INSTITUT FÜR INFORMATIK

Selecting Theories and Nonce Generation for Recursive Protocols

Klaas Ole Kürtz

Bericht Nr. 0709
August 2007

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Arbeitsgruppe Theoretische Informatik Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Selecting Theories and Nonce Generation for Recursive Protocols

Klaas Ole Kürtz

Technical Report Nr. 0709
August 2007

kuertz@ti.informatik.uni-kiel.de

This work was supported by the DFG under grant KU 1434/4-1.

Abstract

In recent years, formal methods have been developed to analyze and verify cryptographic protocols. We will focus on protocols that rely on iteration or recursion. These protocols typically use special security tokens – such as numbers used only once, called nonces, or keys generated by a principal – to achieve their security assertions.

The recursion depth of the computations in such protocols and thus the number of fresh tokens occurring in a run of a protocol is not explicitly bounded by the protocol's description. Therefore, we need a mechanism to provide the protocol's principals with the ability to generate an unbounded number of fresh tokens.

In this report we will extend the model of selecting theories introduced by Truderung – in this model recursive protocols can be analyzed in the presence of a Dolev-Yao intruder. We will present an extended model that allows the principals to generate fresh tokens, and we will show decidability with respect to a bounded number of sessions. In the proof, attacks on such protocols will be represented by a special graph structure introduced by Truderung called ADAG; we will prove our decidability result by bounding the size of ADAGs. In the protocol model and in the ADAGs the modeling of fresh tokens will be based on an infinite set of constants in the signature.

Contents

1 Introduction and Fundamentals

The use of cryptographic protocols is widely spread – for example, e-commerce and online banking are typically secured by protocols using cryptographic methods as encryption or digital signatures, promising security properties as secrecy or authentication. The design of such protocols is complex and error-prone, hence, an analysis of their security properties is essential. But as the analysis itself is likewise complex and laborious, the overall analysis is usually split up into several more manageable tasks.

One of these tasks is a high-level analysis of the protocol's logic, abstracting from most of the implementational and cryptographical details of the cryptographic methods used in a protocol. The main question on this level of abstraction is if an adversary can attack a protocol's security assertions if he has control over the network and can manipulate messages, but has no means like cryptanalysis – this type of adversary is called Dolev Yao intruder. A symbolic approach on this level has allowed models and tools to be develop, which have become useful and practical for the (automatic) analysis of cryptographic protocols [DY83, FHG98, RT01].

While most models can only handle protocols of a simple structure, we will examine a broader class of protocols including ones that can only be modeled by iteration or recursion. These protocols often rely on special security tokens such as numbers used only once, called nonces, or fresh keys generated by principals of the protocol [Mea01, SB05]. As the size of the messages exchanged in such a protocol and thus the recursion depth of the principals' computations is not explicitly bounded by the protocol's description, we need a mechanism to provide the principals with the possibility to generate an unbounded number of nonces and fresh keys.

In this report we will extend a model introduced by Truderung [Tru05b], which uses special Horn theories – called selecting theories – to model the principal's computations. In this model secrecy can be decided for recursive protocols in the presence of a Dolev-Yao intruder; we will extend this model to handle the generation of fresh tokens and show decidability of secrecy of a protocol (with respect to a bounded number of sessions). In the

decidability proof, attacks on such protocols will be represented by a special graph structure called ADAG introduced in [Tru05a], and we will be able to present a decision algorithm by bounding the size of ADAGs. In the protocol model and in our ADAGs the modeling of fresh tokens will be based on an infinite set of constants along the lines of [KW04].

1.1 Cryptographic Protocols

1.1.1 Cryptographic Primitives

In this report we will use cryptographic algorithms such as encryption schemes, digital signatures or hashing algorithms. As we will abstract from the cryptographical and implementational details of these algorithms, we refer the reader to introductory texts like [Sti95] or [DK02], the latter also has a chapter about cryptographic protocols. We will call such algorithms *(cryptographic) primitives*.

One important primitive we refer to are *nonces*, which is an abbreviation for *number used once*. We assume that during computations each computing party can generate numbers that were not generated by this party before. This allows us, for instance, to *tag* messages, i. e., to distinguish between two messages which have the same content, by appending nonces to the messages.

The Handbook of Applied Cryptography [MVO96] defines nonces as

> [...] a value used no more than once for the same purpose. It typically serves to prevent (undetectable) replay.
>
> The term *nonce* is most often used to refer to a »random« number in a challenge-response protocol [...]. Three main classes of time-variant parameters are [...] random numbers, sequence numbers, and time-stamps. Often, to ensure protocol security, the integrity of such parameters must be guaranteed.

In this report, the generation of nonces and the generation of fresh keys will both be referred to as *generation of fresh tokens*, and we will abstract from the details of the implementation as described in Section 1.2.5.

1.1.2 Cryptographic Protocols

In computer science, a *communication protocol* is a convention for the communication between two or more computing parties called *principals*. The Handbook of Applied Cryptography [MVO96] defines it as

> [...] a multi-party algorithm, defined by a sequence of steps precisely specifying the actions required of two or more parties in order to achieve a specified objective.

One has to distinguish between the *service* offered by a protocol (»a specified objective«), i. e., the purpose and benefit when using this protocol, and the description of the *internal structure* (»sequence of steps«), i. e., the way in which this protocol tries to accomplish its purpose. The definition of a protocol may include several levels, from the syntax of electronic signals to the semantics of applications.

Cryptographic protocols apply cryptographic methods such as encryption algorithms and digital signatures to offer a specific security service. Their internal structure can often be described on an abstract level, i. e., as a sequence of abstract messages under certain conditions. Their service may, for example, include secrecy, authentication, key exchange or agreement, secured data transport, anonymity, non-repudiation, or fairness. In this report we will use the term »security of a protocol« – if not stated otherwise – for secrecy, i. e., that an adversary is not able to derive information that is meant to be kept secret by a protocol.

Today the use of cryptographic protocols on modern communication networks is widely spread – for example, Transport Layer Security (TLS, formerly SSL, first presented in [HE95]) and Secure Shell (SSH, introduced in [Ylö96]) are common cryptographic protocol used to secure connections over the internet. Although partly out-of-date, the Clark-Jacob library [CJ97] gives a good survey of cryptographic protocols.

Bellare and Goldwasser state [GB01]:

> Classical cryptography is concerned with the problem of [secure] communication between users by providing privacy and authenticity. The need for an underlying infrastructure for key management leads naturally into the topic of key distribution. For many years this is all there was to cryptography.
>
> One of the major contributions of modern cryptography has been the development of advanced protocols. These protocols enable users to electronically solve

many real world problems, play games, and accomplish all kinds of intriguing and very general distributed tasks. Amongst these are zero-knowledge proofs, secure distributed computing, and voting protocols.

1.1.3 Recursive Cryptographic Protocols

Most of the cryptographic protocols used in real world applications can be abstracted to the mathematical level and then be described by simple means: The protocols consist of a certain sequence of messages exchanged by the principals, where each principal that receives a message replies with a single message that can be computed from the previous messages without the need of complex methods as iteration or recursion.

This is often based on the fact that the cryptographic protocol is separated from the application using it. For example, the standard situation for TLS is »only« to secure communication between two principals by agreeing upon a set of cryptographic algorithms used and then exchanging key material for these primitives. Any application may use this secured communication channel and transmit arbitrary data over it, abstracting from the details of the implementation of TLS.

But some protocols, called *recursive protocols*, contain complex actions or data structures: On the one hand, there are, for instance, group protocols providing services not only for two principals, but for a potentially unbounded number of principals. Such protocols often contain data structures such as lists of the principals involved or rely on a server which sends messages to each principal.

On the other hand, there are protocols that integrate application logic and cryptographic methods: For example, there are protocols based on web services and WS-Security that allow the modeling of business processes. During such a business process, a whole set of data is exchanged between several parties, which have to encrypt or digitally sign parts of the messages or data, or which may only be allowed to read or to add data etc.

One important example of a recursive protocol is the Internet Key Exchange (IKE) protocol [Kau05, Zho99], for more examples see [SB05]. We will use a simple recursive protocol for illustration and as an example protocol, the Recursive Authentication Protocol [BO97, Pau97].

1.2 Analysis of Cryptographic Protocols

1.2.1 Different Levels and Approaches

When cryptographic protocols are used in the real world, their security depends on factors on different levels. One can distinguish between different types of weaknesses:

1. *User level:* When one uses software containing cryptographic primitives or cryptographic protocols, there are several possible problems like weak or inscribed passwords, unconditional trust exploited by social engineering or phising, and running security-critical applications on insecure systems.
2. *Implementation level:* Implementing cryptographic algorithms and protocols is extremely error-prone – there may be numerous ways to attack implementations on hardware (e. g., power consumption) or software level (stack-overflow, deadlocks, timing attacks, weak pseudo-random number generators etc.).
3. *Cryptographic Level:* Some cryptographic protocols are designed by simply applying an arbitrary encryption, signature, or hashing scheme. While some of these schemes are simply outdated or weak (for example, DES), the security of other schemes depends on their usage – e. g., the stream cipher RC4 should be used with a pseudo-random key and the first output bytes should be discarded. Naturally, weak or misused cryptographic primitives lead to serious flaws in protocols, the best-known example is the Wired Equivalent Privacy (WEP) protocol which uses RC4 in an insecure way; for further examples see [Boy90].
4. *Logical level:* Even if one abstracts from all other levels mentioned so far by assuming that the cryptographic primitives used are perfectly secure, there are numerous possible weaknesses in the logic of a protocol itself. As described in the next section, one major example is Lowe's attack on the Needham-Schroeder protocol [NS78, Low95], for more examples see, e. g., [CJ97].

Although the security of an actual protocol execution depends on all these levels, the complexity of the analysis necessitates to split the analysis.

The user level (1) typically eludes any analysis. Applications using cryptographic methods are, in practice, analyzed on the implementational level (2), but rarely systematically: These examinations are very laborious, they have to be done manually and do always remain partial. On the cryptographical level (3), many of the cryptographic primitives used in

protocols have been thoroughly analyzed or are still analyzed; but naturally, the work on this level will never be finished, either. Finally, as there are various potential problems on the logical level (4) of protocol design, it is also necessary to analyze the logic level of protocols.

There are two main approaches on the logical level: The first one links the security of a protocol on the logical level to the security of the cryptographic primitives, i. e., linking level three and four, while the latter one completely abstracts from the details of cryptographic primitives, i. e., solely analyzing the logical level.

- In the *computational model*, messages are viewed as bit strings, and the cryptographic operations applied on messages are considered as functions on bit strings. An adversary is typically modeled by a probabilistic polynomial-time Turing machine; and security is proven by reducing the security of a protocol to security of the cryptographic primitives. This usually yields security results which express the probability or computational complexity of attacks like in [BR93]. In this model, the complex proofs can often only be done manually, which is error-prone [CBH05].
- On the other hand, the *symbolic approach* abstracts from many of these details. Messages are modeled by a term algebra, and so the cryptographic primitives are only modeled by a set of axioms, assuming perfect cryptography. The intruder is able to modify terms in an idealized way. Thus, security properties can be described in formal logics. This enables a (semi-)automatic analysis by model checking, theorem proving or similar methods – if the problem is decidable at all, what is not necessarily the case. We will give references in the next two sections.
- In addition to these two separate approaches, first results were achieved to link both the symbolic and the computational approach, see, e. g., [AR00, BPW03, CW05].

In this report we will analyze protocols on the logical level, pursuing the symbolic approach, i. e., we will abstract from cryptographical details as described in the next sections.

1.2.2 History on the use of Formal Methods

From the beginning of computer networks, protocols were developed, implemented, and tested by hand, i. e., by trial-and-error: A protocol was published (or some software using a protocol), and if a bug or a flaw was found in a protocol or software, a revised version was

developed. But similar to the verification of algorithms, formal methods were developed and began to be practical and useful to analyze the security of cryptographic protocols.

In 1978, Needham and Schroeder published an article entitled »Using Encryption for Authentication in Large Networks of Computers« [NS78]. They described the scenario of a network without a central authority that can ensure authentication etc., and they proposed a threat model:

> We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material. While this may seem an extreme view, it is the only safe one when designing authentication protocols.

This description is the basis for today's formal analysis on the logical level of protocols. In 1983, Dolev and Yao presented the first method which allowed protocols to be modeled on this level, making it possible to formally analyze their secrecy properties with respect to the Needham and Schroeder threat model [DY83].

Ironically, an authentication protocol proposed by Needham and Schroeder [NS78] is today's best-known example for a protocol that was analyzed with formal methods – and failed: Although Denning and Sacco manually analyzed the protocols proposed by Needham and Schroeder in 1981 and found flaws in it [DS81], an important attack on the asymmetric Needham-Schroeder protocol was found by Lowe no earlier than 1995 [Low95].

Lowe then applied formal methods to analyze the flaw and to show that his extension of the protocol is secure [Low96]. Today, a modified version of the protocols proposed by Needham and Schroeder is widely used in the authentication protocol Kerberos [SNS88].

For a survey of the history of the symbolic analysis of cryptographic protocols, for numerous examples and for an outlook on open problems see [Mea95, Mea01, Mea03] and [CS02].

1.2.3 The Dolev-Yao Model

The symbolic approach of protocol analysis is based on the so-called *Dolev-Yao model*, which is a generic term for a set of assumptions and abstractions that has developed out of [DY83]. These assumptions mostly include the following ones, which will be used in our model, too:

1. A protocol run consists of *messages* exchanged by a set of *principals*. Each principal is defined by a set of consecutive actions including receiving and sending messages; such an action is called a *protocol step*.
2. The set of messages is created by a term algebra over a signature that contains constants for atomic keys, the names of principals, nonces etc., as well as function symbols for pairing and cryptographic primitives such as encryption or digital signatures.
3. The principals are connected via a network that is insecure, i. e., there is an *intruder* which has complete control over the network: He can read, manipulate, or delete each message sent by the principals, and he can create messages by combining (parts of) messages he read. This is often illustrated and modeled as »The intruder *is* the network«.
4. There can be several *sessions* of each principal, i. e., there may be multiple parallel runs of the protocol. There typically is a basic knowledge shared between the sessions of one principal, e. g., they all know the principal's name, but further than that, the different sessions are typically not connected nor coordinated except by the network, i. e., by the intruder.
5. The cryptographic primitives work perfectly – for example, the only way to learn the plaintext from an encrypted message is to use the appropriate key. No form of cryptanalysis is possible.

Although these assumptions are not always realistic, they are necessary to analyze protocols on this level of abstraction. Many decision problems and formal methods were developed out of the Dolev-Yao model or inspired by it.

Previous Results and Restrictions

The general decision problem of secrecy of a protocol and many variants thereof are undecidable as shown, e. g., in [DLMS99, EG83, AC02]. But there are important variants of the problem that are decidable – also see Figure 1.1 for selected results: For a bounded total number of sessions, the secrecy problem was shown to be NP-complete [RT01, RT03], for a bounded message size and a bounded number of nonces it is DEXP-complete [DLMS04].

The decidability results for a bounded number of sessions have been extended to protocols that incorporate, e. g., properties of operators like XOR [CKRT03a, CLS03] or Diffie Hellman

exponentiation [CKRT03b]. In addition, while most work on the formal analysis of cryptographic protocols has concentrated on protocols that offer secrecy, authentication or key exchange, there are other security properties that can be analyzed and decided [KKW05].

For decidable classes of protocols, several techniques were developed and have been applied for the (semi-)automatic analysis of secrecy of such protocols, including model checkers [MMS97], theorem provers [BS97], logic programming [CDL+99], and tree automata [Küs03].

In [Mea03], Meadows identified several emerging areas of research: open-ended protocols, new applications and threats (like anonymous communication or electronic commerce), high fidelity (i. e., lowering the level of abstraction), composability of protocols, and »getting it into the real world«. Our focus on recursive protocols falls in the first category.

1.2.4 Analyzing Recursive Cryptographic Protocols

So far, most research has concentrated on non-recursive protocols, i. e., protocols where the protocol steps can be expressed by simple rewrite rules. However, as there exists many recursive protocols, the relevance of their analysis is apparent – this has been pointed out by, e. g., [Mea01, FS00, Zho99]. Recursive protocols have been analyzed manually [PQ01] and (semi-)automatically [BS97, Mea00, SB05] – e. g., our running example, the Recursive Authentication Protocol, has been analyzed using the theorem provers CSP/PVS [BS97] and Isabelle/HOL [Pau97]. Furthermore, a general decision algorithm for an automatic analysis of recursive protocols was presented in [KW04, Tru05b].

Modeling of Fresh Tokens

In the case of a finite number of sessions and non-recursive protocols, the modeling of fresh tokens is quite easy – all nonces and keys used in a protocol run can simply be atomic messages, i. e., constants of the signature used to model terms [RT01]. This is due to the fact that in each session only a bounded number of nonces can be generated, so the total number of nonces is bounded in terms of the size of the protocol.

In models that do not have these restrictions, one has to introduce other means to model fresh tokens correctly. Some models allow an unbounded number of sessions – for example, the approach of *multiset rewriting* [CDL+99, DLMS04] uses existential quantification to model nonces, and the *strand space model* [FHG98] applies special restrictions on strands and bundles to guarantee freshness.

unbounded message size	unbounded number of session	recursive protocols	fresh tokens	decidability and references
–	yes	–	yes	undecidable [DLMS04]
–	yes	–	–	decidable [DLMS04]
yes	–	–	(yes)	decidable [RT01]
yes	–	yes	yes	decidable [KW04]
yes	–	yes	–	decidable [Tru05b]
yes	–	yes	yes	decidable (this report)

Figure 1.1: Decidability of secrecy with respect to different restrictions

In [KW04] the number of sessions is bounded, but protocols can be recursive – therefore, an infinite signature is used to allow an unbounded set of fresh tokens, and the tree automata used to develop the decision algorithm are designed to cope with infinite signatures. We will adopt this method and use an infinite signature of *anonymous constants* (the name also originates from [KW04]), and we will modify the structures used in the proofs in [Tru05b] to cope with infinite signatures or unbounded subsets thereof.

Comparison of Different Approaches

Figure 1.1 compares different approaches and some results. When aiming at recursive protocols, one naturally has to consider unbounded message sizes. Hence, to maintain decidability it is necessary to limit the total number of sessions and thus the number of principals. In [Küs03, KW04] transducers were used to model recursion. As stated in [KT07],

> [...] the expressivity of these transducer-based models is orthogonal to the Horn theory model: While the transducer-based models allow [principals] to output messages of complex structure, in the Horn theory model only lists (or sets) of messages of a more simple structure can be produced. The main disadvantage of the transducerbased model is that, unlike the Horn theory model, messages cannot be tested for equality without losing decidability. This, as already observed in [KW04], immediately implies that security is undecidable in the transducer-based model with XOR (or Diffie Hellman exponentiation) since these operators allow for (implicit) equality tests between arbitrary messages.

On the other hand, properties of the XOR operator can be added to the model based on Horn theories, allowing decidability not for all protocols, but for a major class of protocols [KT07].

In this report we will examine recursive protocols with an unbounded message size and an unbounded number of fresh tokens, but we will limit the number of sessions – in fact, we will limit the number of principals and allow only one session per principal, but a bounded number of sessions per principal can simply be simulated by copying each principal.

1.2.5 Generation of Fresh Tokens as a Cryptographic Primitive

As described above, we abstract from the mathematical and implementational details of the cryptographic primitives used and make certain assumptions, e. g., for encryption schemes. As we view the *generation of fresh tokens* as a cryptographic primitive, we have to formulate assumptions for this primitive as well – similar assumptions are presented, e. g., in [CS02].

During this report we will abstract from the details of the generation of fresh tokens and use the following assumptions:

- If a principal A generates a fresh token, no other principal is able to predict or derive that token other than by receiving this information from A, directly or indirectly.
- Two different principals will never generate the same token.
- There is an unbounded set of values for fresh tokens.

Of course, these assumptions are not realistic: As [MVO96] states, nonces are typically implemented by random numbers, sequence numbers, or timestamps; so there is, for example, usually no guaranteed uniqueness. Nevertheless, the possibilities involved here, e. g., for the incidental generation of equal random numbers on two different personal computers, are usually negligibly small.

1.3 Horn Clauses and Selecting Theories

In [Tru05b] and in our model sets of *Horn clauses* are used to describe the principals of a protocol. Horn clauses are named after Alfred Horn who analyzed special classes of clauses in [Hor51]. They are the basis for many deduction methods in artificial intelligence [Rob65] and furthermore, logic programming languages (e. g., PROLOG) are mostly based on Horn clauses [Llo84].

Kowalski gives an introduction to Horn clause logic in [Kow79] and states:

> The majority of formalisms for computer programming bear greater resemblance to Horn clauses than they do to »non-Horn« clauses. In addition, most of the models of problem-solving which have been developed in artificial intelligence can be regarded as models for problems expressed by means of Horn clauses.

Thus, Horn theories – i. e., sets of Horn clauses – have proven to be a good choice for modeling the knowledge and the deduction capabilities of the principals and the intruder, there are various approaches to the verification of cryptographic protocols using Horn clauses [Bla01, VSS05, Tru05b].

Truderung introduces a special class of Horn theories called *selecting theories* [Tru05b], which are used to model the handling of messages by the protocol's principals and the intruder. He uses three kinds of predicates in the Horn clauses: One is modeling the intruder's knowledge, a second kind is used to describe the recursive processing of terms by a principal, and a third kind is modeling tree automata and provides a regular lookahead when processing messages. Upon receiving a message a principal applies a rewrite rule. Then he uses a selecting theory to process the result, deriving new messages that are added to the intruder's knowledge. The intruder is then able to process these messages by a special Horn theory, composing and decomposing terms to construct new messages.

As we will extend Truderung's model in this report, we refer the reader either to Chapter 2 for the details of the extended model or to [Tru05b] for the basic model.

1.4 Structure of this Report

In Chapter 2 we will first provide an example protocol. Then we will present our model, and we will define a security notion for this kind of protocols. We will also show how to model the example protocol, and we will state our main result. In the following two chapters we will prove the decidability result: First, in Chapter 3 we will introduce a graph structure called ADAG which represents an attack on a protocol, i. e., a protocol is insecure if and only if an ADAG for this protocol exists. In Chapter 4 we show that we can always find ADAGs of a bounded size in terms of the size of the protocol, which leads to the possibility to decide security by checking all these bounded ADAGs in bounded time. In the last chapter we will conclude and show possible extensions of our model.

2 Protocol Model and Main Result

In this chapter we will extend the model of »Selecting Theories« developed in [Tru05b] and briefly introduced in Section 1.3 to be able to generate an unbounded number of fresh tokens and thus model nonces and generated keys more accurately. We will start with a running example, the Recursive Authentication Protocol [BO97, Pau97]. Then we will present preliminaries necessary for our model, the latter is then presented in Section 2.3. Afterwards we will show how to model the running example in Section 2.5 and present our main result in Section 2.6.

2.1 A Running Example

To illustrate the usage of our model, we will use the »Recursive Authentication Protocol« which was presented by Bull and Otway [BO97]. After the description of the protocol in this section we will give a formal model of the protocol in our extended model in Section 2.5.

The Recursive Authentication Protocol is an extended version of the authentication protocol by Otway and Rees [OR87]: The latter can only cope with two principals, while the recursive protocol allows an unbounded number of principals in one protocol run. Note that we will use a modified version provided by Paulson [Pau97] – amongst minor differences the original version uses exclusive or (XOR) instead of encryption, allowing a simple attack on the protocol [RS98].

For this protocol we assume that there is a set of principals $\{P_1, \ldots, P_N\}$ who would like to communicate pairwise, i. e., P_n wants to communicate in a secure way with P_{n+1} for all $n \in \{1, \ldots, N-1\}$.

In addition, there is a server P_{Srv} (with $Srv = N + 1$) that, for each $n \in \{1, \ldots, N\}$, shares a long-term key K_n for a symmetric encryption scheme with the principal P_n. The server shall now generate new symmetric session keys $K_{(n,n+1)}$ for the communication between P_n and P_{n+1} for all $n \in \{1, \ldots, N\}$.

