend the graph G with a set of fresh nodes that are neighbors of all the other nodes. These nodes are called *ubiquitous*. Formally, let V_{ubiq} be a set of $M/2$ fresh names of $\mathcal{I}$ such that $V_{ubiq} \cap names(\mathcal{C}, \Phi_1, \Phi_2, \sigma) = \emptyset$. Let $E' = E_G \cup \{(a,b) |\ a \in E_G \cup V_{ubiq}, b \in V_{ubiq}\}$, and $G' = (V_G \cup V_{ubiq}, E')$.
We build σ' such that for all $x \neq x_\ell, x\sigma' = x\sigma$, and $|x_\ell\sigma'| \leq M$.

In what follows, we say that a list $l = a_1 :: \ldots :: a_n$ is an *ubiquitous variation* of a partially marked list $l' = a'_1 :: \ldots :: a'_n$ if, for every $1 \leq i \leq n$,

- either $a_i = a'_i$,
- or a'_i is not marked and $a_i \in V_{ubiq}$.
- Moreover we require that the ubiquitous nodes of l are all distinct.

x_ℓ is a variable of sort lists, and σ is a solution of $\Phi_1 \wedge \Phi_2$. We build $x_\ell\sigma'$ by marking the names we want to keep in the list in the following manner:

$x_\ell\sigma =$

a_1	a_2	$\ldots$	a_{kN}	$\ldots$	a_P

We mark the first kN names in the list:

$\overline{a_1}$	$\overline{a_2}$	$\ldots$	$\overline{a_{kN}}$	$\ldots$

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

- 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 $x_\ell \in var(l_j)$. Assume $l_j = d_1 :: \ldots :: d_p :: x_\ell$. 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 $(a, c) \in E_G \subseteq E'$.
  - if $l' = c :: b :: l_1$, then $(b, c) \in E_G \subseteq E'$.

  c appears exactly once in $l_j\sigma$, so either there exists n such that $c = d_n$, 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 ubiquitous variation of a list extracted from $x_\ell\sigma$ containing the marked names satisfies this checkl condition for graph G'.

a_1	$\ldots$	$\overline{a_{m-1}}$	$\overline{a_m}$	$\overline{a_{m+1}}$	$\ldots$	a_P

- 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 $l_j = b_1 :: \ldots :: b_p :: x_\ell$. 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, $b_n \neq c$ 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 $x_\ell\sigma$.

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

  - c appears once in $l_j\sigma$, but one of its neighbors in the list is not a neighbor of it in the graph. For example, $c = a_i$ and $(a_i, a_{i+1}) \notin E_G$. We mark c and this false neighbor when they appear in $x_\ell\sigma$. In that case, $(a_i, a_{i+1}) \notin E'$, because $a_i, a_{i+1} \in names(\Phi_1, \sigma)$.

a_1	$\ldots$	$\overline{a_i}$	$\overline{a_{i+1}}$	$\ldots$	a_P

  Any ubiquitous variation of a list extracted from $x_\ell\sigma$ containing the marked names satisfies the $\neg\mathsf{checkl}$ condition for graph G'.
- 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 $x_\ell\sigma$. Any ubiquitous variation of a list extracted from $x_\ell\sigma$ containing the marked names satisfies the loop condition for graph G'. Indeed, the condition does not depend on the graph.
- 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 $p_h = b_1 :: \ldots :: b_p :: x_\ell$. Removing nodes from the list preserves this condition, so any extracted list of $x_\ell\sigma$ satisfies the $\neg\mathsf{loop}$ condition. Moreover, $V_{ubiq} \cap names(\Phi_1, \sigma) = \emptyset$, so no ubiquitous node appears in $p_h\sigma$, and in a ubiquitous variation, the ubiquitous nodes are all distinct. Consequently, any ubiquitous variation of a list extracted from $x_\ell\sigma$ satisfies the $\neg\mathsf{loop}$ condition.
- 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 $r_l = b_1 :: \ldots :: b_p :: x_\ell$. 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 $(c,d) \notin E_G$. We mark c and d if they appear in $x_\ell\sigma$. As $c, d \in names(\Phi_1, \sigma)$, and $(c,d) \notin E_G$, $(c,d) \notin E'$.

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

Any ubiquitous variation of a list extracted from $x_\ell\sigma$ containing the marked names satisfies the $\neg$route condition for G'.

- If there exists r_l such that route(r_l) occurs positively in Φ_1, i.e. $\pm_l = +$, and $x_\ell \in var(r_l)$. Assume $r_l = b_1 :: \ldots :: b_p :: x_\ell$. As σ is a solution for Φ_1, for every l' sublist of $x_\ell\sigma$, if $l' = a :: b :: l_1$, then $(a,b) \in E_G$, and there are no repeated names in $r_l\sigma$. Consider l_1 extracted from $x_l\sigma$ by leaving exactly one unmarked name between sequences of marked names, and let l_2 be the ubiquitous variation of l_1 obtained by replacing all unmarked names of l_1 by ubiquitous names. $b_1\sigma :: \ldots :: b_p\sigma :: l_2$ satisfies the route condition for G': indeed, let l_3 be a sublist of $b_1\sigma :: \ldots :: b_p\sigma :: l_2$.
  - If $l_3 = b_i\sigma :: b_{i+1}\sigma :: l'$, $(b_i\sigma, b_{i+1}\sigma) \in E_G \subseteq E'$
  - If $l_3 = b_p\sigma :: c_1 :: l'$: either a_1 was marked, and $c_1 = a_1$, so $(c_1, b_p) \in E_G \subseteq E'$, or a_1 was unmarked, and $c_1 \in V_{ubiq}$, so $(c_1, b_p) \in E'$.
  - If $l_3 = c_1 :: c_2 :: l'$: if there exists i such that $c_1 = a_i$ and $c_2 = a_{i+1}$, then $(c_1, c_2) = (a_i, a_{i+1}) \in E_G \subseteq E'$. Else, c_1 (or c_2) is ubiquitous, and so $(c_1, c_2) \in E'$.

  Moreover, there are no repeated names in $b_1\sigma :: \ldots :: b_p\sigma :: l_2$.

We now count the number of marked names. We have marked the first kN names in the list. For each constraint $\pm$checkl, we mark at most 3 names in the list. Suppose there are several constraints $\neg$route(l) with x_ℓ sublist of l. Either $\neg$route$(x_\ell\sigma)$ holds, and we can mark two names in $x_\ell\sigma$ which will make all the $\neg$route constraints true; or the constraint is satisfied by marking one name for each constraint. Thus, we need only to mark max$(R, 2)$ names when $R \geq 1$ and 0 otherwise. Thus, in any case, it is sufficient to mark $R+1$ names in $x_\ell\sigma$ to satisfy the $\neg$route constraints. 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 $kN + 3C + (R+1) + (L+1) \leq M/2$. Consider l_1 extracted from $x_l\sigma$ by leaving exactly one unmarked name between sequences of marked names, and l_2 the ubiquitous variation of l_1 obtained by replacing all unmarked names of l_1 by ubiquitous names. For each condition considered above, l_2 satisfies it, as it is a ubiquitous variation of a list extracted from $x_\ell\sigma$. We have no more than $M/2$ ubiquitous names in l_2, so $|l_2| \leq M$.

Let σ_0 be the substitution such that $x\sigma_0 = x\sigma$ for every $x \in dom(\sigma) \setminus \{x_\ell\}$, 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, and thus σ is also a solution for G', thanks to Lemma 15. So by construction, σ' is a solution of Φ_1 for G'.

Now, it remains to show that σ' is a solution of $(\mathcal{C}, \mathcal{I})$ and Φ_2. Thanks to Lemma 4, we can assume w.l.o.g that for every variable x which is not of sort loc or lists, $x\sigma \in \mathcal{I}$.

Deduction constraints. We deduce that σ' is a solution of $(\mathcal{C}, \mathcal{I})$. Indeed, consider a right-hand side variable x_i of $\mathcal{C}$. Either x_i is of sort loc or lists, which means that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma'$, thus $\mathcal{I} \vdash x_i\sigma'$. Or x_i is not of sort loc or lists, so in particular $x_i \in dom(\sigma) \setminus x_\ell$, and $x_i\sigma' = x_i\sigma \in \mathcal{I}$, so $\mathcal{I} \vdash x_i\sigma'$. Hence, in both cases, we have that $\mathcal{I} \vdash x_i\sigma'$. Consequently, σ is a solution of $(\mathcal{C}, \mathcal{I})$.