To start the protocol P_1 sends the following message to P_2, containing the names P_1 and P_2, a nonce $\hat{N}_1$ generated by P_1 and a constant marking $\perp$ the start of the protocol – using the symbol $\mathtt{H}_K(x)$ for modeling x and a keyed hash of x as explained in Section 2.3.1:

$$P_1 \to P_2 \colon m_1 = \mathtt{H}_{K_1}(P_1, P_2, \hat{N}_1, \perp) \ .$$

Principal P_2 sends the following message to P_3, again containing two names, a nonce N_2 generated by P_2 and the first request received from P_1:

$$P_2 \to P_3 \colon m_2 = \mathtt{H}_{K_2}(P_2, P_3, \hat{N}_2, m_1) \ .$$

Accordingly, P_N receives the message m_{N-1} and sends this message to the server P_{Srv}:

$$P_N \to P_{Srv} \colon m_N = \mathtt{H}_{K_N}(P_N, P_{Srv}, \hat{N}_N, m_{N-1}) \ .$$

First the server processes m_N without knowing the size of m_N in advance, which depends on the number of the protocol's principals. The server generates keys $K_{(n,n+1)}$ for $n \in \{1, \ldots, N\}$ and composes the following messages:

$$\begin{aligned}
m'_N &= \langle \{K_{(N-1,N)}, P_{N-1}, P_N, \hat{N}_N\}_{K_N}, \{K_{(N,Srv)}, P_N, P_{Srv}, \hat{N}_N\}_{K_N} \rangle \ , \\
&\vdots \\
m'_n &= \langle \{K_{(n-1,n)}, P_{n-1}, P_n, \hat{N}_n\}_{K_n}, \{K_{(n,n+1)}, P_n, P_{n+1}, \hat{N}_n\}_{K_n} \rangle \ , \\
&\vdots \\
m'_2 &= \langle \{K_{(1,2)}, P_1, P_2, \hat{N}_2\}_{K_2}, \{K_{(2,3)}, P_2, P_3, \hat{N}_2\}_{K_2} \rangle \ , \\
m'_1 &= \{K_{(1,2)}, P_1, P_2, \hat{N}_1\}_{K_1} \ .
\end{aligned}$$

Second the server sends all these messages to P_N, who will decrypt the message m'_N and forward all other messages to his predecessor P_{N-1}. They will act in the same way, resulting

in the following messages to be sent:

$$\begin{aligned} P_{Srv} &\to P_N \colon \langle m'_1, \ldots, m'_N \rangle \ , \\ P_N &\to P_{N-1} \colon \langle m'_1, \ldots, m'_{N-1} \rangle \ , \\ &\vdots \\ P_3 &\to P_2 \colon \langle m'_1, m'_2 \rangle \ , \\ P_2 &\to P_1 \colon m'_1 \ . \end{aligned}$$

At the end of the protocol run, principals P_n and P_{n+1} share the key $K_{(n,n+1)}$ for all $n \in \{1, \ldots, N\}$.

2.1.1 Fresh Tokens

The Recursive Authentication Protocol is an example of a recursive protocol: The server gets a *linked list* of single requests, which is processed recursively. The size of the total message received by the server and thus the number of elements in the list is neither known in advance nor explicitly bounded by the protocol's description, i. e., the server should be able to process messages of arbitrary length (aside from real-world technical bounds as memory or network capacity).

During this process, the server has to generate keys for each request, hence, the server may have to generate an unbounded number of keys. In the previous section, we modeled this by defining a key for each pair of principals. But this may lead to re-use of keys: If a groups of principals (or the intruder) generate a message containing nested requests for symmetric keys for two principals, the server creates duplicate certificates, i. e., he uses the same key twice for both requests.

Consider the following request received by the server:

$$m = \mathtt{H}_{K_B}(B, Srv, N'_B, \mathtt{H}_{K_A}(A, B, N'_A, \mathtt{H}_{K_B}(B, A, N_B, \mathtt{H}_{K_A}(A, B, N_A, \bot)))) \ . \tag{2.1}$$

The server will send the keys $K_{(B,Srv)}$ and $K_{(B,A)}$, but also keys for the communication between A and B initiated by A, i. e., he will send a key $K_{(A,B)}$. When using a model which has a bounded number of constants, we will usually only have one key $K_{(A,B)}$. The server

has to send the following terms, sending the key $K_{(A,B)}$ *twice* to both A and B:

$$\begin{aligned}
&\langle\{K_{(A,B)}, A, B, N'_B\}_{K_B}\ , \quad \{K_{(B,Srv)}, B, Srv, N'_B\}_{K_B}\rangle\ ,\\
&\langle\{K_{(B,A)}, B, A, N'_A\}_{K_A}\ , \quad \{K_{(A,B)}, A, B, N'_A\}_{K_A}\rangle\ ,\\
&\langle\{K_{(A,B)}, A, B, N_B\}_{K_B}\ , \quad \{K_{(B,A)}, B, A, N_B\}_{K_B}\rangle\ ,\\
&\{K_{(A,B)}, A, B, N_A\}_{K_A}\ .
\end{aligned} \tag{2.2}$$

But this is not what a real server would do. Upon receiving this kind of request, a real server would generate a new key for each of the connections, ignoring the fact that some of these connections are intended for the same pair of principals.

Thus, our goal is to allow the server to generate an unbounded number of fresh tokens so that a verification of such a protocol does not have to rely on a finite set of constants defined in the signature of that protocol.

2.2 Preliminaries

In this section we will introduce basic notations and preliminaries necessary for our model.

2.2.1 Signatures and Terms

The messages used in our model will be terms over a formal term algebra, and we will extend the normal (finite) signature with an infinite set of constants, which will contain constants called *anonymous constants*. In addition, we will use three different kinds of variables: regular variables, *anonymous variables*, and *fresh variables*. The main idea is that the anonymous and fresh variables are typed variables, i. e., the only values that can be assigned to them are anonymous constants. Furthermore, we assume that the substitutions we use cannot assign the same anonymous constant to two different *fresh* variables. Thus, we can guarantee freshness of anonymous constants.

Definition 2.1. Let Σ be a finite signature, i. e., a set of function symbols with an arity. By Σ_i we denote the subset of all function symbols of arity i in Σ. Symbols with arity zero are called *constants*. Let $\Gamma = \{c_1, c_2, \ldots\}$ with $\Gamma \cap \Sigma = \emptyset$ denote a countably infinite set of constants which we call *anonymous constants*.

Let $X, Y, Y^\star$ be disjoint finite sets of variables: The elements $x \in X$ will be called *regular variables*, elements $y \in Y$ will be called *anonymous variables*, and variables $y^\star \in Y^\star$ are called *fresh variables*. We use $V = X \cup Y \cup Y^\star$.

The set of terms over a signature $\bar{\Sigma}$ and a set of variables $\bar{V}$ will be denoted by $T(\bar{\Sigma}, \bar{V})$; we will need the set $T_V = T(\Sigma \cup \Gamma, X \cup Y \cup Y^\star)$. Terms are called *ground* if they do not contain any variables, neither regular nor anonymous nor fresh ones. A term is called *linear* if each variable occurs at most once in it. *Flat* terms are of the form $f(t_1, \ldots, t_m)$ with $f \in \Sigma_m$ and $t_1, \ldots, t_m$ are constants or variables. We will call terms *simple* if they are linear or flat, possibly both.

For a term $t \in T_V$, let $\mathrm{sub}(t)$ denote the set of subterms and $\mathrm{depth}(t)$ the depth of that term, i. e., for all $c \in \Sigma_0 \cup \Gamma \cup V$ and for all $f \in \Sigma_n$ (for some $n \in \mathbb{N}_{>0}$ and terms $t_1, \ldots, t_m \in T_V$) we recursively define:

$$\mathrm{sub}(c) = \{c\}\,, \qquad \mathrm{sub}(f(t_1, \ldots, t_m)) = \{f(t_1, \ldots, t_m)\} \cup \bigcup_{j=1}^{m} \mathrm{sub}(t_j)\,, \tag{2.3}$$

$$\mathrm{depth}(c) = 0\,, \qquad \mathrm{depth}(f(t_1, \ldots, t_m)) = 1 + \max\left\{ \mathrm{depth}(t_j) \;\middle|\; j \in \{1, \ldots, m\} \right\}\,. \tag{2.4}$$

Let $\mathrm{var}(t)$ denote the set of all variables occurring in t, i. e., $\mathrm{var}(t) = \mathrm{sub}(t) \cap V$. For each $V' \subseteq V$ we write $\mathrm{var}_{V'}(t)$ for $\mathrm{var}(t) \cap V'$, e. g., the fresh variables of a term t are denoted by $\mathrm{var}_{Y^\star}(t)$. ◁

2.2.2 Anonymous Substitutions

In order to cope with anonymous and fresh variables, *anonymous* substitutions are defined, which map variables to terms, anonymous and fresh variables to anonymous constants, and different fresh variables to different anonymous constants.

Definition 2.2. A *(ground) substitution* is a mapping $\sigma\colon X \to T(\Sigma \cup \Gamma, X)$ from regular variables to (ground) terms. A substitution σ is extended to a substitution from terms to terms $\hat{\sigma}\colon T(\Sigma \cup \Gamma, X) \to T(\Sigma \cup \Gamma, X)$ in the natural way:

$$\hat{\sigma}(t) = \sigma(t) \qquad \text{for } t \in X\,, \tag{2.5}$$

$$\hat{\sigma}(t) = f(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_m)) \qquad \text{for } t = f(t_1, \ldots, t_m) \text{ with } f \in \Sigma_m\,. \tag{2.6}$$

A function $\sigma\colon X \cup Y \cup Y^\star \to T_V$ is called *anonymous substitution* if the following two conditions hold:

$$\sigma(y) \in \Gamma \qquad \text{for all } y \in Y \cup Y^\star \text{ and} \tag{2.7}$$

$$\sigma(y_1^\star) \neq \sigma(y_2^\star) \qquad \text{for all } y_1^\star, y_2^\star \in Y^\star \text{ with } y_1^\star \neq y_2^\star\,. \tag{2.8}$$

An anonymous ground substitution is defined analogously, and both can be extended to a mapping on terms as above. ◁

In the rest of this report, we will simply use σ and the denotation *anonymous substitution* for both the substitution σ and the extended anonymous substitution $\hat{\sigma}$.

Example 2.3. Let $t = f(y_1^\star, y_2^\star)$ for a function symbol $f \in \Sigma_2$ and two distinct fresh variables $y_1^\star \neq y_2^\star$, an anonymous substitution σ cannot map t to $f(\mathsf{c}, \mathsf{c})$ with $\mathsf{c} \in \Gamma$, instead we have $\sigma(t) = f(\mathsf{c}_1, \mathsf{c}_2)$ for some $\mathsf{c}_1, \mathsf{c}_2 \in \Gamma$ and $\mathsf{c}_1 \neq \mathsf{c}_2$. ◁

2.2.3 Registers and Register Sequences

When processing a message, our principals will have access to a special memory containing a bounded number of anonymous constants, called *register sequence*. The principals can read constants from that memory, move constants within the memory and generate anonymous constants at each single position in the register sequence.

Definition 2.4. A tuple $\kappa = (\kappa_1, \ldots, \kappa_Z) \in (\Gamma \cup Y \cup Y^\star)^Z$ for $Z \in \mathbb{N}_{>0}$ will be called a *register sequence with Z registers*. We will denote κ_i by $\kappa[i]$. We define $\mathrm{var}_Y(\kappa) = \{\kappa_1, \ldots, \kappa_Z\} \cap Y$ and $\mathrm{var}_{Y^\star}(\kappa)$ analogously, let $\mathrm{var}(\kappa) = \mathrm{var}_Y \cup \mathrm{var}_{Y^\star}$.

If we have $\kappa \in (Y \cup Y^\star)^Z$, we call it a *register sequence of variables*, if otherwise we have $\kappa \in \Gamma^Z$, it is called a *ground register sequence*. We will use a special term, called *simple register sequence*, which is defined as $\kappa_\star = (y_1, \ldots, y_Z) \in Y^Z$ with fixed, pairwise distinct anonymous variables $y_1, \ldots, y_Z$. We will later use $\kappa[i]$ to denote the ith element in a register sequence.

Let $\Gamma = \{c_1, c_2, \ldots\}$, then we define $\kappa^{(n)} = (r_{n\cdot Z+1}, \ldots, r_{n\cdot Z+Z})$ for each $n \in \mathbb{N}$. ◁

2.2.4 Horn Theories and Proofs

As stated in Section 1.3, we will use *Horn theories* to model both the protocol's principals and the intruder, allowing recursive computations for the handling of messages.

Definition 2.5. Let P be a set of unary predicate symbols and R be a set of binary predicate symbols. For $p \in P$ and $t \in T_V$ we say $p(t)$ is an *atomic (ground) formula* if t is a (ground) term. For $r \in R$ and a (ground) register sequence κ let $r(t, \kappa)$ also be an atomic (ground) formula. We will use the term *(Horn) fact* for atomic ground formulas.

A *Horn theory* Φ is a finite set of *Horn clauses* of the form $[a_1, \ldots, a_m \to a_0]$ where $a_0, \ldots, a_m$ are atomic formulas. In addition we assume that fresh variables do only appear in a_0. ◁

Definition 2.6. Let Φ be a Horn theory, and let A and B be sets of facts. Let $\Pi = [b_1, \ldots, b_m]$ be a finite sequence of facts with $B \subseteq \{b_1, \ldots, b_m\}$. We further assume that there is a set $\{C_1, \ldots, C_m\}$ of pairwise disjoint subsets of Γ, let $C = \bigcup_{j=1}^{m} C_j$. We assume that the following holds for each $j \in \{1, \ldots, m\}$:

1. Either b_j is an *assumed fact* (a-*fact)*, i. e., $b_j \in A$ and $C_j = \emptyset$; or
2. b_j is a *constructed fact* (c-*fact)*, i. e., there is a clause $\varphi = [t_1, \ldots, t_n \to t_0] \in \Phi$ and an anonymous substitution σ_j such that there exists $\{t'_1, \ldots, t'_n\} \subseteq \{b_1, \ldots, b_{j-1}\}$ with $\sigma(t_0) = b_j$ and $\sigma(t_i) = t'_i$ for all $i \in \{1, \ldots, n\}$; with $C_j = \sigma(\mathrm{var}_{Y^\star}(t_0))$. We call b_j *constructed by* φ or *obtained by* φ.

Such a sequence Π is called *(Horn) proof of B with respect to Φ assuming A assigning C.* ◁

If there is a Horn proof of B with respect to Φ assuming A which assigns C, we write $A \vdash_\Phi^C B$, calling A the *assumption* and B the *conclusion* of the proof. For a single atomic formula b we write $A \vdash_\Phi^C b$ instead of $A \vdash_\Phi^C \{b\}$. If $A \vdash_\Phi^C B$ holds for some $C \subset \Gamma$, we will also omit C and write $A \vdash_\Phi B$.

Note that we use squared brackets to denote sequences, e. g., $\Pi = [b_1, \ldots, b_m]$, and that we will denote the ith element of a sequence Π by $\Pi[i]$, e. g., we have $\Pi[i] = b_i$.

Remark 2.7. In this report we will use algorithms to construct Horn proofs. This may lead to duplicate facts in the same proof. While this is no problem for a-facts and c-facts constructed by clauses without fresh variables, this may lead to two identical facts both being constructed by the same clause containing a fresh variable: This would contradict the assumption that the sets C_i are pairwise disjoint. Therefore, for the rest of the report we assume that each fact is added to a proof only once. ◁

We have now defined all the preliminaries we need to describe our protocol model.

2.3 The Formal Model

2.3.1 Messages

We will start with the term algebra defining the set of messages that can be used in our protocols.

Definition 2.8. Let Σ be a finite signature containing following symbols:

- a subset $K \subseteq \Sigma_0$ of constants modeling keys,
- a constant $\mathfrak{c}$ modeling the intruder's initial knowledge,
- a constant \$ modeling the secret,
- a binary function symbol $\langle \cdot, \cdot \rangle$ for pairing,
- a binary function symbol $\{\cdot\}_\cdot$ for symmetric encryption,
- a binary function symbol $\{\!|\cdot|\!\}_\cdot$ for asymmetric encryption,
- an unary function symbol $\mathtt{h}(\cdot)$ where $\mathtt{h}(t)$ represents the hash of a term t,
- an unary function symbol $\mathtt{H}(\cdot)$, where $\mathtt{H}(t)$ represents the hash of t plus the term t itself,
- one unary function symbol $\mathtt{h}_k(\cdot)$ for each $k \in K$ where $\mathtt{h}_k(t)$ represents the keyed hash of t under the key k,
- one unary function symbol $\mathtt{H}_k(\cdot)$ for each $k \in K$, where $\mathtt{H}_k(t)$ represents the keyed hash of t under the key k plus the term t itself,
- additional constants depending on the protocol, e. g., for names of principals etc.

Note that keys are constants, both for symmetric as for asymmetric encryption. We assume that there exists a bijection $\cdot^{-1}$ on the set of keys which maps every public (private) key k to its corresponding private (public) key k^{-1}.

The elements of the set $H = \{\mathtt{h}, \mathtt{H}, \} \cup \{\mathtt{h}_k, \mathtt{H}_k \mid k \in K\} \subset \Sigma$ are called *hash symbols*. A *hash term* is a term of the form $h(t)$ for some hash symbol $h \in H$.

Let Γ be an infinite set of constants. Ground terms over $\Sigma \cup \Gamma$ are called *messages*. ◁

For the rest of this report, we will consider Σ, K, and Γ to be fixed sets. We will omit the pairing symbols if possible – for instance, we will write $\mathtt{I}(x, y)$ instead of $\mathtt{I}(\langle x, y\rangle)$, or $\langle x_1, \ldots, x_4\rangle$ instead of $\langle x_1, \langle x_2, \langle x_3, x_4\rangle\rangle\rangle$.

2.3.2 Selecting Theories

Now the main ingredient of our model is defined – special Horn theories called *selecting theories*. We will use them to process messages: When a message t is sent over the network, there will be a predicate symbol r such that a principal's processing of that message is handled by a selecting theory applied on the fact $r(t)$.

We will take three different kinds of predicate symbols and accordingly we define three types of clauses, whose function can informally be described as follows:

1. Using *push clauses* we can process messages recursively, moving from a term in the message to one of its subterms (or some of it subterms in parallel). There will be a register sequence where we can store anonymous constants as explained above, and we will be able to generate anonymous constants during each push step.
2. *Send clauses* allow principals to send messages to the network – and thus directly to the intruder, as proposed by the Dolev-Yao model.
3. *Pop clauses* will provide a lookahead when processing messages as they can simulate runs of a nondeterministic bottom-up tree automata when viewing the message terms as trees: They assign pop symbols from a set of predicate symbols Q to subterms of messages, allowing the use of these symbols in other clauses. Note that we will be able to assign pop symbols to anonymous constants, and that our pop clauses are even able to compare parts of messages.

For the resemblance between pop clauses and tree automata we refer the reader to [Tru05a].

Definition 2.9. Let Q and R be disjoint sets of predicate symbols. The elements of Q are assumed to be unary predicate symbols and are called *pop symbols*, the elements of R are binary predicate symbols called *push symbols*. Let $\mathtt{I}$ be an unary predicate symbols, it will model the intruder's knowledge.

Let κ, κ' be register sequences of variables with $\mathrm{var}(\kappa) \subseteq Y$ and $\mathrm{var}(\kappa') \subseteq \mathrm{var}(\kappa) \cup Y^\star$. Let $m \in \mathbb{N}$, $j \in \{1, \ldots, m\}$, pop symbols $q, q_1, \ldots, q_m \in Q$, push symbols $r, r' \in R$, function symbols $f \in \Sigma_m \cup Y$ and $g \in \Sigma_m$, variables $x_1, \ldots, x_m \in X$. Let $t_r, t_\mathtt{I} \in T(\Sigma, X)$ be simple terms with $\mathrm{depth}(t_r) > 0$, let $x_r \in \mathrm{var}(t_r)$, let $s_\mathtt{I} \in T(\Sigma, \mathrm{var}(t_\mathtt{I}) \cup \mathrm{var}(\kappa))$.

We define the following three types of Horn clauses:

$$\textit{pop clauses} \qquad q_1(x_1), \ldots, q_m(x_m) \to q(f(x_1, \ldots, x_m)) \,, \tag{2.9 a}$$

$$\textit{push clauses} \qquad q_1(t_r), \ldots, q_m(t_r), r(t_r, \kappa) \to r'(x_r, \kappa') \,, \tag{2.9 b}$$

$$\textit{send clauses} \qquad q_1(t_{\mathtt{I}}), \ldots, q_m(t_{\mathtt{I}}), r(t_{\mathtt{I}}, \kappa) \to \mathtt{I}(s_{\mathtt{I}}) \,. \tag{2.9 c}$$

Pop, push, and send clauses are special Horn clauses. Hence, a finite set Φ of such clauses is a special Horn theory called *selecting theory using anonymous constants over* (Q, R) *with* Y *registers*. ◁

We remark that by using an anonymous variable $f \in Y$ (and thus, $m = 0$) we are able to assign pop symbols to anonymous constants.

Example 2.10. Consider the following selecting theory Φ containing three clauses over $Q = \{q\}$ and $R = \{r\}$ with one register, regular variables x, x', an anonymous variable y and a fresh variable $y^\star$:

$$\to q(y) \qquad \text{assign a pop symbol to all anonymous constants,} \tag{2.10}$$

$$r(\langle x, x' \rangle, y) \to r(x', y^\star) \qquad \text{descend in the right component,} \tag{2.11}$$

$$q(x), r(x, y) \to \mathtt{I}(\langle x, y \rangle) \qquad \text{send } x \text{ and a fresh anonymous constant.} \tag{2.12}$$

Now consider a term $t = \langle \langle c_1, c_2 \rangle, \langle d, c_3 \rangle \rangle$ with anonymous constants $c_1, c_2, c_3 \in \Gamma$ and $d \in \Sigma_0$. Then we have $r(t,\ c_4) \vdash_\Phi^{\{c_5, c_6\}} \mathtt{I}(\langle c_3, c_6 \rangle)$ as the following Horn proof shows:

$$r(\langle \langle c_1, c_2 \rangle, \langle d, c_3 \rangle \rangle,\ c_4) \qquad \text{a-fact} \tag{2.13}$$

$$r(\langle d, c_3 \rangle,\ c_5) \qquad \text{c-fact by applying (2.11) on (2.13)} \tag{2.14}$$

$$r(c_3,\ c_6) \qquad \text{c-fact by applying (2.11) on (2.14)} \tag{2.15}$$

$$q(c_3) \qquad \text{c-fact by applying (2.10)} \tag{2.16}$$

$$\mathtt{I}(\langle c_3, c_6 \rangle) \qquad \text{c-fact by applying (2.12) on (2.15) and (2.16)} \tag{2.17}$$

But we will never be able to construct a fact $\mathtt{I}(\langle d, c' \rangle)$ for any c' because we cannot construct the fact $q(d)$. ◁

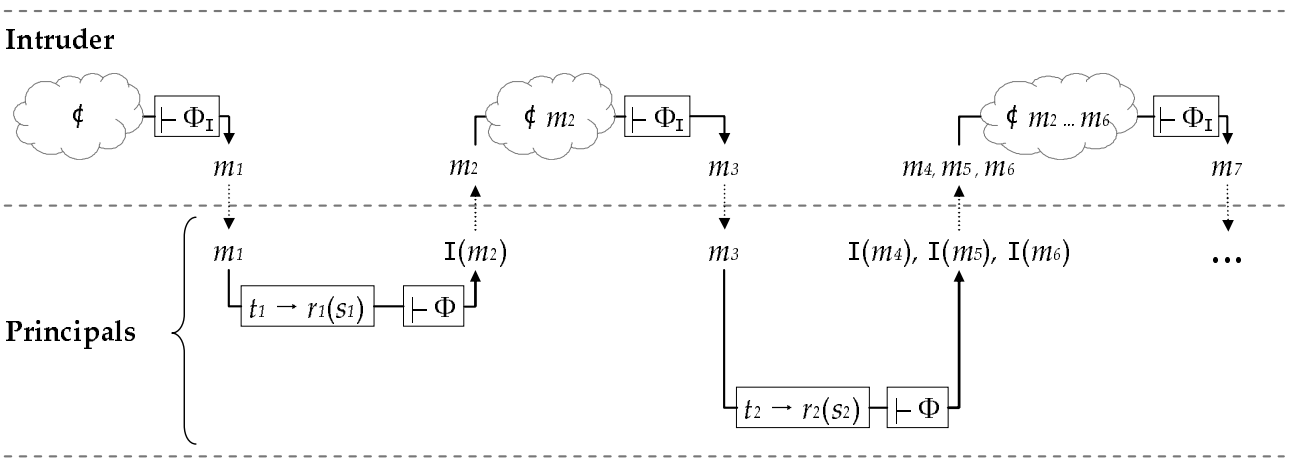

Figure 2.1: The run of a protocol: The intruder derives a message m_1, a principal applies a protocol step $t_1 \to r_1(s_1)$ and derives a message m_2 sent to the intruder and so on.

2.3.3 Principals and Protocols

Now we can define principals which mainly consists of rewrite rule which is then processed by a selecting theory.

Definition 2.11. A *protocol step* τ over (Q, R) is of the form (t, r, s), where $t, s \in T(\Sigma, X)$ are terms and $r \in R$ is a push symbol.

A *principal* $\vec{\tau}$ over (Q, R) is a finite sequence $[\tau_1, \ldots, \tau_N]$ of protocol steps such that, for every $n \in \{1, \ldots, N\}$ and $\tau_n = (t_n, r_n, s_n)$ we have

$$\mathrm{var}(s_n) \subseteq \bigcup_{i=1}^{n} \mathrm{var}(t_i) \,. \tag{2.18}$$

A *protocol* over (Q, R) with Z registers is a pair $\mathcal{P} = (P, \Phi)$ consisting of a finite set of principals P and a selecting theory Φ using anonymous constants over (Q, R) with Z registers. ◁

By abuse of notation we will write $t \to r(s)$ instead of (t, r, s) even if $r(s)$ is no atomic formula because of the arity of r.