Disequality constraints. Consider $\forall Y.u \neq v \in \Phi_2$. 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 $\tau = \{x_\ell \mapsto x\sigma\} \circ \mu$. We have that:

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

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.

Else, there exists a term t such that $\mu(x_\ell) = t$, and $var(t) \subseteq Y$. We apply Lemma 12 to $T = \{u\sigma_0, v\sigma_0\}$, $P = \{u\sigma_0 = v\sigma_0\}$ and μ. We get that $|x_\ell\mu|_d \leq \#dom(\mu).\mathsf{max}(|u\sigma_0|_d, |v\sigma_0|_d) \leq \#dom(\mu).\mathsf{max}(|u|_d, |v|_d) \leq kN$, because σ_0 is ground. 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 $t = a_1 :: \ldots :: a_n :: \bot$, with $a_1, \ldots, a_n$ terms of type loc. We write $t = t_1@t_2$ with t_2

ground term of maximal size, where @ denotes the concatenation of lists. We have 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 ubiquitous constant. In particular, this means that $a_\ell \notin names(\Phi_2, \sigma)$. On the other hand, $a_{k'} \in names(t_2) \subseteq names(\Phi_2, \sigma_0)$. Consequently, $a_{k'} \neq a_\ell$, and so for any substitution θ, we have that $x_\ell\sigma' \neq t\theta$.

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 $|t|_d \leq kN$, hence we have that $n < kN$. 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

$$\rho' \circ \mu \quad = \quad \theta' \circ \{x_\ell \mapsto x_\ell\sigma'\} \qquad (1)$$

  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 $M \geq kN \geq n$. 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 $x \in dom(\rho) \setminus \{y_\ell\}$, 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 $v\sigma$.

  First, we have that:

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

  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.

Hence, we have shown that σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G', where $|x_\ell\sigma'| \leq M$. We can repeat this procedure recursively on all lists longer than M, and we get in the end a substitution σ'' and a graph G'' such that for every variable x of sort lists, $|x\sigma''| \leq M$, and σ'' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G''. □

Lemma 6 *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 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 $\mathcal{C}, \Phi_1, \Phi_2$ and G.*

Proof. We write $G = (V_G, E_G)$, $\Phi_2 = \bigwedge_n \forall Y_n.u_n \neq v_n$,

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

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

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

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, $|x\sigma| \leq M$, and
- $|x\sigma| = 1$ otherwise.

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:

$$|\sigma| = \sum_{x \in dom(\sigma)} |x\sigma|.$$

First, we have that $|x\sigma| = 1$ when x is a variable of sort loc. Moreover, thanks to Lemma 4, we can assume that $x\sigma \in \mathcal{I}$ (and thus $|x\sigma| = 1$) when x is a variable that is neither of sort loc nor of type lists. Now, either $|x\sigma| \leq M$ for all variables x of sort lists and 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 $|x_\ell\sigma'| \leq M < |x_\ell\sigma|$.

In what follows, we say that a list $l = a_1 :: \ldots :: a_n$ is a *variation* of a partially marked list $l' = a'_1 :: \ldots :: a'_n$ if there exists $1 \leq j \leq n$ such that:

- a'_j is not marked and a_j is a fresh name,
- $\forall i \neq j, a_i = a'_i$

x_ℓ is a variable of sort lists, and σ is a solution of $\Phi_1 \wedge \Phi_2$. We build $x_\ell\sigma'$ by marking the names we want to keep in the list in the following manner:

$x_\ell\sigma =$

a_1	a_2	$\ldots$	a_{kN}	$\ldots$	a_P

We mark the first kN names in the list:

$\overline{a_1}$	$\overline{a_2}$	$\ldots$	$\overline{a_{kN}}$	$\ldots$

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

- 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 $x_\ell \in var(l_j)$. Assume $l_j = d_1 :: \ldots :: d_p :: x_\ell$. 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 $(a, c) \in E_G$.
  - if $l' = c :: b :: l_1$, then $(b, c) \in E_G$.

  c appears exactly once in $l_j\sigma$, so either there exists n such that $c = d_n$, 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 the marked names satisfies the checkl condition for graph G.

a_1	$\ldots$	$\overline{a_{m-1}}$	$\overline{a_m}$	$\overline{a_{m+1}}$	$\ldots$	a_P

- 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 $l_j = b_1 :: \ldots :: b_p :: x_\ell$. 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, $b_n \neq c$ 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 $x_\ell\sigma$.

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

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

a_1	$\ldots$	$\overline{a_i}$	$\overline{a_{i+1}}$	$\ldots$	a_M

  Any variation of a list extracted from $x_\ell\sigma$ containing the marked names satisfies the $\neg\mathsf{checkl}$ condition for graph G.

- 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 $x_\ell\sigma$.

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

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

- 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 $p_h = b_1 :: \ldots :: b_p :: x_\ell$. 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.

- 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 $r_l = b_1 :: \ldots :: b_p :: x_\ell$. 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 $(c, d) \notin E_G$. 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 the marked names satisfies the $\neg\mathsf{route}$ condition for G.

- 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 $r_l = b_1 :: \ldots :: b_p :: x_\ell$. Write $r_l\sigma = a_1 :: \ldots :: a_n$. As σ is a solution for Φ_1 in G, for every $0 < i < n$, $(a_i, a_{i+1}) \in E_G$ and for every $i \neq j$, $a_i \neq a_j$. Consequently, $|r_l\sigma| \leq |V_G|$, and as $|x_\ell\sigma| \leq |r_l\sigma|$, $|x_\ell\sigma| \leq |G|$. But our hypothesis tells us that $|x_\ell\sigma| > M \geq |G|$. 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 $|x_\ell\sigma| \geq M$. 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 $x \in dom(\sigma) \setminus \{x_\ell\}$, 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 right-hand side variable x_i of $\mathcal{C}$. Either x_i is of sort loc or lists, which means that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma'$, thus $\mathcal{I} \vdash x_i\sigma'$. Or x_i is not of sort loc or lists, so in particular $x_i \in dom(\sigma) \setminus x_\ell$, and $x_i\sigma' = x_i\sigma \in \mathcal{I}$, so $\mathcal{I} \vdash x_i\sigma'$. Hence, in both cases, we have that $\mathcal{I} \vdash x_i\sigma'$. Consequently, σ' is a solution of $(\mathcal{C}, \mathcal{I})$.

Disequality constraints. Consider $\forall Y.u \neq v \in \Phi_2$. 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 $\tau = \{x_\ell \mapsto x\sigma\} \circ \mu$. We have that:

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

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.

Else, there exists a term t such that $\mu(x_\ell) = t$, and $var(t) \subseteq Y$. We apply Lemma 12 to $T = \{u\sigma_0, v\sigma_0\}$, $P = \{u\sigma_0 = v\sigma_0\}$ with $\mu = \mathsf{mgu}(P)$. We get that $|t|_d \leq \#dom(\mu).\mathsf{max}(|u\sigma_0|_d, |v\sigma_0|_d) \leq \#dom(\mu).\mathsf{max}(|u|_d, |v|_d) \leq kN$, because σ_0 is ground. 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 $t = a_1 :: \ldots :: a_n :: \bot$, with a_n terms of type loc. We write $t = t_1@t_2$ with t_2 ground term of maximal size, where @ denotes the concatenation of lists. We have 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 constant 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

$$\rho' \circ \mu \quad = \quad \theta' \circ \{x_\ell \mapsto x_\ell\sigma'\} \qquad (2)$$

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 $M \geq kN \geq n$. 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 $x \in dom(\rho) \setminus \{y_\ell\}$, 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 $v\sigma$.

First, as $x_\ell\sigma$ and $x_\ell\sigma'$ have the same first kN elements by construction, and $n < kN$, we have that:

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

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. □

C.2 Proof of decidability

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

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

Proof. 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 $\mathcal{P}$, $\mathcal{S}$, Φ or $\mathcal{M}$. Our decision procedure works as follows:

Step 1. We partially guess the graph $G = (V, E)$. Actually, we guess whether $(n_1, n_2) \in E$ for every $n_1, n_2 \in V_K$. Let $G_K = (V_K, E_K)$ 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 σ be an mgu of the equality constraints in $\mathcal{C}' = \mathcal{C} \wedge \{u = \mathsf{error}\}$.