We will now explain the behavior of principals in our model – see Figure 2.1 for illustration, with the intruder theory Φ_I as introduced in the next section.

Let Φ be a selecting theory with anonymous constants over (Q, R). Assume that we have a principal which has a protocol step $t \to r(s)$ with $r \in R$, and he is receiving a term $\sigma(t)$ for a ground substitution σ. Let κ be a register sequence containing fresh anonymous constants, i. e., constants not used so far in the run of the protocol. The principal will now send each term s' to the network for which $r(\sigma(s), \kappa) \vdash^{C}_{\Phi} \mathtt{I}(s')$ holds – for a set of fresh anonymous constants C not used so far and not used to construct any other term s''.

In fact, there could be an infinite number of messages s' as there are infinitely many possibilities to choose each anonymous constant – but we will show that for analyzing security it is sufficient to check a bounded number of messages. Note that the number of messages sent to the network in one protocol step is not bounded in terms of the size of the protocol, even if we only consider messages that do not contain any anonymous constant.

Remark 2.12. It is also possible to model non-recursive protocol steps of the form $t \to \mathtt{I}(s)$: For each such step τ, we take an unused push symbol $r^{\tau}_{\mathtt{I}} \in R$ and add the following send clause to the selection theory:

$$r^{\tau}_{\mathtt{I}}(x, \kappa_\star) \to \mathtt{I}(x) \; . \tag{2.19}$$

This allows us to express $t \to \mathtt{I}(s)$ by $r \to r^{\tau}_{\mathtt{I}}(s)$. ◁

Remark 2.13. For the rest of this report we will consider the sets of predicate symbols Q, and R as well as the number of registers Z to be fixed, i. e., when referring to a protocol we will omit »over (Q, R) with Z registers«. ◁

2.3.4 The Intruder

Our threat model is the same as presented in Section 1.2. Thus, we will describe an intruder that is able to read, modify, delete, replay or create all messages sent over the network. But, according to the level of abstraction of the Dolev-Yao model, he is not able to apply any form of cryptanalysis or similar, e. g., he is only able to read and decompose messages that are not encrypted at all or encrypted with keys he knows. These abilities of the intruder, i. e., to decompose messages and generate new messages, are also modeled by a Horn theory.

Definition 2.14. The *intruder theory* $\Phi_{\mathtt{I}}$ is defined as follows for $x, x_1, x_2 \in X$ and each key

$k \in K$:

$$\mathtt{I}(\langle x_1, x_2\rangle) \to \mathtt{I}(x_1)\,, \qquad \mathtt{I}(\{x\}_k), \mathtt{I}(k) \to \mathtt{I}(x)\,, \tag{2.20 a}$$

$$\mathtt{I}(\langle x_1, x_2\rangle) \to \mathtt{I}(x_2)\,, \qquad \mathtt{I}(\{|x|\}_k), \mathtt{I}(k^{-1}) \to \mathtt{I}(x)\,, \tag{2.20 b}$$

$$\mathtt{I}(\mathtt{H}(x)) \to \mathtt{I}(x)\,, \qquad \mathtt{I}(\mathtt{H}_k(x)) \to \mathtt{I}(x)\,, \tag{2.20 c}$$

$$\mathtt{I}(\mathtt{H}(x)) \to \mathtt{I}(\mathtt{h}(x))\,, \qquad \mathtt{I}(\mathtt{H}_k(x)) \to \mathtt{I}(\mathtt{h}_k(x))\,, \tag{2.20 d}$$

$$\mathtt{I}(x_1), \mathtt{I}(x_2) \to \mathtt{I}(\langle x_1, x_2\rangle)\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\{x\}_k)\,, \tag{2.20 e}$$

$$\mathtt{I}(x), \mathtt{I}(k) \qquad \to \mathtt{I}(\{|x|\}_k)\,, \tag{2.20 f}$$

$$\mathtt{I}(x) \to \mathtt{I}(\mathtt{h}(x))\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{h}_k(x))\,, \tag{2.20 g}$$

$$\mathtt{I}(x) \to \mathtt{I}(\mathtt{H}(x))\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{H}_k(x))\,. \tag{2.20 h}$$

$$\mathtt{I}(x), \mathtt{I}(\mathtt{h}_k(x)) \to \mathtt{I}(\mathtt{H}_k(x))\,, \tag{2.20 i}$$

For a set of messages A, let $\mathtt{I}(A) = \{\mathtt{I}(t) \mid t \in A\}$. We define that *the intruder can derive a message t from A* if $\mathtt{I}(A) \vdash_{\Phi_\mathtt{I}} \mathtt{I}(t)$. ◁

As later stated in Lemma 3.1, this derivation can be divided into two phases: *decomposition*, clauses (2.20 a) to (2.20 d), followed by *composition*, clauses (2.20 e) to (2.20 i).

Note that we have no special clauses for anonymous constants – for the intruder, these constants are not distinguishable from regular constants. By the principal described in 2.4.1 we enable the intruder to use anonymous constants, but this is possible within the means presented so far.

2.3.5 Attacks and Security

Definition 2.15. Let $\mathcal{P} = (P, \Phi)$ be a protocol with M principals $\vec{\tau}^1, \ldots, \vec{\tau}^M$, where for each $m \in \{1, \ldots, M\}$ the principal $\vec{\tau}^m$ has N_m protocol steps $\tau_1^m, \tau_2^m, \ldots, \tau_{N_m}^m$.

An *execution scheme* π for $\mathcal{P}$ is a sequence $[\pi_1, \ldots, \pi_N]$ of protocol steps if there is a function $f\colon \{1, \ldots, N\} \to \{1, \ldots, M\}$ that witnesses for each protocol step in which principal this protocol step occurs, i. e., has the following property: $f^{-1}(m) = \{p_1, \ldots, p_{N'_m}\}$ with $p_1 < p_2 < \cdots < p_{N'_m}$ and $N'_m \leq N_m$ for all $m \in \{1, \ldots, M\}$, and $\pi_{p_n} = \tau_n^m$ for all $n \in \{1, \ldots, N'_m\}$. ◁

To complete our model we define the security of a protocol – keep in mind the illustration in Figure 2.1.

Definition 2.16. Let (π, σ) be a pair with a ground substitution σ and an execution scheme $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$ for a protocol $\mathcal{P}$, and set $t_{N+1} = \$$.

We take the pairwise disjoint ground register sequences $\kappa^{(1)}, \ldots, \kappa^{(N)}$ we defined earlier and set C_0 to contain all anonymous constants in these N register sequences. We further assume that $\{C_0, \ldots, C_N\}$ is a set of pairwise disjoint subsets of Γ.

Let there be another set $\{T_1, \ldots, T_N\}$ of sets of messages sent to the intruder during that step, i. e., T_n is a set of facts of the form $\mathtt{I}(t)$ for messages t, such that the following conditions hold:

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash_{\Phi}^{C_n} T_n \qquad \text{for all } n \in \{1, \ldots, N\} \text{ and} \tag{2.21 a}$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup T_1 \cup \cdots \cup T_n \vdash_{\Phi_{\mathtt{I}}} \mathtt{I}(\sigma(t_{n+1})) \qquad \text{for all } n \in \{0, \ldots, N\}\,. \tag{2.21 b}$$

Then (π, σ) is called an *attack* on $\mathcal{P}$. A protocol is called *insecure* if there is an attack on it, otherwise it is called *secure*. ◁

This concludes the definition of our model. In the next section we will single our some features of our model, whereas in Section 2.5 we will show how to model the Recursive Authentication Protocol.

2.4 Features of the Model

2.4.1 Fresh Tokens for the Intruder

In the real world, an intruder can obtain an arbitrary number of fresh tokens by simply generating them randomly. Our intruder, on the other hand, is not yet able to generate fresh anonymous constants.

We present one possible way to allow the intruder to generate constants: a principal whose only purpose is to provide the intruder with fresh tokens. We leave it up to the user of our protocol model to add such a principal if appropriate.

This affects the security of a protocol – i. e., a protocol without this principal may be secure in our model, while adding the principal makes it insecure. But as stated above, this

change to the notion of »secure« is usually justified, as a real intruder is able to generate fresh tokens.

This principal has the following protocol steps, with $x_\star$ being a regular variable not used elsewhere and $r_\star$ being a new push symbol:

$$x_\star \to r_\star(x_\star) \,. \tag{2.22}$$

The following clauses are added to the selecting theory with $y^\star$ being a fresh variable and y_z being an anonymous variable for each $z \in \{1, \ldots, Z\}$:

$$r_\star(\langle x_0, x_1\rangle, (y_1, y_2, \ldots, y_Z)) \to r_\star(x_1, (y_\star, y_2, \ldots, y_Z)) \,, \tag{2.23}$$

$$r_\star(\langle \text{ç}, x_1\rangle, (y_1, y_2, \ldots, y_Z)) \to \mathtt{I}(y_1) \,. \tag{2.24}$$

If the intruder wants to generate i anonymous constants, he constructs the following message:

$$\langle \underbrace{\text{ç}, \ldots, \text{ç}}_{(i+1) \text{ times}} \rangle \,. \tag{2.25}$$

Upon receiving this message the new principal replies with i messages each containing a fresh anonymous constant.

Note that our principals are not able to distinguish between different anonymous constants because there is no possibility to check anonymous constants for inequality. Hence, it would also be sufficient to allow the intruder to derive only one fresh anonymous constant.

2.4.2 Challenges to the Intruder

As later used in Section 2.5, we can add a principal which models security by using an additional key K_Λ and the following clause for the selecting theory with an unused regular variable x_Λ:

$$\langle x_\Lambda, \{x_\Lambda\}_{K_\Lambda}\rangle \to \mathtt{I}(\$) \,. \tag{2.26}$$

Any principal can now claim a whole term t to remain secret by sending $\{t\}_{K_\Lambda}$ to the intruder: If the intruder is able to derive t, then he is able to send $\langle t, \{t\}_{K_\Lambda}\rangle$ to the additional principal, which replies with \$.

On the other hand, the intruder is not able to extract t from $\{t\}_{K_\Lambda}$ because of the assumption that the key K_Λ is not used anywhere else in the protocol.

2.5 Modeling the Example Protocol

To illustrate our extension in comparison with [Tru05b], we give a formal model of the Recursive Authentication Protocol as described in Section 2.1. Our model presented here is slightly imprecise as the server does originally send a *list* of messages, which cannot be modeled by the means available – our server just sends each message on its own. Additionally, our principals do not forward the server's messages to their predecessors. One can show that both imprecisions are not relevant for the security of the protocol.

We will start by defining the protocol in the basic model presented in [Tru05b] and then show how our extension of the model improves the analysis.

2.5.1 Basic Model of the Protocol

We assume that there are $N+1$ principals $\{P_1, \ldots, P_N, P_{Srv}\}$ using this protocol, with P_{Srv} being the server (again, $Srv = N+1$). Besides the constants and function symbols from Definition 2.8, the signature Σ contains the following constants:

- for each $n \in \{1, \ldots, N, Srv\}$ the principal's name P_n, a nonce $\hat{N}_n$, and a key K_n which is shared between the principal and the server,
- for each possible pair of principals (n, m) with $Srv \neq n \neq m$, a key $K_{(n,m)}$ which can be sent to the principals by the server,
- the symbol $\perp$ necessary for the initialization of the protocol, and
- a key K_Λ which is used for formulating challenges.

Let $x_{Srv}, x_\Lambda, x_{(a,b)}, x_{(a,b,c,d)}$ be variables for $a, b, c \in \{1, \ldots, N, Srv\}$ and $d \in \{1, \ldots, 5\}$. We will now define the protocol steps and the selecting theory of the protocol:

Security

For modeling security as explained in Section 2.4.2, we first add a principal which uses the key K_Λ, i. e., the principal has only one protocol step:

$$\langle x_\Lambda, \{x_\Lambda\}_{K_\Lambda} \rangle \to \mathtt{I}(\$) \; . \tag{2.27}$$

The following principals can now state »The term t should remain secret« by sending $\{t\}_{K_\Lambda}$ to the intruder: As long as the key K_Λ is, besides in this principal, only used for *en*cryption, the intruder can only obtain \$ from this principal if he can derive a term t for which he knows $\{t\}_{K_\Lambda}$.

Initiator

Protocol steps for each $n \in \{1, \ldots, N\}$, $m \in \{1, \ldots, N, Srv\}$ (with $\bar{x} = x_{(n,m)}$):

$$\dot{\varsigma} \to \mathtt{I}(\mathtt{H}_{K_n}(P_n, P_m, \hat{N}_n, \bot)) \; , \tag{2.28}$$

$$\{\bar{x}, P_n, P_m, \hat{N}_n\}_{K_n} \to \mathtt{I}(\{\bar{x}\}_{K_\Lambda}) \; . \tag{2.29}$$

The last step models that the term $\bar{x}$ sent to the principal in step 2 should remain secret during the run of the protocol – this is necessary because the initiator would use $\bar{x}$ as the key established by this protocol. Now if the intruder is able to derive $\bar{x}$, he will also be able to derive \$ because of a the first principal, resulting in the protocol being called insecure.

Inner Principals

Protocol steps for each $n, m_1 \in \{1, \ldots, N\}$ and $m_2 \in \{1, \ldots, N, Srv\}$ (with $\bar{x}_l = x_{(n,m_1,m_2,l)}$):

$$\mathtt{H}_{\bar{x}_1}(P_{m_1}, P_n, \bar{x}_2, \bar{x}_3) \to \mathtt{I}(\mathtt{H}_{K_n}(P_n, P_{m_2}, \hat{N}_n, \mathtt{H}_{\bar{x}_1}(P_{m_1}, P_n, \bar{x}_2, \bar{x}_3))) \; , \tag{2.30}$$

$$\langle \{\bar{x}_4, P_{m_1}, P_n, \hat{N}_n\}_{K_n}, \{\bar{x}_5, P_n, P_{m_2}, \hat{N}_n\}_{K_i} \rangle \to \mathtt{I}(\{x_4\}_{K_\Lambda}, \{x_5\}_{K_\Lambda}) \; . \tag{2.31}$$

Both $\bar{x}_4$ and $\bar{x}_5$ correspond to keys received from the server and thus should remain secret, as again stated by the last send action.

Server

The server is modeled with only one protocol step:

$$x_{Srv} \to r_{Srv}(x_{Srv}) \,. \tag{2.32}$$

The following clauses are used as the selecting theory of the protocol for each $n, m_1 \in \{1, \ldots, N\}$ and $m_2 \in \{1, \ldots, N, Srv\}$:

$$r_{Srv}(\mathtt{H}_{K_n}(P_n, P_{m_2}, x_1, x_2)) \to r_{Srv}(x_2) \,, \tag{2.33}$$

$$\begin{aligned} r_{Srv}(\mathtt{H}_{K_n}(P_n, P_{m_2}, x_1, \mathtt{H}_{K_{m_1}}(P_{m_1}, P_n, x_2, x_3))) \to\ & \mathtt{I}(\{K_{(m_1,n)}, P_{m_1}, P_n, x_1\}_{K_n}) \,, \\ & \mathtt{I}(\{K_{(n,m_2)}, P_n, P_{m_2}, x_2\}_{K_n}) \,, \\ & \mathtt{I}(\{K_{(m_1,n)}\}_{K_\Lambda}) \,, \mathtt{I}(\{K_{(n,m_2)}\}_{K_\Lambda}) \,, \end{aligned} \tag{2.34}$$

$$\begin{aligned} r_{Srv}(\mathtt{H}_{K_n}(P_n, P_m, x_1, \bot)) \to\ & \mathtt{I}(\{K_{(n,m)}, P_n, P_m, x_1\}_{K_n}) \,, \\ & \mathtt{I}(\{K_{(n,m)}\}_{K_\Lambda}) \,. \end{aligned} \tag{2.35}$$

Additional Principal: Constants

An additional principal is necessary to allow the intruder to gain knowledge of the commonly used constants, i. e., the names of the principals and $\bot$:

$$¢ \to \mathtt{I}(\bot, P_1, \ldots, P_N, P_{Srv}) \,. \tag{2.36}$$

2.5.2 Extending the Model of the Protocol

We will now modify the selecting theory presented in the previous section to use our extended model. We use $Z = 2$ as the number of registers. We still need most of the constants defined above, but not the keys $K_{(n,m)}$.

The protocol's principals are modeled in exactly the same way, we only have to change the clauses of the selecting theory. Hence, in our model we use the following selecting theory,

replacing the clauses (2.33) to (2.35):

$$\begin{aligned} r_{Srv}(\mathtt{H}_{K_i}(P_i, P_{j_2}, x_1, x_2), (y_1, y_2)) \rightarrow & r_{Srv}(x_2, (y_2, y_\star)) \,, \\ & \mathtt{I}(\{y_1\}_{K_\Lambda}) \,, \end{aligned} \tag{2.37}$$

$$\begin{aligned} r_{Srv}(\mathtt{H}_{K_i}(P_i, P_{j_2}, x_1, \mathtt{H}_{K_{j_1}}(P_{j_1}, P_i, x_2, x_3)), (y_1, y_2)) \rightarrow & \mathtt{I}(\{y_2, P_{j_1}, P_i, x_1\}_{K_i}) \,, \\ & \mathtt{I}(\{y_1, P_i, P_{j_2}, x_2\}_{K_i}) \,, \end{aligned} \tag{2.38}$$

$$r_{Srv}(\mathtt{H}_{K_i}(P_i, P_j, x_1, \bot), (y_1, y_2)) \rightarrow \mathtt{I}(\{y_1, P_i, P_j, x_1\}_{K_i}) \,. \tag{2.39}$$

We also add a principal that is generating anonymous constants for the intruder as described in Section 2.4.1.

This model of the protocol is slightly more accurate than the one presented first. Indeed, we did not model the nonces used by the principals by anonymous constants. But this is not necessary because the total number of nonces used by the principals in the protocol is bounded by the number of principals.

But the key generation of the server is modeled using anonymous constants – thus, we eliminated the problem described in Section 2.1.1, i. e., the server will no longer »generate« the same key twice.

Note that in this protocol such a situation can only occur if there are multiple principals with the same name, which would be the case if we had $P_i = P_j$ in our model for $i \neq j$, or if the intruder would be able to modify or generate a message to the server such that it contains multiple requests for a single connection as shown in Section 2.1.1.

2.6 Main Result

Theorem 2.17 (Main Theorem)**.** *Let $\mathcal{P}$ be a protocol using anonymous constants. For a bounded number of principals, there is an algorithm that decides in nondeterministic double exponential time if $\mathcal{P}$ is secure.*

To prove this theorem, we will later describe the algorithm which nondeterministically guesses a possible attack and checks if the guess is an attack. To do so, we present a structure called *directed acyclic graph of the attack* (ADAG) introduced in [Tru05a] and extended in [Tru05b], which represents an attack on a protocol.

The proof has two main steps: First we will show that a protocol is insecure if and only if there is an ADAG for that protocol. Second we will prove that for each ADAG we find one of at most double exponentially bounded size, resulting in the possibility to guess and examine each ADAG in nondeterministic double exponential time.

More precisely, in Chapter 3 we will begin with transforming the definition of an attack presented above to a characterization more suitable for defining an ADAG. This transformation consists of two steps: First, we will merge the selecting theory of a protocol with the intruder theory, resulting in a Horn theory called the *theory of a protocol* (see 3.2) which can directly obtain the additions to the intruder's knowledge from a protocol step $t \rightarrow r(s)$. Then we will extend this Horn theory to a selecting theory called *stage theory* (see 3.3), which is not modeling one protocol step at a time, but the whole run of the protocol in one application of the theory. The characterization of an attack by a stage theory is then related to an ADAG, which is a graph structure with special labeling functions, altogether restricted by a set of local properties defined in 3.5.

Subsequently, in Chapter 4 we will show how to scale down ADAGs. In 4.2 an ADAG will be transformed to *simple* ADAGs, i. e., we further restrict the structure of an ADAG. In 4.3 we scale down the number of a certain kind of nodes called *goals*, and finally in 4.4 we will show how to scale down the number of all nodes in an ADAG by merging nodes until the whole ADAG has a bounded size. Finally in 4.5 we can present an algorithm which decides security by nondeterministically guessing and checking ADAGs.

3 The Graph of an Attack

3.1 The Intruder Theory

We will begin with the proof of the following lemma mentioned in the definition of the intruder in Section 2.3, which is based on [CJM98]:

Lemma 3.1 (Intruder Theory)**.** *Let A and B be sets of messages. It holds $A \vdash_{\Phi_I} B$ if and only if there exists a Horn proof $b_1, \ldots, b_n$ of B with respect to Φ_I assuming A such that there are numbers $l_1 \leq l_2 \leq n$ with the following properties:*

- *$b_1, \ldots, b_{l_1} \in A$ are a-facts,*
- *$b_{l_1+1}, \ldots, b_{l_2}$ are c-facts obtained by clauses* (2.20 a) *to* (2.20 d)*, and*
- *$b_{l_2+1}, \ldots, b_n$ are c-facts obtained by clauses* (2.20 e) *to* (2.20 i)*.*

Proof. Let A and B be sets of messages. Clauses (2.20 a) to (2.20 d) will be called *decomposition clauses*, clauses (2.20 e) to (2.20 i) *composition clauses*.

One implication of the lemma directly holds: If there exists a Horn proof with the restrictions mentioned in the lemma, then we have $A \vdash_{\Phi_I} B$.

We show the other implication: We assume that $A \vdash_{\Phi_I} B$. This means there is a Horn proof $b_1, \ldots, b_n$ of B with respect to Φ_I assuming A, we now show that there is a sequence $[b'_1, \ldots, b'_{n'}] \subseteq \{b_1, \ldots, b_n\}$ which still is a Horn proof of B, but with $l_1 \leq l_2 \leq m$ as mentioned above.

By definition of a Horn proof, all the a-facts $b_i \in A$ occurring in the proof do not depend on other facts in the proof, hence, we can assume that we find an index l_1 with $b_1, \ldots, b_{l_1} \in A$ and $b_{l_1+1}, \ldots, b_n \notin A$.

As long as we do not find an l_2 satisfying the condition in the lemma for $b_{l_1+1}, \ldots, b_n$, we repeat the following steps:

- First we find an index $i > l_1$ with b_i being obtained by a composition clause, but b_{i+1} being obtained by a decomposition clause – we call such a situation an inversion. Such an index exists because we have moved all facts from A to the beginning of the Horn proof and we cannot find an index l_2.
- Let the fact b_i be produced by a composition clause $c_1 \to c_0$ or $c_1, c_2 \to c_0$; and b_{i+1} be produced by a decomposition clause $d_1 \to d_0$ or $d_1, d_2 \to d_0$.

  Consider three cases:

  a) If $c_0 \neq d_1$, then b_{i+1} does not rely on b_i: We find an anonymous substitution σ and sets with $\sigma(d_0) = b_{i+1}$, and with $\sigma(d_1), \sigma(d_2) \in \{b_1, \ldots, b_{i-1}\}$ (analogously if there is no d_2).

  Then we can swap both steps: We set $n' = n$, $b'_i = b_{i+1}$, and $b'_{i+1} = b_i$, as well as $b'_j = b_j$ for all $j \neq i, i+1$.

  b) If $c_0 = d_1$ with $c_0 = \mathtt{I}(\mathtt{H}(x))$ or $c_0 = \mathtt{I}(\mathtt{H}_k(x))$ and $d_0 = \mathtt{I}(\mathtt{h}(x))$ or $d_0 = \mathtt{I}(\mathtt{h}_k(x))$, one can directly obtain d_0 from the c_1 (and possibly c_2).

  As above, we swap both steps: We set $n' = n$, $b'_i = b_{i+1}$, and $b'_{i+1} = b_i$, as well as $b'_j = b_j$ for all $j \neq i, i+1$.

  c) If $c_0 = d_1$, but case b) did not apply, then b_{i+1} is a fact which is already known before b_i is composed, i. e., we have $d_0 \in \{c_1, c_2\}$ and we can delete step $i+1$ of the Horn proof: We set $n' = n - 1$, $b'_j = b_j$ for all $j \leq i$, and $b'_j = b_{j+1}$ for all $j > i$.

We now repeat these steps as long as there is an index i as above, using $b'_1, \ldots, b'_m$ as the new Horn proof in the next iteration.

In every iteration we shorten the proof by one fact or eliminate an inversion, hence, we can have no more than n^2 iterations. As soon as there is no index i left, the proof meets the requirements mentioned in the lemma. □

3.2 The Theory of a Protocol

In our model each of the intruder's steps consists of composing a term and sending it to a principal, after which the principal applies a protocol step and uses the selecting theory to determine which terms are sent to the intruder.