Step 4. The system $\mathcal{C}'\sigma$ is equal to $(\mathcal{D}, \mathcal{I}_0) \wedge \Phi_1$ where $(\mathcal{D}, \mathcal{I}_0)$ is a special constraint system and Φ_1 is a formula that contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. We get rid of the trivial constraints of the form $t = t$. We have that $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}_i$ where $\mathcal{I}_i$ is a finite set of terms and we can choose $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \setminus names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{I}_i)$.

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

$$\Phi_1' = \bigwedge_k \phi_1^k \vee \cdots \vee \phi_{i_k}^k.$$

We choose non-deterministically $\phi^k_{\alpha_k}$ for every k. Let $\Phi_2 = \bigwedge_k \phi^k_{\alpha_k}$, which we can write $\Psi_2 \wedge \Psi_1$, where Ψ_2 only contains conjunctions of disequality constraints and Ψ_1 is a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$.

Step 7. Let $\mathsf{V} = \#var(\mathcal{P})$ be the number of variables occurring in $\mathcal{P}$ and $\mathsf{S} = |\mathcal{P}|$ be the size of $\mathcal{P}$. Let $\mathcal{I}_0'$ be a finite subset of $\mathcal{I}_0$ of size

$$2\mathsf{VS} \times (\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 2).$$

Guess the values of variables which are not of sort lists in $\mathcal{I}_0' \cup V_K$. Guess the values of variables of sort lists among lists of nodes in $\mathcal{I}_0' \cup V_K$ of length at most

$$2 \times (\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 2).$$

This gives us a substitution σ and we guess a graph $G = (V, E)$ such that $V = \mathcal{I}_0' \cup V_k$ 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.

We now explain each step of our algorithm.

Step 1. We have that $|V_K| < |\mathcal{P}| + |\Phi| + |\mathcal{S}| + |\mathcal{M}|$. Hence, we can guess G_K whose size is polynomially bounded.

Step 2. For every graph $G' = (V', E')$ such that $V_K \subseteq V'$ and $E_K = \{(n_1, n_2) \in E' \mid n_1, n_2 \in V_K\}$, we have

that $(\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^{s}_{G_K,\mathcal{M}} (\mathcal{P}';\mathcal{S}';\mathcal{I}';\mathcal{C}')$ if, and only if, $(\mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^{s}_{G',\mathcal{M}} (\mathcal{P}';\mathcal{S}';\mathcal{I}';\mathcal{C}')$.

So we can guess the transitions knowing only E_K. Now, thanks to Proposition 1 we deduce that there is an $\mathcal{M}$-attack on K and Φ for graph G if, and only if, there is a derivation

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

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's 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 $\sigma = \bot$, then $\mathcal{C}'$ has no solution. Else, $\mathcal{C}'$ has a solution if, and only if, $\mathcal{C}'\sigma$ has a solution.

Step 4. We have that $\mathcal{C}'\sigma = (\mathcal{D},\mathcal{I}_0) \wedge \Phi_1$ where $(\mathcal{D},\mathcal{I}_0)$ is a special constraint system, and Φ_1 contains the remaining constraints, i.e. disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. In particular, $(\mathcal{D},\mathcal{I}_0)$ satisfies the origination property since application of a substitution preserves this property.

Step 5. We apply Theorem 3. 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 $\theta = \sigma' \circ \theta'$, $\mathcal{D} \rightsquigarrow^{*}_{\sigma'} \mathcal{D}'$ and θ' is a solution for $\mathcal{D}'$ and $\Phi_1\sigma'$ for graph G.

Step 6. This step is straightforward. $\Phi_1\sigma'$ contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. Consequently, Φ_1' also contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$, and $\Phi_2 = \bigwedge_k \phi^k_{\alpha_k}$, obtained from Φ', can be written:

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

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 $\Phi_2 = \Psi_1 \wedge \Psi_2$, where:

- $\Psi_2 = \bigwedge_i \forall Y_i.u_i \neq v_i$
- $\Psi_1 = \bigwedge_j \pm_j\, \mathsf{check}(a_j,b_j) \ \wedge\ \bigwedge_l \pm_l\, \mathsf{route}(r_l) \ \wedge\ \bigwedge_k \bigwedge_i \pm_{i_k}\, \mathsf{checkI}(c_{i_k},l_k) \ \wedge\ \bigwedge_h \pm_h\, \mathsf{loop}(p_h)$ with $\pm \in \{+,-\}$.