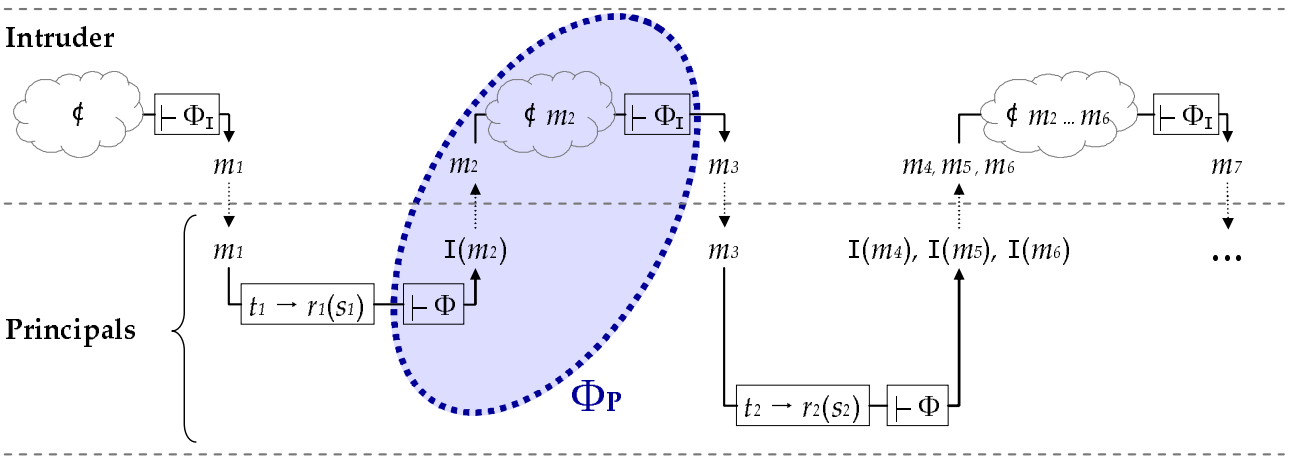

Figure 3.1: Merging the selecting theory Φ and the intruder theory Φ_I into the theory of a protocol $\Phi_{\mathcal{P}}$

As shown in Figure 3.1 we now merge two steps: After a principal applies the selecting theory Φ of the protocol, the intruder uses his theory Φ_I to derive new messages. Because both Φ and Φ_I are Horn theories, we can merge them into one Horn theory, the *theory of a protocol*.

In order to bound the ADAG in later steps of our proof, we introduce an optimization: If the selecting theory contains a send clause constructing a term t, the intruder may be able to decompose this term – possibly because he already learned some keys used in the protocol – and derive a subterm $s \in \mathrm{sub}(t)$. Therefore, we add an additional send clause to the theory of the protocol which directly sends the term s if the intruder knows the necessary keys.

First we define a relation on the set of keys expressing accessibility: For a send clause φ and a set of keys K we need to know which subterms of the send term can be derived by the intruder after the clause φ has been applied, assuming the intruder already knew the keys in K.

Definition 3.2. Let K be a set of keys. We will inductively define the relation $\rightrightarrows_K \subseteq T_V \times T_V$. Let t, s be terms and let $k \in K$ be a key. Then we set:

$$t \rightrightarrows_\emptyset t\,, \tag{3.1 a}$$

$$\langle t, s\rangle \rightrightarrows_\emptyset t\,, \qquad \{t\}_k \rightrightarrows_{\{k\}} t\,, \tag{3.1 b}$$

$$\langle t, s\rangle \rightrightarrows_\emptyset s\,, \qquad \{\!|t|\!\}_k \rightrightarrows_{\{k^{-1}\}} t\,, \tag{3.1 c}$$

$$\mathtt{H}(t) \rightrightarrows_\emptyset t\,, \qquad \mathtt{H}_k(t) \rightrightarrows_\emptyset t\,, \tag{3.1 d}$$

If $t_1 \rightrightarrows_{K_1} t_2$ and $t_2 \rightrightarrows_{K_2} t_3$ for terms t_1, t_2, t_3, we define $t_1 \rightrightarrows_{K_1 \cup K_2} t_3$, if not already $t_1 \rightrightarrows_{K_3} t_3$ for a set $K_3 \subsetneq K_1 \cup K_2$. ◁

For example, we have $a \rightrightarrows_{\emptyset} a$, $\{a\}_k \rightrightarrows_{\{k\}} a$, and $\langle k_1, \{\!|\{a\}_{k_1}|\!\}_{k_2}\rangle \rightrightarrows_{\{k_2^{-1}\}} a$.

In addition, we need to generalize the notion of push and send clauses:

Definition 3.3. A *generalized push clause* is a push clause where facts of the form $\mathtt{I}(k)$ for keys $k \in K$ are allowed on the left-hand-side. *Generalized send clauses* also allow facts of the form $\mathtt{I}(k)$ on the left-hand-side, but in addition they allow facts of the form $q(s_{\mathtt{I}})$ on the left-hand-side for a pop symbol q with $\mathtt{I}(s_{\mathtt{I}})$ being the fact on the right-hand-side of the send clause.

Let κ, κ' be register sequences of variables with $\mathrm{var}(\kappa) \subseteq Y$ and $\mathrm{var}(\kappa') \subseteq \mathrm{var}(\kappa) \cup Y^\star$. Let $m \in \mathbb{N}$, $j \in \{1, \dots, m\}$, pop symbols $q_1, \dots, q_m, q'_1, \dots, q'_m \in Q$, push symbols $r, r' \in R$, and keys $k_1, \dots, k_n \in K$. Let $t_r, t_{\mathtt{I}} \in T(\Sigma, X)$ be simple terms with $\mathrm{depth}(t_r) > 0$, let $x_r \in \mathrm{var}(t_r)$, let $s_{\mathtt{I}} \in T(\Sigma, \mathrm{var}(t_{\mathtt{I}}) \cup \mathrm{var}(\kappa))$. Then we define

$$\mathtt{I}(k_1), \dots, \mathtt{I}(k_n), q_1(t_r), \dots, q_m(t_r), r(t_r, \kappa) \to r'(x_r, \kappa')\,,$$
generalized push clauses, (3.2 b)

$$q'_1(s_{\mathtt{I}}), \dots, q'_m(s_{\mathtt{I}}), \mathtt{I}(k_1), \dots, \mathtt{I}(k_n), q_1(t_{\mathtt{I}}), \dots, q_m(t_{\mathtt{I}}), r(t_{\mathtt{I}}, \kappa) \to \mathtt{I}(s_{\mathtt{I}})\,,$$
generalized send clauses. (3.2 c)

◁

Now we can define the theory of a protocol.

Definition 3.4. Let $\mathcal{P} = (P, \Phi)$ be a protocol. Let $R_{\mathtt{I}} = \{r_{\mathtt{I}}, r_{\mathtt{I}_{\mathtt{H}}}\} \cup \left\{ r_{\mathtt{I}_{\mathtt{H}_k}} \;\middle|\; k \in K \right\}$ be a set of new binary push symbols, let $q_{\mathtt{I}\downarrow}$ be a new pop symbol. Let $\mathtt{c}_{\mathtt{I}} \in \Gamma$ be an anonymous constant, leading to the unique register sequence $\kappa^{(\mathtt{I})} = (\mathtt{c}_{\mathtt{I}}, \dots, \mathtt{c}_{\mathtt{I}}) \in \{\mathtt{c}_{\mathtt{I}}\}^Z$. The *theory* $\Phi_{\mathcal{P}}$ *of the protocol* $\mathcal{P}$ consists of the clauses (3.3 i a) to (3.3 vi b):

- *(generalized) intruder pop clauses* for each key k

$$\mathtt{I}(x_1), \mathtt{I}(x_2) \to \mathtt{I}(\langle x_1, x_2\rangle)\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\{x\}_k)\,, \tag{3.3 i a}$$

$$\mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\{\!|x|\!\}_k)\,, \tag{3.3 i b}$$

$$\mathtt{I}(x) \to \mathtt{I}(\mathtt{h}(x))\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{h}_k(x))\,, \tag{3.3 i c}$$

$$\mathtt{I}(x) \to \mathtt{I}(\mathtt{H}(x))\,, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{H}_k(x))\,, \tag{3.3 i d}$$

$$\mathtt{I}(x) \to q_{\mathtt{I}\downarrow}(\mathtt{H}_k(x))\,, \tag{3.3 i e}$$

- the *generalized intruder push clauses* for each key k

$$r_{\mathtt{I}}(\langle x_1, x_2\rangle, \kappa_\star) \to r_{\mathtt{I}}(x_1, \kappa^{(\mathtt{I})}) , \qquad \mathtt{I}(k), r_{\mathtt{I}}(\{x\}_k, \kappa_\star) \to r_{\mathtt{I}}(x, \kappa^{(\mathtt{I})}) , \tag{3.3 ii a}$$

$$r_{\mathtt{I}}(\langle x_1, x_2\rangle, \kappa_\star) \to r_{\mathtt{I}}(x_2, \kappa^{(\mathtt{I})}) , \qquad \mathtt{I}(k), r_{\mathtt{I}}(\{|x|\}_{k^{-1}}, \kappa_\star) \to r_{\mathtt{I}}(x, \kappa^{(\mathtt{I})}) , \tag{3.3 ii b}$$

$$r_{\mathtt{I}}(\mathtt{H}(x), \kappa_\star) \to r_{\mathtt{I}}(x, \kappa^{(\mathtt{I})}) , \qquad r_{\mathtt{I}}(\mathtt{H}_k(x), \kappa_\star) \to r_{\mathtt{I}}(x, \kappa^{(\mathtt{I})}) , \tag{3.3 ii c}$$

$$r_{\mathtt{I}}(\mathtt{h}(x), \kappa_\star) \to r_{\mathtt{I}_\mathtt{H}}(x, \kappa^{(\mathtt{I})}) , \qquad r_{\mathtt{I}}(\mathtt{h}_k(x), \kappa_\star) \to r_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa^{(\mathtt{I})}) , \tag{3.3 ii d}$$

$$r_{\mathtt{I}}(\mathtt{H}(x), \kappa_\star) \to r_{\mathtt{I}_\mathtt{H}}(x, \kappa^{(\mathtt{I})}) , \qquad r_{\mathtt{I}}(\mathtt{H}_k(x), \kappa_\star) \to r_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa^{(\mathtt{I})}) , \tag{3.3 ii e}$$

- the *intruder send clauses* for each key k

$$r_{\mathtt{I}}(x, \kappa_\star) \to \mathtt{I}(x) , \tag{3.3 iii a}$$

$$r_{\mathtt{I}_\mathtt{H}}(x, \kappa_\star) \to \mathtt{I}(\mathtt{h}(x)) , \qquad r_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa_\star) \to \mathtt{I}(\mathtt{h}_k(x)) , \tag{3.3 iii b}$$

$$\mathtt{I}(x), r_{\mathtt{I}_\mathtt{H}}(x, \kappa_\star) \to \mathtt{I}(\mathtt{H}(x)) , \qquad \mathtt{I}(x), r_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa_\star) \to \mathtt{I}(\mathtt{H}_k(x)) , \tag{3.3 iii c}$$

- each pop clause in Φ

$$q_1(x_1), \ldots, q_n(x_n) \to q(f(x_1, \ldots, x_n)) , \tag{3.3 iv}$$

- each push clause in Φ

$$q_1(t), \ldots, q_n(t), r(t, \kappa) \to r'(x, \kappa') , \tag{3.3 v}$$

- for each send clause $[q_1(t), \ldots, q_n(t), r(t, \kappa) \to \mathtt{I}(s)] \in \Phi$, each $s' \in \mathrm{sub}(s)$ each $K = \{k_1, \ldots, k_l\}$ with $s \rightrightarrows_K s'$, and each key k, add the following *generalized push clause* (case a) or *generalized send clause(s)* (case b to e) to $\Phi_\mathcal{P}$:

$$\mathtt{I}(k_1), ..., \mathtt{I}(k_l), q_1(t), ..., q_n(t), r(t, \kappa) \to r_{\mathtt{I}}(s', \kappa^{(\mathtt{I})}) \text{ if } s' \in X , \tag{3.3 vi a}$$

$$\mathtt{I}(k_1), ..., \mathtt{I}(k_l), q_1(t), ..., q_n(t), r(t, \kappa) \to \mathtt{I}(s') \quad \text{if } s' \notin X , \tag{3.3 vi b}$$

$$\mathtt{I}(k_1), ..., \mathtt{I}(k_l), q_1(t), ..., q_n(t), r(t, \kappa) \to \mathtt{I}(\mathtt{h}(s'')) \quad \text{if } s' = \mathtt{H}(s'') , \tag{3.3 vi c}$$

$$\mathtt{I}(k_1), ..., \mathtt{I}(k_l), q_1(t), ..., q_n(t), r(t, \kappa) \to \mathtt{I}(\mathtt{h}_k(s'')) \quad \text{if } s' = \mathtt{H}_k(s'') , \tag{3.3 vi d}$$

$$q_{\mathtt{I}\downarrow}(\mathtt{H}_k(s'')), \mathtt{I}(k_1), ..., \mathtt{I}(k_l), q_1(t), ..., q_n(t), r(t, \kappa) \to \mathtt{I}(\mathtt{H}_k(s'')) \quad \text{if } s' = \mathtt{h}_k(s'') . \tag{3.3 vi e}$$

◁

As we need to clarify in which steps of a protocol execution fresh tokens are generated, we define two subsets of the theory of a protocol: The only clauses that contain fresh variables are the clauses of the form (3.3 v); the only prerequisites for applying these clauses may be pop clauses. Therefore, we write $\text{Fresh}(\Phi_{\mathcal{P}})$ for the set of all clauses in $\Phi_{\mathcal{P}}$ which are of the form (3.3 iv) or (3.3 v). In addition, we write $\text{Basic}(\Phi_{\mathcal{P}})$ for the set $\Phi_{\mathcal{P}}$ without the push clauses of the form (3.3 v).

We now formulate the following lemma, which characterizes attacks on protocols using the theory of a protocol:

Lemma 3.5 (Theory of a Protocol)**.** *Let $\mathcal{P}$ be a protocol, let $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$ be an execution scheme, and use $t_{N+1} = \$$. Let σ be a ground substitution.*

We take the pairwise disjoint ground register sequences $\kappa^{(1)}, \ldots, \kappa^{(N)}$ defined in Section 2.2.3 and set C_0 to contain all anonymous constants in these N register sequences. We further assume that there is a set $\{C_0, \ldots, C_N\}$ of pairwise disjoint subsets of Γ.

Then (π, σ) is an attack if and only if there are sets of facts R_n for each $n \in \{1, \ldots, N\}$ with

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash^{C_n}_{\text{Fresh}(\Phi_{\mathcal{P}})} R_n \qquad \text{for all } n \in \{1, \ldots, N\} \text{ and} \tag{3.4 a}$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup R_1 \cup \cdots \cup R_n \vdash_{\text{Basic}(\Phi_{\mathcal{P}})} \mathtt{I}(\sigma(t_{n+1})) \qquad \text{for all } n \in \{0, \ldots, N\}\,. \tag{3.4 b}$$

3.2.1 Proof of Lemma 3.5, Part 1

Proof. Let $\mathcal{P} = (P, \Phi)$ be a protocol, let $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$ be an execution scheme, set $t_{N+1} = \$$. Let σ be a ground substitution.

First, we assume (π, σ) to be an attack. Then let $\{C_1, \ldots, C_N\}$ be the set of subsets of Γ from the definition of an attack (2.21 a). We show that conditions (3.4 a) and (3.4 b) hold for every $n \in \{0, \ldots, N\}$ for a set R_n we will define and for the C_n as defined above.

First consider $n = 0$. We have nothing to show for condition (3.4 a). By (2.21 b) we know that $\mathtt{I}(\dot{\varsigma}) \vdash_{\Phi_{\mathtt{I}}} \mathtt{I}(\sigma(t_1))$, let Π_0^B be a Horn proof of that. Using Lemma 3.1 and the fact that $\dot{\varsigma}$ is a constant, we know that there is no decomposition in Π_0^B, hence, all clauses used in the proof are composition clauses, which are of the form (2.20 e) and (2.20 f). Note that in this step, each fact derived by Clause (2.20 i) can be derived by a clause from (2.20 h), as the intruder has to know the key k to derive $\mathtt{h}_k(x)$ which is a prerequisite for applying (2.20 i). As the composition clauses are elements of $\text{Basic}(\Phi_{\mathcal{P}})$, called intruder pop clauses, we also have $\mathtt{I}(\dot{\varsigma}) \vdash_{\text{Basic}(\Phi_{\mathcal{P}})} \mathtt{I}(\sigma(t_1))$ and Π_0^B is a Horn proof of that, which is our claim for $n = 0$.

Now let $T_1, \ldots, T_N$ be sets of facts such that for all $n \in \{1, \ldots, N\}$ the sequence Π_n^A is a proof of (2.21 a) and Π_n^B is a proof of (2.21 b) with the restriction mentioned in Lemma 3.1.

We define $\widehat{\Pi}_n^A$ as the Horn proof Π_n^A without all the c-facts constructed by send clauses; then let R_n be the set of facts in $\widehat{\Pi}_n^A$ for each $n \in \{1, \ldots, N\}$.

We now take an $n \in \{1, \ldots, N\}$. Let $\Pi_n^A \setminus \widehat{\Pi}_n^A = [a_1, \ldots, a_{i_1}]$ be the sequence of all c-facts constructed by send clauses, i. e., all terms sent to the intruder; and let $\Pi_n^B = [a_{i_1+1}, \ldots, a_{i_2}]$. We will construct a Horn proof $\widehat{\Pi}_n^B = [b_1, \ldots, b_{j_3}]$ of (3.4 b) for that n by the following algorithm, which uses variables $\mathtt{i}$ and $\mathtt{j}$ to iterate over $a_\mathtt{i}$ and $b_\mathtt{j}$:

1. Let $[b_1, \ldots, b_{j_1}]$ be the concatenation of $R_1, \ldots, R_n$. We set $\mathtt{j} = j_1$, and $\mathtt{i} = 1$.
2. Now $a_\mathtt{i}$ is a c-fact, i. e., $a_\mathtt{i} = \mathtt{I}(t)$ for a term t, and we look at the send clause producing $a_\mathtt{i}$:
   a) If it is of the form $[\ldots, r(t', \kappa) \to \mathtt{I}(x)]$ with $x \in X$, we increment $\mathtt{j}$ by two and set $b_{\mathtt{j}-1} = r_\mathtt{I}(t, \kappa^{(\mathtt{I})})$ as well as $b_\mathtt{j} = a_\mathtt{i} = \mathtt{I}(t)$.
   b) If the send clause has the form $[\ldots \to \mathtt{I}(s)]$ for $s \notin X$, but with $s = h(s')$ for $h \in \{\mathtt{H}, \mathtt{H}_k\}$, we increment $\mathtt{j}$ by two and choose
      - $b_{\mathtt{j}-1} = \mathtt{I}(\mathtt{H}(s'))$ and $b_{\mathtt{j}-1} = \mathtt{I}(\mathtt{h}(s'))$, if $h = \mathtt{H}$, or
      - $b_{\mathtt{j}-1} = \mathtt{I}(\mathtt{H}_k(s'))$ and $b_{\mathtt{j}-1} = \mathtt{I}(\mathtt{h}_k(s'))$, if $h = \mathtt{H}_k$.
   c) Otherwise, we increment $\mathtt{j}$ by one and choose $b_\mathtt{j} = a_\mathtt{i}$.
3. Now increment $\mathtt{i}$ and repeat from step 2 on as long as the new $\mathtt{i}$ is less than i_1 or equal, otherwise set $j_2 = \mathtt{j}$ and continue.
4. a) If $a_\mathtt{i}$ is produced by (2.20 i), i. e., $a_\mathtt{i} = \mathtt{I}(\mathtt{H}_k(t))$ we increment $\mathtt{j}$ by two and set $b_{\mathtt{j}-1} = q_{\mathtt{I}\downarrow}(\mathtt{H}_k(t))$ as well as $b_\mathtt{j} = a_\mathtt{i}$.
   b) If $a_\mathtt{i}$ is produced by another composition clause, we increment $\mathtt{j}$ by one and set $b_\mathtt{j} = a_\mathtt{i}$.
   c) If $a_\mathtt{i} = \mathtt{I}(t)$ is a c-fact produced by a decomposition clause $\varphi \in \Phi_\mathtt{I}$ from (2.20 a) to (2.20 c) with an anonymous ground substitution σ, formally we have to differentiate for the different types of decomposition clauses. As example we will consider $\varphi = [\mathtt{I}(\langle x_1, x_2 \rangle) \to \mathtt{I}(x_1)]$, with $t = \sigma(x_1)$.
      i. If we find a fact of the form $r_\mathtt{I}(\langle t, t' \rangle, \kappa^{(\mathtt{I})})$ in $\{b_1, \ldots, b_\mathtt{j}\}$, we increment $\mathtt{j}$ by two and set $b_{\mathtt{j}-1} = r_\mathtt{I}(t, \kappa^{(\mathtt{I})})$ and $b_\mathtt{j} = a_\mathtt{i}$.

ii. If we do not find such a fact, there is a clause ψ in $\Phi_{\mathcal{P}}$ of the form

$$\psi = \left[\mathtt{I}(k_1), \ldots, \mathtt{I}(k_l), q_1(t_1), \ldots, q_m(t_1), r(t_1, \kappa) \to \begin{cases} \mathtt{I}(t_2) \text{ or} \\ r_{\mathtt{I}}(t_2, \kappa^{(\mathtt{I})}) \end{cases} \right]. \quad (3.5)$$

and a substitution σ' with $t = \sigma'(t_2)$. In the case $r_{\mathtt{I}}$, we increment $\mathtt{j}$ by two and set $b_{\mathtt{j}-1} = r_{\mathtt{I}}(t, \kappa^{(\mathtt{I})})$ and $b_{\mathtt{j}} = a_{\mathtt{i}}$. Otherwise for $\mathtt{I}$, we increment $\mathtt{j}$ by one and set $b_{\mathtt{j}} = a_{\mathtt{i}}$.

We proceed analogously for the decomposition clauses (2.20 a) to (2.20 c).

d) If $a_{\mathtt{i}}$ is produced by (2.20 d), we simply increment $\mathtt{j}$ by one and set $b_{\mathtt{j}} = a_{\mathtt{i}}$.

5. Now increment $\mathtt{i}$ and repeat from step 4 on as long as the new $\mathtt{i}$ is less than i_2 or equal, otherwise set $j_3 = \mathtt{j}$.

This algorithm produces a Horn proof of (3.4 b): For each $a_{\mathtt{i}}$ with $a_{\mathtt{i}} = \mathtt{I}(t)$ we have a fact $b_{\mathtt{j}}$ with $b_{\mathtt{j}} = a_{\mathtt{i}}$. We will show that the facts $b_{\mathtt{j}}$ are justified, i. e., the definition of a Horn proof holds for the sequence $[b_1, \ldots, b_{j_3}]$.

Together with the fact that the algorithm terminates because he is iterating over a finite sequence this proves our claim.

Let $j < j_3$, and assume as an inductive hypothesis that the facts $b_1, \ldots, b_j$ conform to the definition of a Horn proof. For $j \leq j_1$, the fact b_j is an a-fact, hence, we have nothing to show. For $j > j_1$ we will show that the next steps of the algorithm are correct: As j is incremented within this step, we have to show that the fact b_j (and possibly also b_{j-1}) conform to the definition of a Horn proof.

2. In this case, we have $a_i \in \Pi_n^A \setminus \widehat{\Pi}_n^A$.

a) The fact b_{j-1} is justified as by construction of the theory of the protocol (3.3 vi a) we have $[q_1(t'), \ldots, r(t'', \kappa) \to r_{\mathtt{I}}(x, \kappa^{(\mathtt{I})})] \in \mathrm{Basic}(\Phi_{\mathcal{P}})$. The fact b_j is derived by the clause $[r_{\mathtt{I}}(x, \kappa_\star) \to \mathtt{I}(x)]$ (3.3 iii a).

b) This step is also justified because for $s \notin X$, the original send clause in Φ is copied to the theory of the protocol $\mathrm{Fresh}(\Phi_{\mathcal{P}})$, see (3.3 vi b), and there are additional clauses (3.3 vi c) and (3.3 vi d).

c) Again, this step is justified because for $s \notin X$, the original send clause in Φ is copied (3.3 vi b).