Step 7. Let $\mathsf{V} = \#var(\mathcal{P})$ be the number of variables occurring in $\mathcal{P}$ and $\mathsf{S} = |\mathcal{P}|$ be the size of $\mathcal{P}$.

First, we show that $|t|_d \leq \mathsf{VS}^2 + \mathsf{S}$ for any term $t \in \mathcal{D}' \wedge \Psi_1 \wedge \Psi_2$. By construction, we know that there exists t' in $\mathcal{C}'$ such that $(t'\sigma)\sigma' = t$.

Actually, we have that $\sigma' \circ \sigma = \mathsf{mgu}(P)$ for some set P of equations between subterms in $\mathcal{C}'$. In order to prove this result, we show that:

- $\sigma_i = \mathsf{mgu}(P^i_{\mathcal{D}})$ for some set of equations $P^i_{\mathcal{D}}$ between subterms of $\mathcal{D}$, and
- $St(\mathcal{D}_i) \subseteq St(\mathcal{D})\sigma_i$

when $\mathcal{D} \rightsquigarrow_{\sigma_i} \mathcal{D}_i$. We prove this result by induction on the length n of the derivation and we rely on Lemma 14 and Lemma 13.

Hence, we have that $\sigma = \mathsf{mgu}(P_{\mathcal{C}'})$ where $P_{\mathcal{C}'}$ is a set of equations between subterms in $\mathcal{C}'$ and we have also that $St(\mathcal{D}) \subseteq St(\mathcal{C}'\sigma) \subseteq St(\mathcal{C}')\sigma$ (Lemma 14). We have shown that $\sigma' = \mathsf{mgu}(P^n_{\mathcal{D}})$ and we have that $P^n_{\mathcal{D}} = P'_{\mathcal{C}'}\sigma$ for some set $P'_{\mathcal{C}'}$ (equations between subterms in $\mathcal{C}'$). Thanks to Lemma 13, we deduce that $\sigma' \circ \sigma = \mathsf{mgu}(P)$ where $P = P_{\mathcal{C}'} \cup P'_{\mathcal{C}'}$ is a set of equations between subterms in $\mathcal{C}'$. Thanks to Lemma 12, we deduce that

$$|(x\sigma)\sigma'|_d \leq \#\{dom(\sigma' \circ \sigma)\} \times \mathsf{max}\{|t|_d \mid t \in \mathcal{C}'\}.$$

By inspection of the symbolic transition rules, we see that at each step, the constraint system can gain at most V variables (this case corresponds to the communication rule), so the total number of variables in the resulting constraint system is bounded: $\#var(\mathcal{C}') \leq \mathsf{VS}$. Moreover, $\mathsf{max}\{|t|_d \mid t \in \mathcal{C}'\} \leq \mathsf{S}$. Hence, we have that $|t|_d \leq \mathsf{VS}^2 + \mathsf{S}$ for any term $t \in \mathcal{D}' \wedge \Psi_1 \wedge \Psi_2$. In particular, we have that $N \leq \mathsf{VS}^2 + \mathsf{S}$ where N is the maximal depth of variables in the terms of all disequality constraints in Ψ_2.

Let L be the number of occurrences of a loop predicate in Ψ_1, R be the number of occurrences of a route predicate in Ψ_1, and C be the number of occurrences of a checkI predicate in Ψ_1. We have that:

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

Let n be the number of variables in $\mathcal{D}' \wedge \Psi_1 \wedge \Psi_2$. We have that $n \leq \#var(\mathcal{C}') \leq \mathsf{VS}$. Let k be the maximal total number of variables in a disequality constraint. We have also that $k \leq \mathsf{VS}$.

Now, we have to show that if there exists a graph $G = (V, E)$ such that $V_K \subseteq V$ and $E_K = \{(n_1, n_2) \in E \mid n_1, n_2 \in V_K\}$ 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 4, we know that there is a solution where the variables which are not of sort loc or lists are substituted by names of sort loc (independently of the underlying graph).
- Thanks to Lemmas 5 and 15, we know that if there is a graph leading to a solution, there exists a substitution σ where the size of the instantiated variables of type *lists* is bounded by $M = 2 \times (kN + 3C + R + L + 2)$ and there exists a graph $G = (V, E)$ where V is the set of names that occur in $\mathcal{D}'$, Ψ_1, Ψ_2, σ and V_K, such that σ is a solution for G and $E_K = \{(n_1, n_2 \in E \mid n_1, n_2 \in V_K\}$.