4. In this case, we have $a_i \in \Pi_n^B$.

   a) We know that $a_{\mathtt{i}} = \mathtt{I}(\mathtt{H}_k(t))$. As clause (2.20 i) was applied to derive that fact, we already have the fact $\mathtt{I}(x)$ and thus can derive $b_{\mathtt{j}-1} = q_{\mathtt{I}\downarrow}(\mathtt{H}_k(t))$ by (3.3 i e). Then $b_{\mathtt{j}}$ can be derived by a clause φ_2 depending on the clause φ_1 which was used to derive the fact $\mathtt{I}(\mathtt{h}_k(t))$ (which is used as a prerequisite to apply (2.20 i)):

original clause φ_1	corresponding clause φ_2
(3.3 i c)	(3.3 i d)
(3.3 iii b)	(3.3 iii c)
(3.3 vi b)	(3.3 vi e)
(3.3 vi d)	(3.3 vi b)

   Each of the clauses φ_2 has the same or fewer prerequisites that the corresponding clause φ_1, except for $b_{\mathtt{j}-1}$ which we derive as above.

   b) We know that a_i is either an a-fact from T_n or a_i is a c-fact constructed by a composition clause. The case $a_i \in T_n$ implies $b_j = a_i \in \{b_1, \ldots, b_{j_2}\}$, and hence, b_j is justified. If on the other hand a_i is produced by a composition clause, b_j is justified by the same composition clause, as the composition clauses are elements of Basic($\Phi_{\mathcal{P}}$), see (3.3 i a) and (3.3 i b), and for every fact $\mathtt{I}(t) \in \{a_1, \ldots, a_{i-1}\}$ we have $\mathtt{I}(t) \in \{b_1, \ldots, b_{j-1}\}$.

   c) i. The justification for b_{j-1} is the corresponding (generalized) intruder push clause, see (3.3 ii b) and (3.3 ii c); the fact b_j is again derived by the clause $[r_{\mathtt{I}}(x, \kappa_\star) \to \mathtt{I}(x)]$ (3.3 iii a).

   ii. This step is the most difficult one. We know that in the assumed Horn proof Π_n^B the fact $a_i = \mathtt{I}(t)$ was produced by a decomposition clause φ, we again assume that $\varphi = [\mathtt{I}(\langle x, y\rangle) \to \mathtt{I}(x)]$. We know $a_{i'} = \mathtt{I}(\langle t, t'\rangle)$ for an index $i' < i$. But as we are in case ii., there is no fact $r_{\mathtt{I}}(\langle t, t'\rangle, \kappa^{(\mathtt{I})}) \in \{b_1, \ldots, b_{j-2}\}$. By algorithm, there has to be a fact $b_{j'} = \mathtt{I}(\langle t, t'\rangle)$ with $j' < j - 1$.

   Now we take the justification for the fact $b_{j'}$. As in Π_n^B the restrictions of Lemma 3.1 hold, $a_{i'}$ is not produced by a composition clause. By inductive hypothesis $b_{j'}$ has to be generated by a (generalized) send clause

$$\hat{\psi} = \left[\mathtt{I}(k_1), \ldots, \mathtt{I}(k_l), q_1(t_1), \ldots, q_m(t_1), r(t_1, \kappa) \to \begin{cases} \mathtt{I}(t_2) \text{ or} \\ r_{\mathtt{I}}(t_2, \kappa^{(\mathtt{I})}) \end{cases}\right]. \quad (3.6)$$

The case $r_{\mathtt{I}}$ is not possible, hence, we have a clause with $\mathtt{I}(t_2)$ on the right hand side, and a substitution σ with $\sigma(t_2) = \langle t, t' \rangle$. By (3.3 vi a) we know that $t_2 \notin X$, and because $t_2 \notin Y$ (in that case t_2 could not be mapped to a term containing a pairing symbol) we know that $t_2 = \langle t_{2a}, t_{2b} \rangle$ for terms t_{2a}, t_{2b}. Now by definition of the theory of the protocol, there is a clause

$$\psi = \left[\mathtt{I}(k_1), \ldots, \mathtt{I}(k_l), q_1(t_1), \ldots, q_m(t_1), r(t_1, \kappa) \to \begin{cases} or\ \mathtt{I}(t_{2a}) \\ r_{\mathtt{I}}(t_{2a}, \kappa^{(\mathtt{I})}) \end{cases} \right]. \quad (3.7)$$

For the decryption clauses amongst the decomposition clauses, we may have to add another key on the left hand side (i. e., $\mathtt{I}(k_{l+1})$), but this key has already been used in Π_n^A to construct a_i.

The clause ψ can now be used in step 4. b) ii. of the algorithm.

d) For (2.20 d), we have $a_{\mathtt{i}} \in \{\mathtt{I}(\mathtt{h}(t)), \mathtt{I}(\mathtt{h}_k(t))\}$ for some term t. As clause (2.20 d) was applied in the original proof, we already have a fact $b' = \mathtt{I}(\mathtt{H}(t))$ or $b' = \mathtt{I}(\mathtt{H}_k(t))$ in $\widehat{\Pi}_n^B$. Now take a look at the clause φ_1 justifying b—in each case, there is a corresponding clause φ_2 which justifies $a_{\mathtt{i}}$ and which has the same or fewer prerequisites:

original clause φ_1	corresponding clause φ_2
(3.3 i d)	(3.3 i c)
(3.3 iii c)	(3.3 iii b)
(3.3 vi b)	(3.3 vi c) or (3.3 vi d)
(3.3 vi e)	(3.3 vi a)

We have proven that for every attack (π, σ), the characterization in Lemma 3.5 holds.

3.2.2 Proof of Lemma 3.5, Part 2

We will now show the other direction of the lemma. Assume that for all $n \in \{0, \ldots, N\}$ the two conditions (3.4 a) and (3.4 b) hold. We now have to show that (π, σ) is an attack, i. e., we find sets T_n of facts of the form $\mathtt{I}(t)$ such that the following conditions hold:

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash_{\Phi}^{C_n} T_n \qquad \text{for all } n \in \{1, \ldots, N\} \text{ and} \quad (3.8\,a)$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup T_1 \cup \cdots \cup T_n \vdash_{\Phi_{\mathtt{I}}} \mathtt{I}(\sigma(t_{n+1})) \qquad \text{for all } n \in \{0, \ldots, N\}. \quad (3.8\,b)$$

For $n \in \{1, \ldots, N\}$, let Π_n^A be a Horn proof of (3.4 a), and let Π_n^B be a proof of (3.4 b) for $n \in \{0, \ldots, N\}$. We will construct Horn proofs $\widehat{\Pi}_n^A$ and $\widehat{\Pi}_n^B$ for (3.8 a) and (3.8 b) by transforming Π_n^A and Π_n^B by a method that preserves the semantics of each fact in these proofs.

Let $\Pi_n^{\text{push}}, \Pi_n^{\text{send}}, \Pi_n^{\text{intruder}}$ be empty sequences for each $n \in \{0, \ldots, N\}$, and let Π^{pop} also be an empty sequence. We will now iterate over the facts in Π_n^A and Π_n^B; in every iteration we will append facts to the sets we just defined, e. g., Π^{pop}.

For every $n \in \{1, \ldots, N\}$ and each c-fact a in the concatenation $\Pi_n^A \cdot \Pi_n^B$, we consider the following cases:

i. a) If a was produced by an intruder pop clause from (3.3 i a) to (3.3 i d), we append a to Π_n^{intruder}.

   b) If a was produced by the intruder pop clause (3.3 i e), we do nothing.

ii. a) If a was produced by a (generalized) intruder push clause (3.3 ii a) to (3.3 ii c), we know $a = r_{\mathtt{I}}(t, \kappa^{(\mathtt{I})})$ for a ground term t, and we append $\mathtt{I}(t)$ to Π_n^{intruder}.

   b) If a was produced by an intruder push clause (3.3 ii d) or (3.3 ii e), we do nothing.

iii. a) If a was produced by the intruder send clause (3.3 iii a), we have nothing to do.

   b) If a was produced by an intruder send clause from (3.3 iii b) or (3.3 iii c), we append a to Π_n^{intruder}.

iv. If a was produced by a pop clause (3.3 iv), we append a to Π^{pop}.

v. If a was produced by a push clause from Φ (3.3 v), we append a to Π_n^{push}.

vi. a) If a was produced by a (generalized) send or push clause from (3.3 vi a) or (3.3 vi b), we have $a = \mathtt{I}(\hat{t})$ or $a = r_{\mathtt{I}}(\hat{t}, \kappa^{(\mathtt{I})})$ for a term $\hat{t}$. We now take that clause $\psi \in \Phi_{\mathcal{P}}$ and the corresponding substitution σ with $\sigma(t) = \hat{t}$, and know by definition of the theory of the protocol that there is a clause $\varphi \in \Phi$, such that we have:

$$\psi = \left[\mathtt{I}(k_1), \ldots, \mathtt{I}(k_l), q_1(t'), \ldots, q_{n'}(t'), r(t', \kappa) \to \left\{ \begin{array}{l} \mathtt{I}(t) \text{ or} \\ r_{\mathtt{I}}(t, \kappa^{(\mathtt{I})}) \end{array} \right.\right], \tag{3.9}$$

$$\varphi = [q_1(t'), \ldots, q_{n'}(t'), r(t', \kappa) \to \mathtt{I}(s)] \tag{3.10}$$

$$\text{with } s \rightrightarrows_{\{k_1, \ldots, k_l\}} t\,. \tag{3.11}$$

We append $\mathtt{I}(\sigma(s))$ to Π_n^{send}.

By $s \rightrightarrows_{\{k_1,\dots,k_l\}} t$ we know there is a sequence $[s_1,\dots,s_m]$ with $s = s_1$, $s_m = t$ in which $s_{j+1} \in \mathrm{sub}(s_j)$ and $\mathrm{depth}(s_{j+1}) = \mathrm{depth}(s_j) - 1$ for all $j \in \{1,\dots,m-1\}$. Let $\sigma(s_j) = t_j$, we then append each $\mathtt{I}(t_j)$ to Π_n^{intruder}.

b) If a was produced by a (generalized) send clause from (3.3 vi c) to (3.3 vi e), we have $a = \mathtt{I}(h(s''))$ for some hash symbol h. Now we first look at the corresponding clause (3.3 vi b) which produces a fact $a' = \mathtt{I}(s')$ where $s' = h'(s'')$ for a hash symbol h' that corresponds to h.

First, we apply case vi. a), i. e., append one fact to Π_n^{send} and possibly a sequence of facts including a' to Π_n^{intruder}. Then we append a to Π_n^{intruder}.

This method terminates as we are iterating over finite sequences. Now we set $T_n = \Pi_n^{\text{send}}$ and define the following Horn proofs for all $n \in \{0,\dots,N\}$:

$$\widehat{\Pi}_n^A = \Pi^{\text{pop}} \cdot \Pi_n^{\text{push}} \cdot \Pi_n^{\text{send}} \text{ and} \tag{3.12}$$

$$\widehat{\Pi}_n^B = \Pi^{\text{pop}} \cdot \Pi_1^{\text{send}} \cdot \dots \cdot \Pi_n^{\text{send}} \cdot \Pi_n^{\text{intruder}} \,. \tag{3.13}$$

Now we know that $\widehat{\Pi}_n^A$ is a Horn proof of (3.8 a) and $\widehat{\Pi}_n^B$ is one of (3.8 b). First take a look at the facts in $\widehat{\Pi}_n^A$:

- The elements in Π^{pop} are independent of all other facts, they are justified by the same pop clauses in Φ as in $\Phi_{\mathcal{P}}$.
- All facts derived by a push clauses (3.3 v), i. e., a clause copied from Φ to $\Phi_{\mathcal{P}}$, can be derived by exactly this clause in Φ because we keep the order among the push facts, and because we put all facts derived by pop clauses in Π^{pop}.
- The applications of send clauses in Φ only depend on facts obtained by pop and push clauses of Φ, and so the facts in Π_i^{send} are justified.

Finally, as $T_n = \Pi_n^{\text{send}} \subseteq \widehat{\Pi}_n^A$, this is a Horn proof for T_n.

Now take a look at the facts in $\widehat{\Pi}_n^B$. All facts in Π^{pop} are justified as above; whereas all the facts in Π_i^{send} for an index $i \leq n$ are a-facts. But we have to look at the facts in Π_n^{intruder}.

i. The facts added in step i. can be derived by the composition clauses in the intruder theory.

ii. The facts appended in step ii. are justified as we transform all facts of the form $r_{\mathtt{I}}(t, \kappa^{(\mathtt{I})})$ to $\mathtt{I}(t)$, such that for every application of an intruder push clause on push facts there is an equivalent application of decomposition clauses.

More precisely, the prerequisites for an application of a clause φ from (3.3 ii a) to (3.3 ii c) are transformed from $r_{\mathtt{I}}(t, \kappa^{(\mathtt{I})})$ to $\mathtt{I}(t)$ and appear in $\Pi_n^{\text{send}} = T_n$ or earlier in Π_n^{intruder}, hence, in $\widehat{\Pi}_n^B$ we can apply a decomposition clause corresponding to φ from the intruder theory.

iii. In step iii., each fact a that we add depends on a push fact a' with symbol $r_{\mathtt{I}_{\mathtt{H}}}$ or $r_{\mathtt{I}_{\mathtt{H}_k}}$, i. e., a' can only have been derived by (3.3 ii d) or (3.3 ii e) in the original Horn proof Π_n^B.

- If a' was derived by (3.3 ii d) and a was derived by (3.3 iii b), the fact a is simply an a-fact. The same applies for a' being derived by (3.3 ii e) and a being derived by (3.3 iii c).
- If a' was derived by (3.3 ii e) and a was derived by (3.3 iii b), the fact a can be derived by the decomposition clause (2.20 d).
- If a' was derived by (3.3 ii d) and a was derived by (3.3 iii c), the fact a can be derived by the composition clause (2.20 i).

vi. a) In step vi. we first have to take a look at the facts $\mathtt{I}(k)$ for keys k. In the original Horn proof Π_n^B, each fact $\mathtt{I}(k)$ has to be known prior to the application of this generalized push or send clause ψ. The algorithm puts this fact $\mathtt{I}(k)$ in either Π_n^{send} or Π_n^{intruder}, in the latter case $\mathtt{I}(k)$ prior to the application of the clause ψ. Therefore, we can just assume that the necessary keys are known.

The facts added in step vi. are justified as we added the result of a send clause to Π_n^{send}, hence, we have a copy in T_n and can decompose this fact using the decomposition clauses of the intruder theory and the relevant keys.

b) The application of the clauses from (3.3 vi c) to (3.3 vi e) in the original Horn proof Π_n^B can be reduced to applying case a) plus applying either the decomposition clause (2.20 d) or the composition clause (2.20 i). In the latter case, the fact $q_{\mathtt{I}\downarrow}(\mathtt{H}_k(s''))$ guarantees $\mathtt{I}(s'')$.

We now have Horn proofs $\widehat{\Pi}_n^A$ and $\widehat{\Pi}_n^B$ for (3.8 a) and (3.8 b), which concludes our proof. □

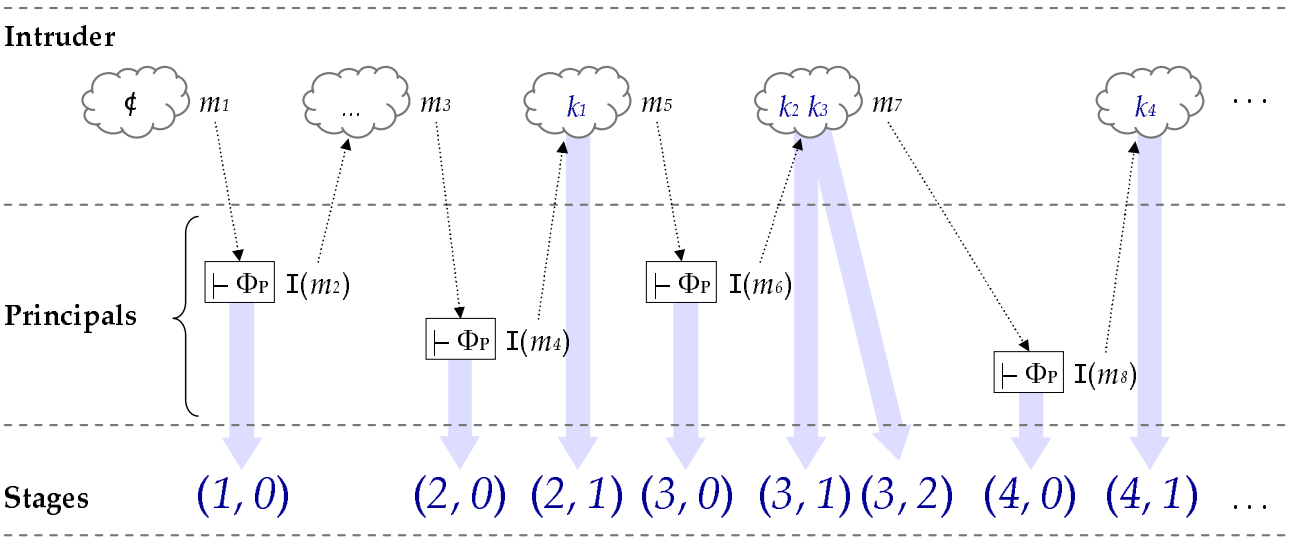

Figure 3.2: Example for the stages of a protocol with protocol steps 1 to 4

3.3 Stage Theory

The characterization of an attack presented in Lemma 3.5 will be transformed anew in this section. We start with an execution scheme of a protocol. First, each of the protocol steps resembles a major stage in the execution of the protocol. For each of these major stages it is now possible to state which keys are added to the intruder's knowledge during this protocol step, allowing us to divide the major stage into multiple stages. This is illustrated in Figure 3.2.

As a consequence we can merge the $(2 \cdot N + 1)$ Horn proofs necessary to show that the definition of an attack or the characterization in Lemma 3.5 holds, resulting in only *one* Horn proof. To do so annotate the predicate symbols with the stage they are used in – see Figure 3.3.

Formally we will annotate the predicate symbols in the Horn clauses with special indices: These indices contain the current protocol step. But during each step, the intruder may be able to gain knowledge of several keys – therefore, we will use indices of the form (n, m) where n is the current protocol step and m is the number of keys obtained so far during that step. This gives us a partitioning of the protocol run into several stages as shown in the following example.

Example 3.6. A possible run of a protocol as illustrated in Figure 3.2 could be as follows:

$(0, 0)$ The intruder composes and sends his initial message m_1 to start the protocol.

$(1,0)$ The principal of the first protocol step receives m_1, processes it and sends a message m_2. The intruder decomposes m_2 but does not learn any keys from it. He composes a message m_3 and sends it.

$(2,0)$ Another principal receives m_3 and sends m_4, containing a key k_1 and the subterm $\{k_3\}_{k_2}$. The intruder extracts k_1.

$(2,1)$ The intruder cannot extract k_3 as he does not know k_2 yet, but he composes and sends a message m_5.

$(3,0)$ After receiving m_5, one principal sends m_6 which contains the key k_2. The intruder is able to decompose m_6 and gain knowledge of k_2.

$(3,1)$ With the help of k_2 the intruder is able to decompose m_4 so that he can extract k_3.

$(3,2)$ Now the intruder composes a message m_7 that may, for example, contain terms encrypted by k_1 or k_2.

$(4,0)$ The next protocol step is executed – and so on. ◁

Note that in the example and in the figure each of our principals is only sending one message – as stated earlier, each principal could just as well send a whole set of messages.

Remark 3.7. We must remark that up to this point, it does not seem necessary to divide a protocol step into multiple stages. But the second component of the indices is necessary to prevent cycles in Horn proofs. Even though this cannot happen in the Horn proofs we used so far because they are just sequences of facts, we will later see that in the ADAG there may be cyclic situations if we do not prevent this:

For example, consider a protocol step in which the intruder learns two keys k_1, k_2. We assume that there is a message $m = \langle\{k_2\}_{k_1}, \{k_1\}_{k_2}\rangle$. Now in a Horn proof as defined in Definition 2.6 it is clear that the intruder can only learn k_1 and k_2 if he knew one of these keys before receiving m.

But in the ADAG we loose that linearity and may get cycles, i. e., the intruder would be able to derive k_1 from the message m by using k_2, which he derived from that same message m by using k_1. By introducing stages we ensure that if k_1 is derived in stage e_1, this key can only be used in stages $e_2 > e_1$, avoiding the occurrence of cycles. ◁

Now we are ready to give the formal definition of stages and stage mappings.

Definition 3.8. Let $\mathcal{P}$ be a protocol with an execution scheme π with N protocol steps, and let K be the set of keys in Σ. Let $E = \{0, \dots, N\} \times \{0, \dots, |K|\}$ be the *set of stages* of $\mathcal{P}$, let $<$ be the lexicographical ordering on the set of stages E, and let $\ll$ be the ordering on the first component (and define $\leq$, $\geq$, $>$, and $\gg$ in the natural way):

$$(n, m) < (n', m') \qquad \text{if } n < n' \text{ or } (n = n' \text{ and } m < m') \,, \tag{3.14}$$

$$(n, m) \ll (n', m') \qquad \text{if } n < n' \,, \tag{3.15}$$

$$n \ll (n', m') \qquad \text{if } n < n' \,. \tag{3.16}$$

We will denote $\{ e' \in E \mid e' \leq e \}$ by $E_{\leq e}$. For a set of predicate symbols R, let $R^{(E)}$ be the following set of predicate symbols (maintaining their arity):

$$R^{(E)} = \{ r^e \mid r \in R, e \in E \} \,. \tag{3.17}$$

A partial function $f \colon K \dashrightarrow E$ is called *stage mapping* for $\mathcal{P}$ if

$$f(k) \in \{ (n, m) \in E \mid n \neq 0, m \neq 0 \} \tag{3.18}$$

for all k. We will write K_f for the set of keys for which f is defined. Additionally, for each $e \in E$ we will use $K(e) = \{ k \in K \mid f(k) \leq e \}$. ◁

The intended semantic of a stage mapping f is that for a key k with $f(k) = e$, this key is derived in a stage $e_k < e$ and thus can be used from stage e onwards to derive other messages or keys. This will be used during the definition of the stage theory.

From now on we will use $\widetilde{Q}$ as pop symbols and $\widetilde{R}$ as push symbols:

$$\widetilde{\mathtt{I}} = \{\mathtt{I}\}^{(E)} \,, \tag{3.19 a}$$

$$\widetilde{\mathtt{I}}^{+} = \{\mathtt{I}, q_{\mathtt{I}\downarrow}\}^{(E)} \,. \tag{3.19 b}$$

$$\widetilde{Q} = Q \cup \widetilde{\mathtt{I}}^{+} \,, \tag{3.19 c}$$

$$\widetilde{R} = \left(R \cup \{ r_{\mathtt{I}}, r_{\mathtt{I}_{\mathtt{H}}} \} \cup \left\{ r_{\mathtt{I}_{\mathtt{H}_k}} \;\middle|\; k \in K \right\} \right)^{(E)} \,, \tag{3.19 d}$$

Thus, we have unary predicate symbols like $\mathtt{I}^{(n,m)}$ and binary predicate symbols like $r^{(n,m)}$. By abuse of notation we will write $\mathtt{I}^{(n)}$ if we have $\mathtt{I}^{(n,m)}$ for some m.

Definition 3.9. Let $\mathcal{P} = (P, \Phi)$ be a protocol. Let E be the set of stages of $\mathcal{P}$, $f \colon K \dashrightarrow E$ be a stage mapping for $\mathcal{P}$. The *stage theory Φ_f of the protocol $\mathcal{P}$ and the stage mapping f* is a

selecting theory over $(\widetilde{Q}, \widetilde{R})$ consisting of the following clauses:

- intruder pop clauses for each $e \in E$, each key $k \in K(e)$, and each $e', e'' \in E_{\leq e}$

$$\mathtt{I}^{e'}(x_1), \mathtt{I}^{e''}(x_2) \to \mathtt{I}^{e}(\langle x_1, x_2 \rangle)\,, \qquad \mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\{x\}_k)\,, \tag{3.20 i a}$$

$$\mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\{\!|x|\!\}_k)\,, \tag{3.20 i b}$$

$$\mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\mathtt{h}(x))\,, \qquad \mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\mathtt{h}_k(x))\,, \tag{3.20 i c}$$

$$\mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\mathtt{H}(x))\,, \qquad \mathtt{I}^{e'}(x) \to \mathtt{I}^{e}(\mathtt{H}_k(x))\,, \tag{3.20 i d}$$

$$\mathtt{I}^{e'}(x) \to q^{e}_{\mathtt{I}\downarrow}(\mathtt{H}_k(x))\,, \tag{3.20 i e}$$

- intruder push clauses for each $e \in E$, each key $k \in K(e)$, and each $e' \in E_{\leq e}$

$$r^{e'}_{\mathtt{I}}(\langle x_1, x_2 \rangle, \kappa_\star) \to r^{e}_{\mathtt{I}}(x_1, \kappa^{(\mathtt{I})})\,, \qquad r^{e'}_{\mathtt{I}}(\{x\}_k, \kappa_\star) \to r^{e}_{\mathtt{I}}(x, \kappa^{(\mathtt{I})})\,, \tag{3.20 ii a}$$

$$r^{e'}_{\mathtt{I}}(\langle x_1, x_2 \rangle, \kappa_\star) \to r^{e}_{\mathtt{I}}(x_2, \kappa^{(\mathtt{I})})\,, \qquad r^{e'}_{\mathtt{I}}(\{\!|x|\!\}_{k^{-1}}, \kappa_\star) \to r^{e}_{\mathtt{I}}(x, \kappa^{(\mathtt{I})})\,, \tag{3.20 ii b}$$

$$r^{e'}_{\mathtt{I}}(\mathtt{H}(x), \kappa_\star) \to r^{e}_{\mathtt{I}}(x, \kappa^{(\mathtt{I})})\,, \qquad r^{e'}_{\mathtt{I}}(\mathtt{H}_k(x), \kappa_\star) \to r^{e}_{\mathtt{I}}(x, \kappa^{(\mathtt{I})})\,, \tag{3.20 ii c}$$

$$r^{e'}_{\mathtt{I}}(\mathtt{h}(x), \kappa_\star) \to r^{e}_{\mathtt{I}_\mathtt{H}}(x, \kappa^{(\mathtt{I})})\,, \qquad r^{e'}_{\mathtt{I}}(\mathtt{h}_k(x), \kappa_\star) \to r^{e}_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa^{(\mathtt{I})})\,, \tag{3.20 ii d}$$

$$r^{e'}_{\mathtt{I}}(\mathtt{H}(x), \kappa_\star) \to r^{e}_{\mathtt{I}_\mathtt{H}}(x, \kappa^{(\mathtt{I})})\,, \qquad r^{e'}_{\mathtt{I}}(\mathtt{H}_k(x), \kappa_\star) \to r^{e}_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa^{(\mathtt{I})})\,, \tag{3.20 ii e}$$

- intruder send clauses for each $e \in E$, each key k, and each $e', e'' \in E_{\leq e}$

$$r^{e'}_{\mathtt{I}}(x, \kappa_\star) \to \mathtt{I}^{e}(x)\,, \tag{3.20 iii a}$$

$$r^{e'}_{\mathtt{I}_\mathtt{H}}(x, \kappa_\star) \to \mathtt{I}^{e}(\mathtt{h}(x))\,, \qquad r^{e'}_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa_\star) \to \mathtt{I}^{e}(\mathtt{h}_k(x))\,, \tag{3.20 iii b}$$

$$\mathtt{I}^{e''}(x), r^{e'}_{\mathtt{I}_\mathtt{H}}(x, \kappa_\star) \to \mathtt{I}^{e}(\mathtt{H}(x))\,, \qquad \mathtt{I}^{e''}(x), r^{e'}_{\mathtt{I}_{\mathtt{H}_k}}(x, \kappa_\star) \to \mathtt{I}^{e}(\mathtt{H}_k(x))\,, \tag{3.20 iii c}$$

- each pop clause in Φ, which is, by definition, also in $\Phi_{\mathcal{P}}$

$$q_1(x_1), \ldots, q_n(x_n) \to q(f(x_1, \ldots, x_n))\,, \tag{3.20 iv}$$

- for each push clause $[q_1(t), \ldots, r(t, \kappa) \to r'(x, \kappa')] \in \Phi$, which, by definition, is also in $\Phi_{\mathcal{P}}$, and for each $e = (i, 0) \in E$

$$q_1(t), \ldots, r^{e}(t, \kappa) \to r'^{\,e}(x, \kappa')\,, \tag{3.20 v}$$

- for each (generalized) push clause $[\mathtt{I}(k_1), \ldots, \mathtt{I}(k_l), q_1(t), \ldots, r(t, \kappa) \to r(x, \kappa^{(\mathtt{I})})] \in \Phi_{\mathcal{P}}$,

and accordingly, for each (generalized) send clause $[\ldots \to \mathtt{I}(s)] \in \Phi_{\mathcal{P}}$, for each $e \in E$, $e' = (n, 0) \in E_{\leq e}$, $e'' \in E_{\leq e}$, and $\{k_1, \ldots, k_l\} \subseteq K(e)$

$$\left.\begin{array}{l} \\ q^{e''}_{\mathtt{I}\downarrow}(\mathtt{H}_k(s'')), \end{array}\right\} q_1(t), \ldots, q_n(t), r^{e'}(t, \kappa) \to \begin{cases} r^e(x, \kappa^{(\mathtt{I})}) \text{ or} \\ \mathtt{I}^e(s)\ . \end{cases} \tag{3.20 vi}$$

◁

As we later need a unique stage e_t for each fact $I^{e_t}(t)$ for a term t, we introduce the notion of normal Horn proofs. Afterwards we can formulate the lemma that links the description of an attack to the stage theory.

Definition 3.10. A Horn proof Π is called *normal* if for each term t there is at most one stage $e \in E$ with $\mathtt{I}^e(t) \in \Pi$. ◁

Lemma 3.11 (Stage Theory)**.** *Let $\mathcal{P} = (P, \Phi)$ be a protocol. Let $\pi = [t_n \to r_n(s_n)]_{n=1}^N$ be an execution scheme, set $t_{N+1} = \$$. Take the pairwise disjoint ground register sequences $\kappa^{(1)}, \ldots, \kappa^{(N)}$ and again set C_0 to contain all anonymous constants in these N register sequences. Let σ be a ground substitution, $C \subset \Gamma$ a set of anonymous constants disjoint with C_0.*

Then (π, σ) is an attack if and only if there exists a stage mapping f (which can be efficiently constructed) such that there is a normal Horn proof of the following statement for stages $e_n \ll n+1$ and $e_k < f(k)$:

$$\left\{\mathtt{I}^{(0,0)}(\dot{\varsigma})\right\} \cup \left\{r^{(n,0)}(\sigma(s_n), \kappa^{(n)}) \;\middle|\; n \in \{1, \ldots, N\}\right\} \vdash^{C}_{\Phi_f} \left\{\mathtt{I}^{e_n}(\sigma(t_{n+1})) \;\middle|\; n \in \{0, \ldots, N\}\right\} \cup \left\{\mathtt{I}^{e_k}(k) \;\middle|\; k \in \mathrm{dom}(f)\right\} \tag{3.21}$$

3.3.1 Proof of Lemma 3.11, Part 1

Proof. In the first part of the proof, we assume that there is an attack (π, σ) on the protocol, and we construct a stage mapping f. To show that the condition holds we construct a Horn proof of statement (3.21), consisting of the concatenation of modified facts taken from Horn proofs used in Lemma 3.5, i. e., we annotate most of the facts with stages. Figure 3.3 illustrates this transformation.

Let $\mathcal{P} = (P, \Phi)$ be a protocol. Let $\pi = [t_n \to r_n(s_n)]_{n=1}^N$ be an execution scheme, set $t_{N+1} = \$$. Take the pairwise disjoint ground register sequences $\kappa^{(1)}, \ldots, \kappa^{(N)}$ and again set C_0 to contain all anonymous constants in these N register sequences. Let σ be a ground

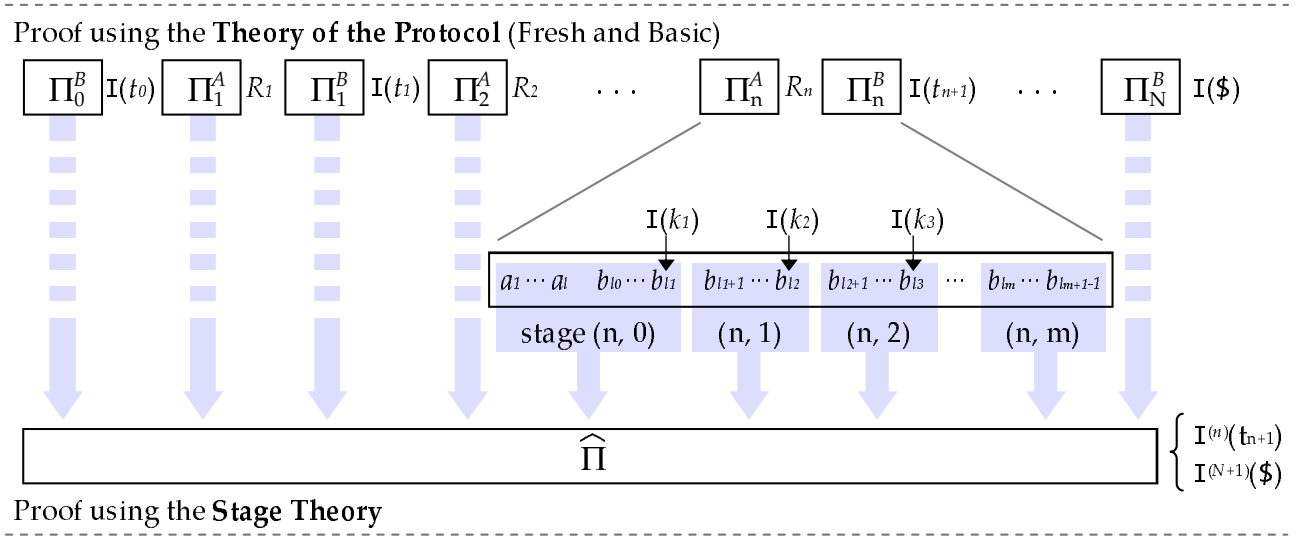

Figure 3.3: Transformation of multiple proofs with respect to the theory of a protocol $\Phi_{\mathcal{P}}$ into a single proof with respect to a stage theory Φ_f

substitution such that (π, σ) is an attack on $\mathcal{P}$. By Lemma 3.5 we know there is a set $\{C_0, \ldots, C_N\}$ of pairwise disjoint subsets of Γ such that:

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash^{C_n}_{\mathrm{Fresh}(\Phi_{\mathcal{P}})} R_n \qquad \text{for all } n \in \{1, \ldots, N\} \text{ and} \tag{3.22 a}$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup R_1 \cup \cdots \cup R_n \vdash_{\mathrm{Basic}(\Phi_{\mathcal{P}})} \mathtt{I}(\sigma(t_{n+1})) \qquad \text{for all } n \in \{0, \ldots, N\}\,. \tag{3.22 b}$$

Let Π^A_n be a Horn proof of (3.22 a) for $n \in \{1, \ldots, N\}$ (and the empty sequence for $n = 0$, respectively). Let Π^B_n be a proof of (3.22 b) for $n \in \{0, \ldots, N\}$.

Let K_n be the sequence of keys added to the intruder's knowledge during step $n \in \{1, \ldots, N\}$, i. e., not known before. This means K_n is a sequence such that for each $m \leq |K_n|$ we have $\mathtt{I}(K_n[m]) \in \Pi^B_n$, but also $\mathtt{I}(K_n[m]) \notin \Pi^B_i$ for all $i < n$, with the same ordering of the keys in K_n as in Π^B_n. We define $M = \max\{\, |K_n| \mid n \in \{1, \ldots, N\}\}$. Now we can construct a stage mapping f as

$$f\colon K \dashrightarrow \{0, \ldots, N\} \times \{0, \ldots, M\} \text{ with } f(k) = \begin{cases} (n, m) & \text{if } k = K_n[m]\,, \\ \bot & \text{if } k \notin \bigcup_{i=n}^{N} K_n\,. \end{cases} \tag{3.23}$$

We have efficiently constructed f from (π, σ), now we have to show that (3.21) holds by constructing a Horn proof Π thereof.

We take $n \in \{0, \ldots, N\}$. Let $\Pi^A_n = [a_1, \ldots, a_j]$ and $a \in \Pi^A_n$. We derive a' from a by replacing

$r \in R \cup \{r_{\mathtt{I}}, \mathtt{I}\}$ with $r^{(n,0)}$ (which also means we know $a' = a$ if we have $a = q(t)$ for a pop symbol $q \in Q$).

Assume that $|K_n| = M'$, and let $\Pi_n^B = [b_1, \ldots, b_{j_{M'+1}}]$. We choose j_m such that $b_{j_m} = \mathtt{I}(k_m)$ for every $m \in \{1, \ldots, M'\}$, and we choose $j_0 = 0$. Now let $m \in \{0, \ldots, M'\}$ and $b \in [b_{j_m+1}, \ldots, b_{j_{m+1}}]$.

If b is an a-fact, we derive b' by replacing any $r \in R$ with $r^{(n,0)}$ and $\mathtt{I}$ with $\mathtt{I}^{(0,0)}$. If b is a c-fact, we derive b' from b by replacing $r \in R \cup \{r_{\mathtt{I}}, \mathtt{I}\}$ with $r^{(n,m)}$. Hence, if $b = q(t)$ for a pop symbol $q \in Q$, we again have $b' = b$.

We then set $\widehat{\Pi}_n = [a'_1, \ldots, a'_j, b'_1, \ldots, b'_{j_{M'+1}-1}]$. Now the concatenation $\widehat{\Pi} = \widehat{\Pi}_0 \cdot \cdots \cdot \widehat{\Pi}_N$ is a Horn proof of (3.21).

First we check the conclusion of the Horn proof, i. e., if all facts in the conclusion occur in the proof: As $\mathtt{I}(\sigma(t_{n+1}))$ is an element of Π_n^B for all $n \in \{0, \ldots, N\}$, we have $\mathtt{I}^{(n,m)}(\sigma(t_{n+1})) \in \widehat{\Pi}_n$ for some m (and therefore, we have $\mathtt{I}^{(n)}(\sigma(t_{n+1}))$). For each $k \in K$ with $f(k) = (n, m)$, we have $\mathtt{I}(k) = b_{j_m} \in \Pi_n^B$ by construction of f and so we have $\mathtt{I}^{(n,m-1)}(k) \in \widehat{\Pi}_n$. We can now choose $e_k = (n, m-1) < f(k)$.

Now we check if each fact is justified: The facts of the form a' for some $a \in \Pi_n^A$ are justified because the pushed facts are all indexed with stage $(n, 0)$, and we have $r^{(n,0)}(\sigma(s_n), \kappa^{(n)})$ in the assumption. The facts constructed by pop clauses are not modified.

Of the facts b' for some $b \in \Pi_n^B$, the a-facts are clearly justified. For the facts constructed in Π_n^B by a clause φ, we know that φ is a (generalized) push or send clause, an intruder pop or push clause, or a pop clause. In the latter case, we have $b' = b$. In the other cases, we have a copy of φ in Φ_f which has the appropriate stages as indices, as we use all possible combinations of $e', e'' \leq e$ in the rules.

This shows that for each attack (π, σ), we can efficiently construct a stage mapping f, such that (π, σ, f) fulfills statement (3.21). But it remains to prove that we can also construct a normal Horn proof of that statement.

For each term t define $e_t = \min\{e \in E \mid \mathtt{I}^e(t) \in \Pi\}$ (and similar for other $s \in S$). We now obtain $\widehat{\Pi}$ from Π by replacing each $\mathtt{I}^e(t)$ for some $e \in E$ with $\mathtt{I}^{e_t}(t)$ (and again, similar for other $s \in S$). Now $\widehat{\Pi}$ is a Horn proof of (3.21), and all facts in it are justified: We have $\mathtt{I}^{e_t}(t) \in \Pi$, and for each $\mathtt{I}^e(t)$ used as a prerequisite for the application of a clause $\varphi \in \Phi_f$, we find a clause $\varphi' \in \Phi_f$ which allows $\mathtt{I}^{e_t}(t)$ to be used instead.

This completes the proof of the first part of our claim.

3.3.2 Proof of Lemma 3.11, Part 2

We now assume that (3.21) holds. Hence, we take an appropriate ground substitution σ, a stage mapping f and an execution scheme π as above such that (3.21) holds for (π, σ, f); let Π be a normal Horn proof of that.

Due to Lemma 3.5 we now only need to define appropriate sets R_n and C_n such that the following holds:

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash^{C_n}_{\mathrm{Fresh}(\Phi_{\mathcal{P}})} R_n \qquad \text{for all } n \in \{1, \dots, N\} \text{ and} \tag{3.24 a}$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup R_1 \cup \cdots \cup R_n \vdash_{\mathrm{Basic}(\Phi_{\mathcal{P}})} \mathtt{I}(\sigma(t_{n+1})) \qquad \text{for all } n \in \{0, \dots, N\}\,. \tag{3.24 b}$$

We divide Π in several parts and modify them to get the Horn proofs for theses statements.

We first define a subsequence $\Pi^{\mathrm{pop}} \subseteq \Pi$ of all the facts of the form $q(t)$ for a pop symbol $q \in Q$ and a term t, preserving the relative ordering amongst the elements of Π^{pop}.

For all $n \in \{0, \dots, N\}$, let $\Pi^A_n \subseteq \Pi$ be the subsequence of all facts of the form $r^{(n,0)}(t, \kappa)$ for a push symbol $r \in R$ (note that $r_{\mathtt{I}} \notin R$). Accordingly, let $\Pi^B_n \subseteq \Pi \setminus \Pi^A_n$ be the set of facts with a predicate symbol $p^{(n,m)}$, ordered by their stage, but preserving the ordering amongst facts of the same stage.

Now Π^A_n is nearly a Horn proof of (3.24 a) for each $n \in \{1, \dots, N\}$, we only need to remove the stage annotation. Let ξ be a substitution on terms (extended to sequences of terms) which replaces each predicate symbol of the form r^e with the according symbol r. We now use this substitutions to derive $\widehat{\Pi}^{\mathrm{pop}} = \xi(\Pi^{\mathrm{pop}})$ and $\widehat{\Pi}^A_n, \widehat{\Pi}^B_n$ in the same manner.

Now with $R_n = \widehat{\Pi}^A_n$, the sequence $\widehat{\Pi}^{\mathrm{pop}} \cdot \widehat{\Pi}^A_n$ is a Horn proof of (3.24 a) for all $n \in \{1, \dots, N\}$: This sequence contains only facts which can be derived by clauses in $\mathrm{Fresh}(\Phi_{\mathcal{P}})$, as the sequence Π^{pop} only contains facts obtained by pop clauses, and Π^A_n only uses push clauses of the form (3.20 v). For each fact we can use the same justification as in the Horn proof Π. In addition, we can set C_n to the set of anonymous constants assigned to anonymous variables by the substitutions of the c-facts in Π^A_n.

Similarly, $\widehat{\Pi}^{\mathrm{pop}} \cdot \widehat{\Pi}^A_1 \cdot \dots \cdot \widehat{\Pi}^A_n \cdot \widehat{\Pi}^B_0 \cdot \dots \cdot \widehat{\Pi}^B_n$ is a Horn proof of (3.24 b) for all $n \in \{0, \dots, N\}$. The facts in $\widehat{\Pi}^{\mathrm{pop}}$ are clearly justified, and the choice for $R_1, \dots, R_n$ makes the facts in $\widehat{\Pi}^A_1 \cdot \dots \cdot \widehat{\Pi}^A_n$ a-facts. Thus, we know that no fresh anonymous constants need to be assigned at all: In the Horn proofs Π^B_n no clause of the form (3.20 v) has been applied because no c-fact in Π^B_n is of the form $r^{(n,0)}(t, \kappa)$ (for any r, n, t, κ).

The c-facts in $\widehat{\Pi}_n^B$ are derived by a clause $\varphi \in \Phi_f$ which corresponds to a clause $\varphi' \in \text{Basic}(\Phi_{\mathcal{P}})$, and so we can use this clause as justification. The only clauses weakened during the construction of the stage theory are the (generalized) push and send clauses of the form (3.20 vi), but we know by assumption that each key k occurring in φ' has to be known at the time φ is applied, so for the construction of a fact in stage e, the fact $\mathtt{I}^{e_k}(k)$ for a stage $e_k < e$ has to occur in Π. But then we know that $\mathtt{I}(k)$ is an element of $\widehat{\Pi}_1^B \cdot \dots \cdot \Pi_{n-1}^B$, or an element of Π_n^B occurring prior to the c-fact constructed by φ'.

This concludes the proof of the lemma. □

3.4 Flattening Push Clauses

For our proof we need to *flatten* all push clauses, i. e., only allow clauses of the following form with a predicate symbol $f \in \Sigma_n$ and normal variables $x_1, \dots, x_n$ (and pop predicate symbols and register sequences like in (2.9 b)):

$$q_1(t), \dots, q_m(t), r(t, \kappa) \to r'(x_j, \kappa') \text{ with } t = f(x_1, \dots, x_m), j \in \{1, \dots, n\} \,. \tag{3.25}$$

We will show how to transform push clauses to a sequence of flat push clauses.

Let $\mathcal{P} = (P, \Phi)$ be a protocol, and let φ be a non-flat push clause, i. e., for a simple term t we have $\varphi = [q_1(t), \dots, q_m(t), r(t, \kappa) \to r'(x, \kappa')]$.

First, we take a set of pop clauses that recognize the structure of t, i. e., we add a set of pop symbols Q_t to Q and add a set of pop clauses Φ_t to Φ such that we have $\emptyset \vdash_{\Phi_t} q'(\hat{t})$ for some $q' \in Q_t$ if and only if $\hat{t}$ is an instance of t.

As $x \in \text{sub}(t)$, there is a sequence $[t_1, \dots, t_m]$ of terms with $t = t_1, t_m = x$ with $t_{j+1} \in \text{sub}(t_j)$ and $\text{depth}(t_{j+1}) = \text{depth}(t_j) - 1$ for all $j \in \{1, \dots, m-1\}$. We now choose terms $s_1, \dots, s_{m-1}$ such that for all $j \in \{1, \dots, m-1\}$ we can construct t_j from s_j by substituting $x \in \text{var}(s_j)$ with t_{j+1}.

If, for instance, $t = \langle x', \langle \{x\}_k, x'' \rangle \rangle$, we would have $m = 4$ and the following terms:

$$\begin{aligned} t_1 &= \langle x', \langle \{x\}_k, x'' \rangle \rangle \,, & s_1 &= \langle x', x \rangle \,, \\ t_2 &= \langle \{x\}_k, x'' \rangle \,, & s_2 &= \langle x, x'' \rangle \,, \\ t_3 &= \{x\}_k \,, & s_3 &= \{x\}_k \,, \\ t_4 &= x \,. \end{aligned}$$

Then we take $m - 2$ push symbols $r_2, \ldots, r_{m-1} \in R$ not used before and replace $\varphi \in \Phi$ by the following $m - 1$ clauses:

$$q'(t), q_1(t), \ldots, q_m(t), r(s_1, \kappa) \to r_2(x, \kappa') \,, \tag{3.26}$$

$$r_2(s_2, \kappa_\star) \to r_3(x, \kappa_\star) \,, \tag{3.27}$$

$$\vdots \tag{3.28}$$

$$r_{m-2}(s_{m-2}, \kappa_\star) \to r_{m-1}(x, \kappa_\star) \,, \tag{3.29}$$

$$r_{m-1}(s_{m-1}, \kappa_\star) \to r'(x, \kappa_\star) \,. \tag{3.30}$$

In these clauses we use $\kappa_\star$ from Definition 2.4 to just copy an unmodified version of the register sequence from one fact to the next one.

Note that this transformation is only possible because we have simple terms on the left-hand-side of push clauses, i. e., each variable occurring more than once in a push clause will lead to a new push clause that is non-linear, but flat.

Remark 3.12. From now on we will assume that there are only flat push clauses, so each (generalized) push clause that has a non-flat term on its left-hand-side is transformed according to the method described above. ◁

3.5 The Graph of an Attack

We will now present a structure called *directed acyclic graph of the attack* (ADAG) introduced in [Tru05a, Tru05b], which represents an attack on a protocol: We show that such an ADAG for a protocol exists if and only if there is an attack on this protocol. As we later show a double exponential bound for ADAGs, this structure allows us to present the algorithm which decides security of recursive protocols. Again, we begin with preliminaries.

Definition 3.13. Let Φ and Ψ be stage theories. The theory Ψ is an *instance* of Φ, if each clause in Ψ is an instance of a clause in Φ, more precisely if for each clause $\psi \in \Psi$ there is a substitution $\sigma\colon X \to T(\Sigma, X \cup Y)$ and a clause $\varphi \in \Phi$ such that $\hat{\sigma}(\varphi) = \psi$ where $\hat{\sigma}$ is the extension of σ on clauses. ◁

Term DAGs

Definition 3.14. Let $\bar{\Sigma}$ be a signature. Then $D = (V, F, \mu)$ consisting of a finite set of nodes V, a set of edges $F \subset V \times V \times \mathbb{N}$ with an order in last component, and a labeling function $\mu\colon V \to \bar{\Sigma}$ is a *term DAG* over $\bar{\Sigma}$ if the following holds: For a node $v \in V$ with $\mu(v) = f$ for a function symbol $f \in \bar{\Sigma}_m$, the node v has m ordered successors $v_1, \ldots, v_m$ in the DAG, i. e., $(v, v_j, j) \in F$ for all $i \in \{1, \ldots, m\}$.

In the above situation, we will write $v \doteq_D f(v_1, \ldots, v_m)$, and we will omit D if the DAG is clear from the context. In addition, we recursively define the notation

$$D(v) = c \qquad \text{if } v \doteq_D c\,, \tag{3.31 a}$$

$$D(v) = f(D(v_1), \ldots, D(v_m)) \qquad \text{if } v \doteq_D f(v_1, \ldots, v_m)\,. \tag{3.31 b}$$

A term DAG D is *minimized* if for all nodes v_1, v_2 in D, we have $D(v_1) \neq D(v_2)$. ◁

Embeddings

Definition 3.15. Let $T \subseteq T(\Sigma \cup \Gamma, X \cup Y)$ be a set of terms. Let $D = (V, F, \mu)$ be a term DAG over $\Sigma \cup \Gamma$. A function $\alpha\colon \mathrm{sub}(T) \to V$ is a *D-embedding for T* if for all $v \in V$ and all $y \in Y \cap \mathrm{sub}(T)$ we have

$$v \doteq c \in \Gamma \qquad \text{if } v = \alpha(y)\,, \tag{3.32 a}$$

$$v \doteq f(v_1, \ldots, v_m) \text{ and } \alpha(t_j) = v_j \text{ for all } j \in \{1, \ldots, m\} \qquad \text{if } v = \alpha(f(t_1, \ldots, t_m))\,. \tag{3.32 b}$$

Two embeddings α_1 and α_2 are *compatible* if we have $\alpha_1(x) = \alpha_2(x)$ for each regular and anonymous variable $x \in \mathrm{dom}(\alpha_1) \cap \mathrm{dom}(\alpha_2) \cap (X \cup Y)$.