We have that: $M \leq 2 \times (\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 2)$. Hence, the number of distinct names of sort loc in σ is bounded by $n \times M \leq 2\mathsf{VS} \times (\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 2)$. So there is a solution σ for a graph G with all nodes in $\mathcal{I}'_0 \cup V_K$. □

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 an hole, G be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Proof. 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 $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I})$. We write $G = (V, E)$. 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 σ be an mgu of the equality constraints of $\mathcal{C}' = \mathcal{C} \wedge \{u = \mathsf{error}\}$.

Step 3. The system $\mathcal{C}'\sigma$ is equal to $(\mathcal{D}, \mathcal{I}_0) \wedge \Phi_1$ where $(\mathcal{D}, \mathcal{I}_0)$ is a special constraint system and Φ_1 is a formula that contains disequality constraints and formulas of $\mathcal{L}_{\mathsf{route}}$. We get rid of the trivial constraints of the form $t = t$. We have that $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}_i$ where $\mathcal{I}_i$ is a finite set of terms and we can choose $\mathcal{I}_0 = \mathcal{N}_{\mathsf{loc}} \setminus names(\mathcal{P}, \mathcal{S}, \Phi, \mathcal{I}_i)$.

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

$$\Phi'_1 = \bigwedge_k \phi^k_1 \vee \cdots \vee \phi^k_{i_k}.$$

We choose non-deterministically $\phi^k_{\alpha_k}$ for every k. Let $\Phi_2 = \bigwedge_k \phi^k_{\alpha_k}$, which we can write $\Psi_2 \wedge \Psi_1$, where Ψ_2 only contains conjunctions of disequality constraints and Ψ_1 is a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$.

Step 6. Let $\mathsf{V} = \#var(\mathcal{P})$ the number of variables occurring in $\mathcal{P}$ and $\mathsf{S} = |\mathcal{P}|$ be the size of $\mathcal{P}$. Let $\mathcal{I}'_0$ be a finite subset of $\mathcal{I}_0$ of size

$$\mathsf{VS} \times \mathsf{max}(\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 3, |G|).$$

Guess the values of variables which are not of sort lists in $\mathcal{I}'_0 \cup V$. Guess the values of variables of sort lists among lists of nodes in $\mathcal{I}'_0 \cup V$ of length at most

$$\mathsf{max}(\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 3, |G|).$$

This gives us a substitution σ. We check whether σ is a solution of $(\mathcal{D}', \mathcal{I}_0) \wedge \Phi_2$ for graph G.

The first five steps are the same as Steps 2 to 6 in Theorem 1.

Step 6. First, we have that $|t|_d \leq \mathsf{VS}^2 + \mathsf{S}$ for any term $t \in \mathcal{D}' \wedge \Psi_1 \wedge \Psi_2$, as shown in the proof of Step 7 in Theorem 1.

Let N be the maximal depth of variables in the terms of all disequality constraints in Ψ_2. We have that $N \leq \mathsf{VS}^2 + \mathsf{S}$.

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

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

Let n be the number of variables $\mathcal{D}' \wedge \Psi_1 \wedge \Psi_2$. We have that $n \leq \#var(\mathcal{C}') \leq \mathsf{VS}$. Let k be the maximal total number of variables in a disequality constraint. We also have that $k \leq \mathsf{VS}$.

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 4, we know that there is a solution where the variables which are not of sort loc or lists are substituted by names of sort loc.
- Thanks to Lemma 6, 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

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

Actually, we have that

$$M \leq \mathsf{max}(\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 3, |G|).$$

Hence, the number of distinct names of sort loc in σ is bounded by

$$n \times M \leq \mathsf{VS} \times \mathsf{max}(\mathsf{V}^2\mathsf{S}^3 + \mathsf{VS}^2 + 5\mathsf{S}^2 + 3, |G|).$$

So there is a solution σ for the graph G with all vertices in $\mathcal{I}'_0 \cup V$. □