For $v \in V$ and $t \in T$ we further write $t \mapsto v$ if there exists a unique embedding α for $\{t\}$ which is determined by $\alpha(t) = v$. We write $(t, t') \mapsto (v, v')$ if both $t \mapsto v$ and $t' \mapsto v'$ hold true and both embeddings are compatible. ◁

Register Sequences

Definition 3.16. Let κ be a variable register sequence and κ' be a ground register sequence. We denote the substitution which maps each variable $y \in \kappa$ to the corresponding anonymous constant $\mathsf{c} \in \kappa'$ by $\mathrm{subst}(\kappa, \kappa')$ if such a substitution is well-defined; i. e., if for each $y \in \kappa$ we have exactly one $\mathsf{c} \in \kappa'$:

$$\mathrm{subst}(\kappa, \kappa') = \{\, y \mapsto \mathsf{c} \mid \text{there exists } z \in \{1, \ldots, Z\} \text{ such that } \kappa[z] = y \in Y \text{ and } \kappa'[z] = \mathsf{c} \in \Gamma \,\}\,. \quad (3.33)$$

Again, we will write $(\kappa_1, \kappa_2) \mapsto (\kappa_1', \kappa_2')$ if both substitutions $\sigma_1 = \mathrm{subst}(\kappa_1, \kappa_1')$ and $\sigma_2 = \mathrm{subst}(\kappa_2, \kappa_2')$ exists and are compatible, i. e., if for each $y \in \mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2)$ we have $\sigma_1(y) = \sigma_2(y)$. ◁

For example, the application of a push clause $[\ldots r_1(t_1, \kappa_1) \to r_2(t_2, \kappa_2)]$ to a fact $r_1(t_1', \kappa_1')$ will result in a fact $r_2(t_2', \kappa_2')$ where we have $(\kappa_1, \kappa_2) \mapsto (\kappa_1', \kappa_2')$.

We will now define ADAGs and after the formal definition we will give some intuition of how ADAGs resemble terms, predicate symbols and so on.

ADAGs

Definition 3.17. Let $\mathcal{P} = (P, \Phi)$ be a protocol, let $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$ be an execution scheme and f be a stage mapping for π for the set E of stages.

Let Q_f be the set of all predicate symbols occurring in Φ_f and let $K_\mathcal{P}$ be the keys used in the protocol. By $T_\mathcal{P}$ we denote the set of *static terms* occurring in $\mathcal{P}$, more precisely:

$$T_\mathcal{P} = \{¢, \$\} \cup \{\, t_n, s_n \mid n \in \{1, \ldots, N\}\} \cup K_\mathcal{P}\,. \quad (3.34)$$

Let $\mathcal{D}$ be the tuple $(D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ with

- a finite term DAG $D = (V, F, \mu)$ over $\Sigma \cup \widehat{\Gamma}$,
- a stage theory Ψ which is an instance of the stage theory Φ_f,
- a finite set of anonymous constants $\widehat{\Gamma}$,
- an embedding function $\alpha\colon \mathrm{sub}(T_\mathcal{P}) \to V$, which embeds $T_\mathcal{P}$ in D,
- a witness function $\beta\colon V \times Q_f \times \mathbb{N} \dashrightarrow V \times \mathbb{N} \times \Psi$,

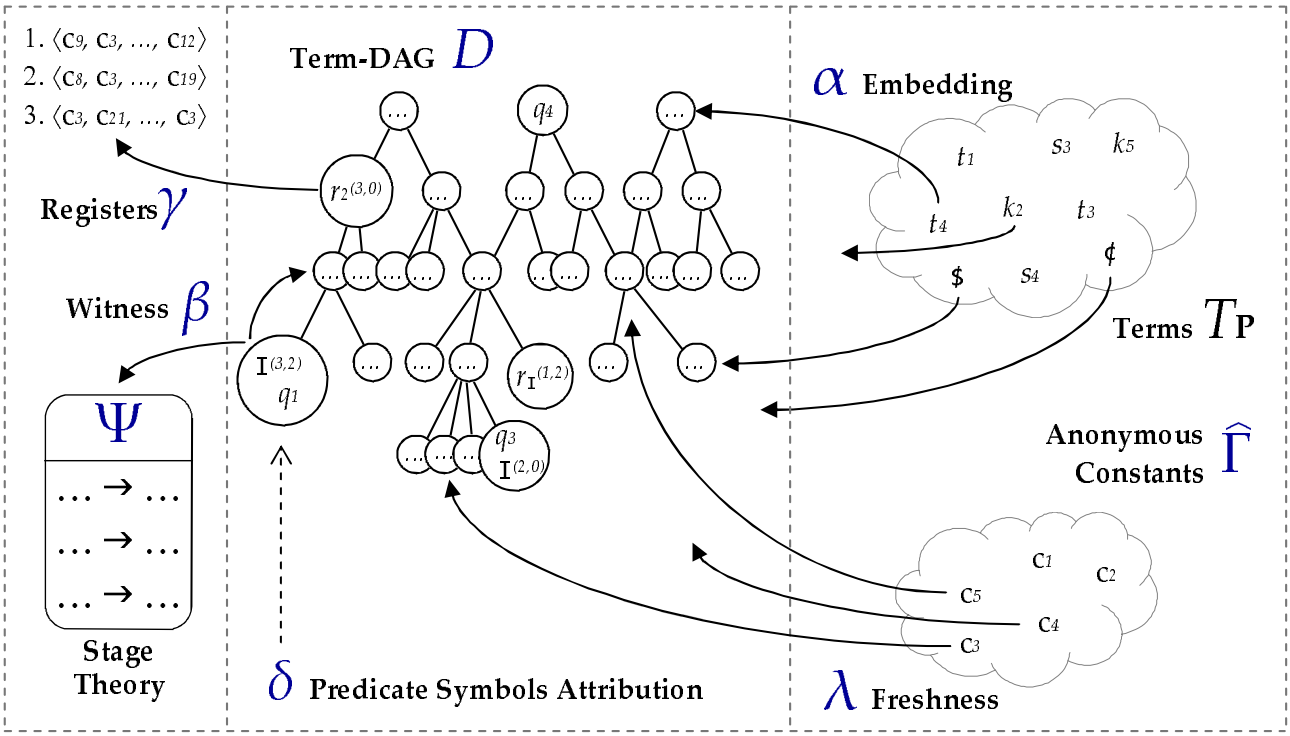

Figure 3.4: Illustration of a graph of an attack $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$

- a register sequence function $\gamma\colon V \times Q_f \times \mathbb{N} \dashrightarrow \widehat{\Gamma}^Z$,
- a predicate symbol labeling function $\delta\colon V \dashrightarrow 2^{Q_f}$, and
- a freshness function $\lambda\colon \widehat{\Gamma} \dashrightarrow V \times Q_f \times \mathbb{N} \times \{1, \ldots, Z\}$.

Then $\mathcal{D}$ is called *DAG of the attack (ADAG)* for $\mathcal{P}$ and (π, f) if the following three conditions hold (see the explanation below for additional notation we use):

(i) For all $n \in \{1, \ldots, N\}$ and all $k \in \mathrm{dom}(f)$ we have

$$\mathtt{I}^e \in \delta(\alpha(t_n)) \text{ for a stage } e \ll n\,, \tag{ADAG i a}$$
$$\mathtt{I}^e \in \delta(\alpha(k)) \text{ for a stage } e < f(k)\,, \text{ and} \tag{ADAG i b}$$
$$\mathtt{I}^e \in \delta(\alpha(\$)) \text{ for a stage } e\,. \tag{ADAG i c}$$

(ii) For each node v and each $s \in S$ we have

$$s^e \in \delta(v) \text{ for at most one } e \in E\,. \tag{ADAG ii}$$

(iii) For each node v and each predicate symbol $p \in \delta(v)$ at least *one* of the following conditions holds, i. e., p is justified because of

(iii a) the application of a protocol step (or the intruder's initial knowledge)

$$\begin{aligned} v = \alpha(\dot{\varsigma}) \text{ with } p = \mathtt{I}^{(0,0)} \qquad & \text{(ADAG iii a)} \\ \text{or } \; v = \alpha(s_n) \text{ with } p = r_n^{(n,0)} & \text{ for some } n \in \{1,\dots,N\}, \\ \gamma^\star(v,p) = \{0\}, & \\ \gamma(v,p,0,z) \neq \gamma(v,p,0,z') & \text{ for } z \neq z', \text{ and} \\ \lambda(\gamma(v,p,0,z)) = (v,p,0,z) & \text{ for all } z \in \{1,\dots,Z\} \; ; \end{aligned}$$

(iii b) the application of an (intruder) pop clause with $p \in \widetilde{Q}$

$$\begin{aligned} v \rightrightarrows f(v_1,\dots,v_m) \text{ and} \qquad & \text{(ADAG iii b)} \\ \text{there is a clause } \psi = \big[p_1(x_1),\dots,p_m(x_m) \to p(f(x_1,\dots,x_m))\big] & \in \Psi \\ \text{with } p_j \in \delta(v_j) \text{ for all } j \in \{1,\dots,m\} \text{ and} & \\ v_{j_1} = v_{j_2} \text{ for all } j_1, j_2 \in \{1,\dots,m\} \text{ with } x_{j_1} = x_{j_2} \; ; & \end{aligned}$$

(iii c) the application of an (intruder) push clause with $p \in \widetilde{R}$ witnessed by β

$$\begin{aligned} \gamma^\star(v,p) \neq \emptyset, \text{ and for all } i \in \gamma^\star(v,p) \qquad & \text{(ADAG iii c)} \\ \text{we have } v' \rightrightarrows f(v_1,\dots,v_m) & \\ \text{with } v = v_j \text{ for some } \; j \in \{1,\dots,m\} & \\ \text{such that } \beta(v,p,i) = (v', p_m, i', \psi), & \\ \text{with } \psi = \big[p_1(t'),\dots,p_m(t',\kappa') \to p(x_j,\kappa)\big] \in \Psi, & \\ t' = f(x_1,\dots,x_m), & \\ p_j \in \delta(v') \text{ for all } j \in \{1,\dots,m\}, & \\ v_{j_1} = v_{j_2} \text{ for all } j_1, j_2 \in \{1,\dots,m\} \text{ with } x_{j_1} = x_{j_2}, \text{ and} & \\ (\kappa,\kappa') \mapsto (\gamma(v,p,i), \gamma(v',p_m,i')), \text{ as well as} & \\ \lambda(\gamma(v,p,i,z)) = (v,p,i,z') \text{ for some } z' \text{ with } \; \kappa[z] = \kappa[z'] & \\ \text{for all } z \text{ with } \kappa[z] \in \Upsilon^\star \; ; & \end{aligned}$$

(iii d) or the application of a send clause with $p \in \widetilde{\mathtt{I}}$ witnessed by β

$$\begin{aligned} &\text{we have } \beta(v,p,0) = (v',p_m,i',\psi) && \text{(ADAG iii d)}\\ &\qquad \text{with } \psi = \big[p_0(t), p_1(t'), \ldots, p_m(t',\kappa') \to p(t)\big] \in \Psi,\\ &\qquad\quad p_0 \in \delta(v), \text{ and}\\ &\qquad\quad p_j \in \delta(v') \text{ for all } j \in \{1,\ldots,m\}, \text{ as well as}\\ &\qquad (\sigma_\kappa(t), t') \mapsto (v, v') \text{ for } \sigma_\kappa = \mathrm{subst}(\kappa', \gamma(v', p_m, i'))\ . \end{aligned}$$

◁

Remark 3.18. For a node v and $p \in \delta(v) \cap \widetilde{R}$ we define $\gamma^\star(v,p)$ to contain what we call the *indices* of the register sequences, i. e.,

$$\gamma^\star(v,p) = \{\, i \in \mathbb{N} \mid \gamma(v,p,i) \text{ is defined} \}\ . \tag{3.35}$$

By abuse of notation, we will use β as a function $\beta\colon V \times Q_f \times \mathbb{N} \dashrightarrow V \times Q_f \times \mathbb{N} \times \Psi$, i. e., mapping a node v, a symbol p and a register sequence i not only to (v', i', φ), but to (v', p', i', φ) if p' is in φ, i. e., if we have $\varphi = [\ldots, p'(t', \kappa') \to p(t)]$ or $\varphi = [\ldots, p'(t', \kappa') \to p(t, \kappa)]$.

Similarly, we will write $\gamma(v,p,i,z)$ instead of $\gamma(v,p,i)[z]$, so we are using γ as a function $\gamma\colon V \times Q_f \times \mathbb{N} \times \{1,\ldots,Z\} \to \widehat{\Gamma}$. ◁

We will now give some intuition what was defined here; keep in mind that we will later show that an ADAG for a protocol represents an attack on that protocol, so we will explain how an ADAG represents a Horn proof of (3.21).

First of all, take a look at Figure 3.4: An ADAG consist mainly of a term DAG D representing all terms that occur in the run of a protocol. The nodes of the DAG are labeled with function symbols and constants, so for each node v there is a unique term t defined by the labeling of v and its descendants, we will say *v corresponds to t*. The function α embeds the static terms of the protocol to D.

In addition the nodes are labeled with predicate symbols by δ: A node v corresponding to a term t is labeled with all predicate symbols p for which there is $p(t)$ or $p(t,\kappa)$ in the Horn proof. For each node and each symbol there may be an unbounded set of register

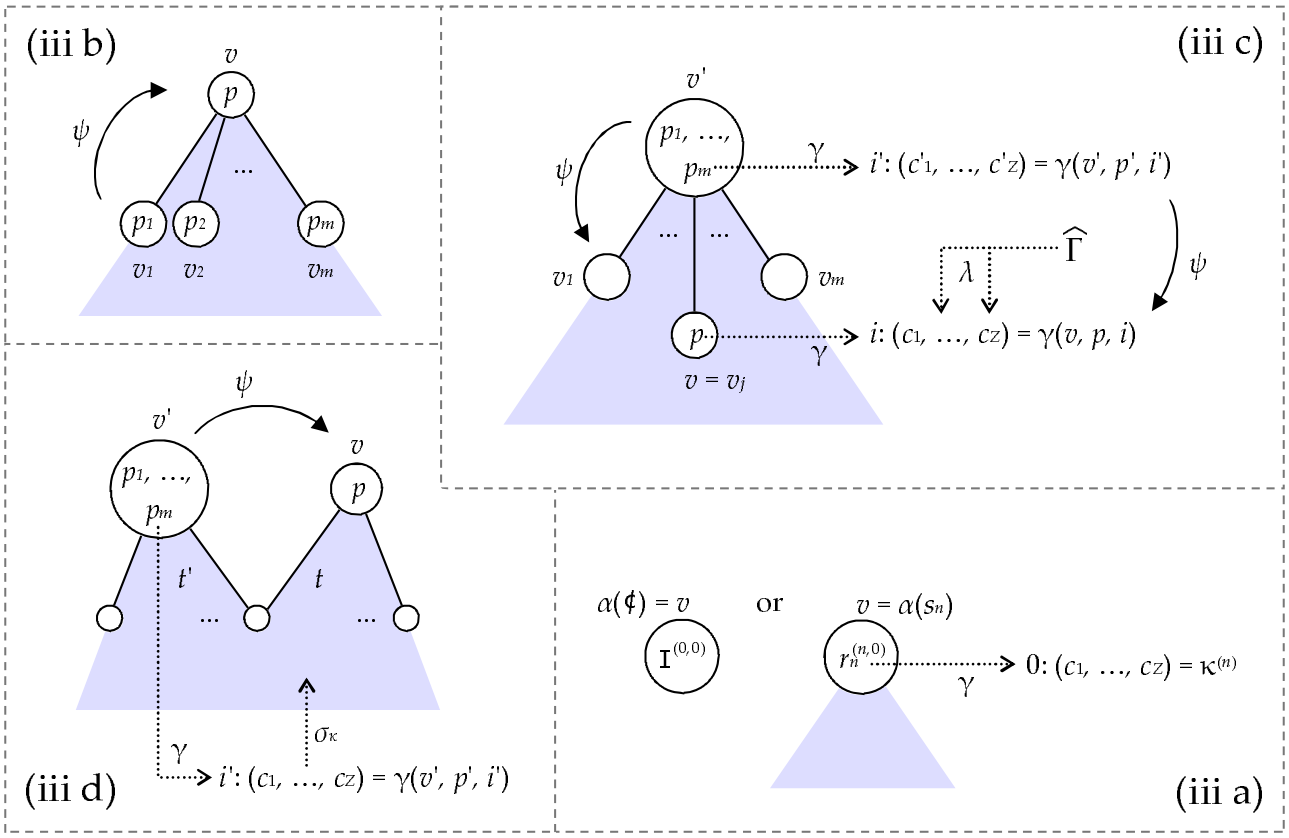

Figure 3.5: Illustration of the ADAG conditions (ADAG iii a) to (ADAG iii d)

sequences κ, in the ADAG they are represented by γ. Moreover, the freshness of a generated anonymous constant in a register sequence of a node is witnessed by λ through the definition in (ADAG iii c).

The set of conditions guarantee the resemblance between ADAGs and Horn proofs of (3.21): Condition (ADAG i) ensures that the conclusion of the Horn proof is fulfilled by the ADAG, while condition (ADAG ii) corresponds to the normality of the Horn proof.

Furthermore, each fact in a Horn proof is either an a-fact or constructed by a clause. Accordingly, each predicate symbol in the ADAG needs a justification. This is ensured by condition (ADAG iii) as illustrated in Figure 3.5: The fact is either an assumed fact (ADAG iii a), or there is a pop clause (ADAG iii b), a push clause (ADAG iii c), or a send clause (ADAG iii d) that justifies the fact. For the last two cases this justification is witnessed by β.

We can now express the following theorem, linking the existence of an ADAG to the insecurity of a protocol:

Theorem 3.19 (Characterization of Attacks by ADAGs)**.** *Let $\mathcal{P} = (P, \Phi)$ be a protocol.*

1. *If (π, σ) is an attack on the protocol then there exists a stage mapping f and an ADAG $(D, \Phi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ for $\mathcal{P}$ and (π, f).*
2. *If there is an ADAG for $\mathcal{P}$ for an execution scheme and a stage mapping (π, f) then there exists an attack (π, σ) on the protocol for a substitution σ.*

3.5.1 Proof of Theorem 3.19, Part 1

Proof. Let $\mathcal{P} = (P, \Phi)$ be a protocol, and let (π, σ) be an attack on that protocol. By Lemma 3.11 we know that there exists a stage mapping f and sets C_0, C such that statement (3.21) holds; let Π be a Horn proof of that. We will now construct a corresponding ADAG.

- Let $T = \{\sigma(t) \mid t \in T_{\mathcal{P}}\}$ be a set of ground terms. We define a minimized term DAG $D = (V, F, \mu)$ over $\Sigma \cup \widehat{\Gamma}$ (with $\widehat{\Gamma}$ as defined below) and the function $\hat{\alpha}$ to be a D-embedding for T. We also define α, which shall be a D-embedding for $T_{\mathcal{P}}$ compatible with $\hat{\alpha}$, i. e., $\alpha(t) = \hat{\alpha}(\sigma(t))$ for each $t \in T_{\mathcal{P}}$.

- Let $\widehat{\Gamma}$ be the set of all anonymous constants occurring in the Horn proof and the assumption, i. e., including C and C_0.

  For all $\mathsf{c} \in C_0$ there is a register sequence $\kappa^{(n)}$ and some $z \in \{1, \dots, Z\}$ with $\mathsf{c} = \kappa^{(n)}[z]$ such that $r_n^{(n,0)}(s_n, \kappa^{(n)})$ appears in the assumption of the Horn proof. Now set $\gamma(\alpha(s_n), r_n^{(n,0)}, 0) = \kappa^{(n)}$ and $\lambda(\mathsf{c}) = (\alpha(s_n), r_n^{(n,0)}, 0, z)$.

- Start with $\delta(v) = \emptyset$ for all $v \in D$. For each fact $p(t) \in \Pi$ with $\hat{\alpha}(t) = v$ we put $p \in \delta(v)$.

  For each push symbol p and each term t for which there is a fact $p(t, \kappa)$ in Π, let $\{\kappa_1, \kappa_2, \dots, \kappa_I\}$ be the set of register sequences with $p(t, \kappa_i) \in \Pi$ for each $i \in \{1, \dots, I\}$. Assume that $\hat{\alpha}(t) = v$. Then we put $p \in \delta(v)$, and in addition we set $\gamma(v, p, i) = \kappa_i$ for each $i \in \{1, \dots, I\}$.

- If $p(\hat{t})$ is a c-fact in Π constructed by a send clause $\varphi = [\dots, p'(t', \kappa') \to p(t)] \in \Phi_f$ and a substitution $\bar{\sigma}$, let $v = \hat{\alpha}(\hat{t})$ and $v' = \hat{\alpha}(\bar{\sigma}(t'))$. Let i' be the index such that $\gamma(v', p', i') = \bar{\sigma}(\kappa')$. We then set $\beta(v, p, 0) = (v', i', \varphi)$.

  If $p(\hat{t}, \hat{\kappa})$ is a c-fact in Π constructed by a push clause $\varphi = [\dots, p'(t', \kappa') \to p(t, \kappa)] \in \Phi_f$ and a substitution $\bar{\sigma}$, let $v = \hat{\alpha}(\hat{t})$ and $v' = \hat{\alpha}(\bar{\sigma}(t'))$. Then for let i be the index such that $\gamma(v, p, i) = \hat{\kappa}$, and let i' be the index with $\gamma(v', p', i') = \hat{\kappa}'$. We then set $\beta(v, p, i) = (v', i', \varphi)$, and we also set $\lambda(\bar{\sigma}(y^\star)) = (v, p, i, z)$ for each $y^\star \in \mathrm{var}_{Y^\star}(\kappa)$ and for one $z \in \{1, \dots, Z\}$ with $y^\star = \kappa[z]$.

Now $\mathcal{D} = (D, \Phi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ is an ADAG for $\mathcal{P}$ and (π, f):

First we need to check if $\mathcal{D}$ is well-defined, i. e., if all elements of $\mathcal{D}$ are of the right type. Most of this directly holds by definition, the interesting thing is to check that λ actually is a functions, meaning there for each $\mathsf{c} \in \widehat{\Gamma}$ there is at most one (v, p, i, z). This is ensured by the definition of Horn proofs: In Π each $\mathsf{c} \in C \subseteq \widehat{\Gamma}$ is assigned only once by one substitution to a fresh variable $y^\star \in Y^\star$, possibly multiple times in one register sequence. In addition, the anonymous constants $\mathsf{c} \in C_0 \subseteq \widehat{\Gamma}$ are used in the assumption of the Horn proof Π, but they are not assigned anywhere else in the protocol.

We can now check the properties (ADAG i a) to (ADAG iii d): Condition (ADAG i) is ensured by the Horn proof Π, condition (ADAG ii) holds because of the normality of the proof, for both conditions we refer to Lemma 3.11.

For the third condition we look at the reason for each node's creation: For a node $v \in D$ with a predicate symbol $p \in \delta(v)$ we have a term $\hat{t} \in T$ with $v = \hat{\alpha}(\hat{t})$. We now take the fact $a \in \Pi$ which contains $\hat{t}$, i. e., which has the form $p(\hat{t})$ or $p(\hat{t}, \hat{\kappa})$. If this is an a-fact, we

know that (ADAG iii a) holds. If this is a c-fact, we have a clause φ constructing a. If φ is an (intruder) pop clause, case (ADAG iii b) holds.

If a was constructed by a push clause $\varphi = [p_1(t'), \ldots, p_m(t', \kappa') \to p(t, \kappa)] \in \Phi_f$ with the substitution $\bar{\sigma}$ (this means $\bar{\sigma}(t) = \hat{t}$), we know (ADAG iii c) holds: We have an index i with $\gamma(v, p, i) = \bar{\sigma}(\kappa)$. There is a fact $p_m(\hat{t}', \hat{\kappa}') \in \Pi$ with $\hat{t}' = \bar{\sigma}(t')$, which leads to a corresponding node $v' = \hat{\alpha}(\hat{t}')$. As the definition of a Horn proof holds for Π, we put the predicate symbols $p_1, \ldots, p_m$ in $\delta(v')$. As our term DAG is minimized, the condition about equal nodes for equal variables holds. Because of the application of φ we have $(\kappa, \kappa') \mapsto (\hat{\kappa}, \hat{\kappa}')$, where we know by definition of γ that we have $(\hat{\kappa}, \hat{\kappa}') = (\gamma(v, p, i), \gamma(v', p_m, i'))$. Finally, for each $y^\star \in \mathrm{var}_{Y^\star}(\kappa)$ we defined λ to point to (v, p, i, z) for some z with $\kappa[z] = y^\star$.

If a was constructed by a send clause $\varphi = [p_0(t), p_1(t'), \ldots, p_m(t', \kappa') \to p(t)] \in \Phi_f$ with the substitution $\bar{\sigma}$ (this again means $\bar{\sigma}(t) = \hat{t}$), we know that (ADAG iii d) holds with a similar argumentation: There is a fact $p_m(\hat{t}', \hat{\kappa}') \in \Pi$ with $\hat{t}' = \bar{\sigma}(t')$, which again leads to the node $v' = \hat{\alpha}(\hat{t}')$. The predicate symbol p_0 is an element of $\delta(v)$, the symbols $p_1, \ldots, p_m$ are elements of $\delta(v')$. The interesting part here is the compatibility of the embeddings:

As t' and t are elements of $T_{\mathcal{P}}$, we know by the definition of the embedding $\hat{\alpha}$ that $(t, t') \mapsto (v, v')$, as for each variable in $x \in \mathrm{var}(t) \cap \mathrm{var}(t')$ we know that $\hat{\alpha}(\bar{\sigma}(x))$ points to exactly one node. However, that is not enough – we have to show that each node corresponding to an anonymous variable $y \in \mathrm{sub}(t)$ is labeled with the correct anonymous constant.

In φ, each anonymous variable $y \in \mathrm{sub}(t)$ has to occur in the register sequence κ', so in the Horn proof Π the anonymous constant $\mathsf{c} = \bar{\sigma}(y)$ has to occur in $\hat{\kappa}'$. By construction of the DAG this gives us $\mathsf{c} \in \gamma(v', p_n, i')$ for an index i'; so we know $v \stackrel{\rightarrow}{=} \sigma_{\kappa'}(t)$ for $\sigma_{\kappa'} = \mathrm{subst}(\kappa', \gamma(v', p_n, i'))$.

As there are no anonymous variables in t', we can substitute each anonymous variable $y \in \mathrm{sub}(t)$ with the corresponding anonymous constant occurring in $\hat{\kappa}'$, without affecting the compatibility of the embeddings. Thus, we know $(\sigma_{\kappa'}(t), t') \mapsto (v, v')$.

This proves that $\mathcal{D}$ is an ADAG for $\mathcal{P}$ and (π, f). We can now address the other direction of the proof.

3.5.2 Proof of Theorem 3.19, Part 2

We now assume that there is an ADAG $\mathcal{D} = (D, \Phi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ for a protocol $\mathcal{P}$ and a pair (π, f) of an execution scheme $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$ and a stage mapping f.

We have to construct a substitution σ such that (π, σ) is an attack. We use Lemma 3.11 and can thus construct a Horn proof Π of statement (3.21). For each $x \in T_{\mathcal{P}} \cap X$ we set $\sigma'(x) = t$ for the unique term t with $\alpha(x) \doteq t$.

1. Now start with Π' being the empty sequence.
2. Put all the facts of the form $q(t)$ in Π' where we have $q \in Q$ in the set $\delta(v)$ for a node $v \doteq t$; putting $q(t_1)$ before $q(t_2)$ if $t_1 \in \mathrm{sub}(t_2)$.
3. For each stage e occurring in $\mathcal{D}$, beginning with the smallest stage and ascending, we add the following facts:
   a) We put all the facts of the form $p(t, \kappa)$ in Π' where we have $p \in \widetilde{R}$ in the set $\delta(v)$ for a node $v \doteq t$, and where $\kappa = \gamma(v, r, i)$ for an index i. We put $p^e(t_1, \kappa_1)$ before $p^e(t_2, \kappa_2)$ if we have $t_2 \in \mathrm{sub}(t_1)$.
   b) We then put all the facts of the form $\mathtt{I}^e(t)$ and $q^e_{\mathtt{I}\downarrow}(t)$ in Π' where we have $\mathtt{I}^e$ or $q^e_{\mathtt{I}\downarrow} \in \delta(v)$ for a stage $e \in E$ and a node $v \doteq t$; putting $\mathtt{I}^e(t_1)$ before $\mathtt{I}^e(t_2)$ if we have $t_1 \in \mathrm{sub}(t_2)$.

Now Π is a Horn proof for statement (3.21), more precisely: Π is a normal Horn proof of the following statement with $t_{N+1} = \$$ for stages $e_n \ll n+1$ and $e_k < f(k)$, and for a set of pairwise disjoint register sequences $\kappa^{(n)}$ with C being disjoint with each $\kappa^{(n)}$:

$$\left\{\mathtt{I}^{(0,0)}(\dot{\varsigma})\right\} \cup \left\{ r^{(n,0)}(\sigma(s_n), \kappa^{(n)}) \;\middle|\; n \in \{1, \ldots, N\} \right\} \vdash^{C}_{\Phi_f} \left\{ \mathtt{I}^{e_n}(\sigma(t_{n+1})) \;\middle|\; n \in \{0, \ldots, N\} \right\} \cup \left\{ \mathtt{I}^{e_k}(k) \;\middle|\; k \in \mathrm{dom}(f) \right\} \quad (3.36)$$

Let A be the assumption and B be the conclusion of this statement. We know that Π is a normal Horn proof because of (ADAG ii). The assumption A corresponds to (ADAG iii a). By (ADAG i a), (ADAG i b), and (ADAG i c), we also know $B \subseteq \Pi$ for stages $e_n \ll i+1$ and $e_k < f(k)$.

Hence, we only have to show that the Horn proof Π is conclusive, i. e., that each fact in Π is a valid a- or c-fact.

Let $a = p(\hat{t})$ or $a = p(\hat{t}, \hat{\kappa})$ be a fact in Π. We know by construction that there is a node v with $v \doteq t$ and $p \in \delta(v)$. We can now use condition (ADAG iii) from the definition of an ADAG to distinct the following cases as one of them has to apply for p, v:

(iii a) In this case a is an a-fact, i. e., $a \in A$.

(iii b) Now a is a c-fact constructed by an (intruder) pop clause. By construction we put $p_j(\hat{t}_j) \in \Pi$ before a for all $j \in \{1, \ldots, m\}$ and for $v_j \stackrel{?}{=} \hat{t}_j$, because $\hat{t}_j \in \mathrm{sub}(\hat{t})$ holds. By (ADAG iii b) we have a matching clause $\psi \in \Psi$, which is an instance of a clause $\varphi \in \Phi_f$. Hence, $p(\hat{t})$ can be derived in Π by that φ.

(iii c) In this case $a = p(\hat{t}, \hat{\kappa})$ is a c-fact constructed by a push clause φ of which ψ is an instance (with $\beta(v, p, i) = (v', i', \psi)$ for indices i, i'); let σ_ψ be the substitution with $\sigma_\psi(\varphi) = \psi$. We put the prerequisites for that construction $p_j(\hat{t}')$ and $p_m(\hat{t}', \hat{\kappa}')$ in Π before a, because we put the pop facts to the beginning and because $\hat{t} \in \mathrm{sub}(\hat{t}')$ holds.

In addition, we find a substitution $\bar{\sigma}_1$ with $\bar{\sigma}_1(t') = \hat{t}'$, and $\bar{\sigma}_1(t) = \hat{t}$, and another substitution, $\bar{\sigma}_2$, which embeds (κ, κ') to $(\hat{\kappa}, \hat{\kappa}') = (\gamma(v, p, i), \gamma(v', p_m, i'))$.

These two substitutions can be chosen disjoint as $\mathrm{var}(t') \cap Y = \emptyset$, so we unify them to $\bar{\sigma}_3 = \bar{\sigma}_1 \cup \bar{\sigma}_2$ and set $\bar{\sigma} = \bar{\sigma}_3 \circ \sigma_\psi$.

Now a is constructed from $p_j(\hat{t}')$ and $p_m(\hat{t}', \hat{\kappa}')$ with the clause φ and the substitution $\bar{\sigma}$.

(iii d) In the last case, a is a c-fact constructed by a send clause, and the same argumentation as in case (ADAG iii c) applies, with $\bar{\sigma}_2$ being $\mathrm{subst}(\kappa', \gamma(v', p_n, i'))$.

This concludes our proof; we have shown that ADAGs and attacks on protocols are equivalent notions. □

4 Scaling Down ADAGs

Theorem 3.19 characterizes the existence of an attack by a special structure with a set of local properties. The ADAG for an attack can now be guessed and checked by an algorithm if we find a bound for the size of the ADAG.

4.1 Preliminaries

To do so we will first define some notation and show certain properties of ADAGs that are important to keep in mind.

Let $\mathcal{P} = (P, \Phi)$ be a protocol over (Q, R), and let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be an ADAG for $\mathcal{P}$ and a (π, f), with V being the set of nodes in D and $\pi = [t_n \to r_n(s_n)]_{n=1}^{N}$.

Let E be a set of stages for P. We again use $\widetilde{Q}$ as pop symbols and $\widetilde{R}$ as push symbols, using

$$\widetilde{\mathtt{I}} = \{\mathtt{I}\}^{(E)} \ , \tag{4.1 a}$$

$$\widetilde{\mathtt{I}}^{+} = \{\mathtt{I}, q_{\mathtt{I}\downarrow}\}^{(E)} \ . \tag{4.1 b}$$

$$\widetilde{Q} = Q \cup \widetilde{\mathtt{I}}^{+} \ , \tag{4.1 c}$$

$$\widetilde{R} = \left(R \cup \{r_{\mathtt{I}}, r_{\mathtt{I}_{\mathtt{H}}}\} \cup \left\{ r_{\mathtt{I}_{\mathtt{H}_k}} \;\middle|\; k \in K \right\}\right)^{(E)} \ , \tag{4.1 d}$$

4.1.1 Notation

Definition 4.1. We define the following relations, sets and notations:

- For two nodes $v, v' \in V$ we will use the relation $v \to_{\mathcal{D}} v'$ if there is an edge (v, v', j) in D for a $j \in \mathbb{N}$. Let $\to_{\mathcal{D}}^{*}$ be the reflexive transitive closure of $\to_{\mathcal{D}}$.

  For a set of nodes W we will also write $W \to_{\mathcal{D}} v$ if $w \to_{\mathcal{D}} v$ holds for a node $w \in W$.

- For two nodes $v, v' \in V$ we define a second kind of edges, *β-edges*, in the following way: $(v, p, i) \rightsquigarrow_{\mathcal{D}} (v', p', i)$ if conditions (ADAG iii c) or (ADAG iii d) hold for v' and p' such that $\beta(v', p', i) = (v, p, i, \varphi)$. Let $\rightsquigarrow_{\mathcal{D}}^*$ be the reflexive transitive closure of $\rightsquigarrow_{\mathcal{D}}$.

  A matching sequence of β-edges $(v_1, p_1, i_1) \rightsquigarrow (v_2, p_2, i_2) \rightsquigarrow \cdots \rightsquigarrow (v_{m+1}, p_{m+1}, i_{m+1})$ is called a *β-path* for some $m \in \mathbb{N}$ called the *length* of the path. We will omit the third component or both the second and third component and just write $v \rightsquigarrow_{\mathcal{D}} v'$ if we have $(v, p, i) \rightsquigarrow_{\mathcal{D}} (v', p', i')$ for some p, p', i, i'.

  In addition, we will further simplify the β-notation: As condition (ADAG ii) guarantees that for each node u there is at most one stage e with $\mathtt{I}^e \in \delta(u)$, we will use $\beta(u, \mathtt{I})$ instead of $\beta(u, \mathtt{I}^e, 0)$ if such a stage e exists, and $(v, p, i) \rightsquigarrow (u, \mathtt{I})$, respectively.

- A node v of D is *static* if it belongs to the static terms of the protocol like t_n and s_n, i. e., if $v = \alpha(t)$ for a term $t \in \mathrm{sub}(T_{\mathcal{P}})$. We also need the nodes reachable from static nodes in a bounded number of steps:

$$\mathrm{Static}_{\mathcal{D}} = \{ v \in V \mid \alpha(t) = v \text{ for a term } t \in \mathrm{sub}(T_{\mathcal{P}}) \} \ , \tag{4.2}$$

$$\mathrm{Static}_{\mathcal{D}}^{+} = \left\{ w \in V \mid v \to_{\mathcal{D}}^{m} w \text{ for a term } v \in \mathrm{Static}_{\mathcal{D}} \text{ with } m \leq |\mathcal{P}|^2 \right\} \ . \tag{4.3}$$

- A node v *belongs to step* n if we have $\alpha(t_n) \to_{\mathcal{D}}^* v$ or $\alpha(s_n) \to_{\mathcal{D}}^* v$.

$$\mathrm{Steps}_{\mathcal{D}}(\leq n) = \{ v \in V \mid \alpha(t) \to_{\mathcal{D}}^* v \text{ for a term } t \in \{ ¢, t_1, s_1, \ldots, t_n, s_n \} \} \ . \tag{4.4}$$

  A node v can belong to multiple steps if, for example, $\alpha(t_n) \to_{\mathcal{D}}^* \alpha(s_{n'}) \to_{\mathcal{D}}^* v$ holds.

- A node $u \in V$ is a *goal* if u is the top node of the resulting term of a send action, i. e., if $\mathtt{I}^e \in \delta(u)$ because of a send clause for a stage e:

$$\mathrm{Goals}_{\mathcal{D}}(n) = \left\{ u \in V \;\middle|\; \begin{array}{r} \text{there is an } m \text{ such that } \mathtt{I}^{(n,m)} \in \delta(u) \text{ and} \\ \text{(ADAG iii d) holds for } u \text{ and } \mathtt{I}^{(n,m)} \end{array} \right\} , \tag{4.5}$$

$$\mathrm{Goals}_{\mathcal{D}} = \bigcup_{n=1}^{N} \mathrm{Goals}_{\mathcal{D}}(n) \ , \tag{4.6}$$

$$\mathrm{Goals}_{\mathcal{D}}(v) = \{ u \in \mathrm{Goals}_{\mathcal{D}} \mid v \rightsquigarrow^* (u, \mathtt{I}^e, 0) \} \ . \tag{4.7}$$

  In that case (n, m) is the *stage of the goal*, and n is the *protocol step of the goal*; we will also call u an n-goal.

- For a goal u with $\mathtt{I}^e \in \delta(u)$ and $\beta(u, \mathtt{I}^e, 0) = (v, p, i, \varphi)$ with $\psi = [\ldots, p(t', \kappa) \to \mathtt{I}^e(t)]$, we find an embedding α_u which embeds $(\sigma_\kappa(t), t')$ to (u, v) for the substitution $\sigma_\kappa = \mathrm{subst}(\kappa, \gamma(v, p, i))$.

  We then define the set of *fixed nodes*, i. e., nodes belonging to t and t', and the set of *nodes of variables*, i. e., nodes representing the regular variables in t:

$$\mathrm{Fix}_{\mathcal{D}}(u) = \left\{ w \in V \mid \alpha_u(s) = w \text{ for a term } s \in \mathrm{sub}(\sigma_\kappa(t)) \cup \mathrm{sub}(\sigma_\kappa(t')) \right\} , \tag{4.8}$$

$$\mathrm{Var}_{\mathcal{D}}(u) = \left\{ w \in V \mid \alpha_u(x) = w \text{ for a normal variable } x \in \mathrm{var}(t) \subseteq \mathrm{var}(t') \right\} . \tag{4.9}$$

  The send clause $\psi \in \Psi$ may either be a send clause occurring in the stage theory of the protocol, i. e., $\psi = \varphi$ for a clause $\varphi \in \Phi_f$, or it may be an *instance* of such a send clause $\varphi \in \Phi_f$ with $\psi \neq \varphi$. In both cases, let $\varphi = [\ldots, p(\hat{t}', \kappa) \to \mathtt{I}^e(\hat{t})]$. Let α'_u and α''_u be two embeddings with $\alpha'_u(u) = \hat{t}$ and $\alpha''_u(v) = \hat{t}'$. We define

$$\mathrm{Var}^\star_{\mathcal{D}}(u) = \left\{ w \in V \mid \alpha'_u(x) = w \text{ or } \alpha''_u(x) = w \text{ for } x \in \mathrm{var}_X(\hat{t}) \subseteq \mathrm{var}(\hat{t}') \right\} . \tag{4.10}$$

  Note that we may have two nodes $\alpha'_u(x) \neq \alpha''_u(x)$ for a normal variable $x \in \mathrm{var}_X(\hat{t})$, while there is only one node $\alpha_u(x)$ for each $x \in \mathrm{var}_X(t)$. Also note that we have $\mathrm{Var}^\star_{\mathcal{D}}(u) = \mathrm{Var}_{\mathcal{D}}(u)$ if $\psi = \varphi$ is not an instance.

For all sets and relations defined here with an index $\mathcal{D}$, we will omit the index if it is clear from the context which ADAG we refer to. ◁

4.1.2 Properties of ADAGs

To understand the proofs in the following sections it is important to point out some general properties of the structure of an ADAG $\mathcal{D}$. Therefore, we will present and prove the following four lemmas.

Lemma 4.2 (Structure of β-Paths)**.** *Let $v \in V$ be a node, let $p \in \delta(V) \cap (\widetilde{R} \cup \widetilde{\mathtt{I}})$ be a predicate symbol and $i \in \gamma^\star(v, p)$ be the index of a register sequence of v and p. Let*

$$\vec{v} = (v_1, p_1, i_1) \rightsquigarrow (v_2, p_2, i_2) \rightsquigarrow \cdots \rightsquigarrow (v_m, p_m, i_m) \tag{4.11}$$

be a β-path of maximal length through (v, p, i), i. e., a path with $(v, p, i) = (v_j, p_j, i_j)$ for a $j \in \{1, \dots, m\}$, and such that there is no (v', p', i') with $(v', p', i') \rightsquigarrow (v_1, p_1, i_1)$ or $(v_m, p_m, i_m) \rightsquigarrow (v', p', i')$.

Then we can assume the following:

1. *The subpath $(v_1, p_1, i_1) \rightsquigarrow^* (v_j, p_j, i_j)$ is uniquely determined.*
2. *$(v_1, p_1, i_1) = (\alpha(s_n), r_n^{(n,0)}, 0)$ for an $n \in \{1, \dots, N\}$,*
3. *$p_1, \dots, p_{m-1} \in \widetilde{R}$, i. e., besides the last step, the path consists of push clause applications,*
4. *$v_1 \to v_2 \to \cdots \to v_{m-1}$, i. e., besides the last step, the β-path moves along the normal edges of the ADAG, and*
5. *$(v_m, p_m, i_m) = (u, \mathtt{I}^e, 0)$ for a goal u and a stage e, and*
6. *if we have $\mathrm{Goals}_{\mathcal{D}}(v) = \emptyset$ for a node v, we also know $\delta(v) \cap \widetilde{R} = \emptyset$.*

In other words, each β-path of maximal length begins at a push symbol r_n in a node corresponding to a term s_n, then it follows the normal edges of the DAG and the push symbols, moving from a node to one of its children, and finally it »jumps« to a goal by applying a send clause.

In terms of Horn proofs this only reflects the fact that each fact containing a push symbol was justified either by assumption (and the only assumption was $r_n(s_n, \kappa^{(n)})$) or through construction by a flat push clause, and the fact is only necessary if it is a prerequisite for a send symbol.

Proof. Let v_j, p_j, i_j and $\vec{v}$ be as described in the lemma. We will prove each assumption made in the lemma.

1. The subpath $(v_1, p_1, i_1) \rightsquigarrow^* (v_j, p_j, i_j)$ is uniquely determined because we can construct it directly: We distinguish between two cases, depending on which condition v and p fulfill in $\mathcal{D}$: If (ADAG iii a) holds for v_j and p_j, we know $j = 1$ by definition of the relation $\rightsquigarrow$. Otherwise, either condition (ADAG iii c) or condition (ADAG iii d) holds for v and p, so because β is a function there is a unique triple $(v_{j-1}, p_{j-1}, i_{j-1})$ with $(v_{j-1}, p_{j-1}, i_{j-1}) \rightsquigarrow (v_j, p_j, i_j)$. Then, by applying this construction recursively on $(v_{j-1}, p_{j-1}, i_{j-1})$, we get a path as assumed. This recursion terminates as we have a finite DAG.
2. The proof of the first condition also shows that condition (ADAG iii a) holds for v_1 and p_1, so we know $v_1 = \alpha(s_n)$, $p_1 = r_n^{(n,0)}$ and $i_n = 0$ for an $n \in \{1, \dots, N\}$.

3. If we have $(v, p, i) \rightsquigarrow (v', p', i') \rightsquigarrow (v'', p'', i'')$, we know that condition (ADAG iii d) cannot hold for v' and p' because $p' \in \widetilde{\mathtt{I}}$ cannot be used as justification for any p'', i. e., we cannot have $\beta(v'', p'', i'') = (v', p', i', \varphi)$ for any φ. By definition of $\rightsquigarrow$, condition (ADAG iii c) holds for v' and p', and so we have $p' \in \widetilde{R}$. Applying this argumentation on all nodes $v \in \{v_1, \ldots, v_{m-2}\}$ and $v'' \in \{v_3, \ldots, v_m\}$ proves the third condition.

4. Here we can use the same argumentation as for the previous condition – knowing that (ADAG iii c) holds for v' and p' leads to $v \to v'$.

5. The proof for this condition is the most technical one: There may be ADAGs where this condition does not hold, but then we can construct a very similar ADAG which only has less register sequences and perhaps a smaller predicate symbol labeling, but for which this condition holds.

   Take a look at $v = v_j$, and assume that there is no goal u with $(v, p, i) \rightsquigarrow^* (u, \mathtt{I})$. We either have $j \neq m$, then condition 3 we have just proven leads to $p_j \in \widetilde{R}$, or we have $j = m$, in which case we also have $p_j \in \widetilde{R}$ because otherwise v_j would be a goal and we could select $u = v_j$.

   We will now remove the register sequence which is indexed by i_j and all of its descendants on all β-paths by restricting γ, β, and λ in the following way:

   For all nodes $v' \in V$, predicate symbols $p' \in \delta(v')$ and all indices $i' \in \gamma^\star(v', p')$ such that we do not have $(v, p, i) \rightsquigarrow^* (v', p', i')$, we set

$$\gamma_1(v', p', i') = \gamma(v', p', i') \,, \tag{4.12}$$

$$\beta_1(v', p', i') = \beta(v', p', i') \,, \tag{4.13}$$

$$\lambda_1(\mathtt{c}) = \lambda(\mathtt{c}) \;\text{ if } \lambda(\mathtt{c}) = (v', p', i', z) \text{ for a } z \in \{1, \ldots, Z\} \,. \tag{4.14}$$

   Otherwise γ_1, β_1, and λ_1 shall be undefined, i. e., $i' \notin \gamma_1^\star(v', p')$.

   We may have created a structure where a predicate symbol $\bar{p}$ for a node $\bar{v}$ does not have any register sequences left, $\gamma_1^\star(\bar{v}, \bar{p}) = \emptyset$, which would contradict the definition of an ADAG in condition (ADAG iii c). But then, this predicate symbol $\bar{p}$ can clearly not lead to any goals, so we can also remove it.

   As $\bar{p}$ may lead to other push symbols, we do not only remove $\bar{p}$, but also all of its descendants on β-paths: We set

$$\delta_2(v') = \left\{ p' \in \delta(v') \;\middle|\; (\bar{v}, \bar{p}) \not\rightsquigarrow^* (v', p') \right\} \,. \tag{4.15}$$

Now for all nodes $v' \in V$, predicate symbols $p' \in \delta_2(v')$ and all indices $i' \in \gamma_1^\star(v', p')$ we set

$$\gamma_2(v', p', i') = \gamma_1(v', p', i') \,, \tag{4.16}$$
$$\beta_2(v', p', i') = \beta_1(v', p', i') \,, \tag{4.17}$$
$$\lambda_2(\mathtt{c}) = \lambda_1(\mathtt{c}) \;\text{ if } \lambda_1(\mathtt{c}) = (v', p', i', z) \text{ for a } z \in \{1, \ldots, Z\} \,. \tag{4.18}$$

We will show that the result $\mathcal{D}' = (D, \Psi, \widehat{\Gamma}, \alpha, \beta_2, \gamma_2, \delta_2, \lambda_2)$ is an ADAG. Then, by repeating this process for all indices i and all predicate symbols p we get an ADAG $\mathcal{D}''$ which fulfills the fifth condition of the lemma.

We will show that we actually constructed an ADAG: As we only restricted the domain of the functions, all the components of the ADAG are still well-defined. The ADAG conditions (ADAG i a) to (ADAG ii) do clearly still hold, as we modified no I-symbols. For each node v and each symbol $p \in \delta'(v)$, we know that one of the four conditions (ADAG iii a) to (ADAG iii d) holds in $\mathcal{D}$. Then this condition will still hold in $\mathcal{D}'$: The cases (ADAG iii a) and (ADAG iii b) were not modified.

In case (ADAG iii c) we know because of $p \in \delta(v)$ that $(v, p, i) \rightsquigarrow^* (u, \mathtt{I})$ for an index i and a goal u; otherwise, we would have removed p from $\delta(v)$. But then, we also know that $(\hat{v}, \hat{p}, \hat{i}) \rightsquigarrow^* (u, \mathtt{I})$ for $\beta(v, p, i) = (\hat{v}, \hat{p}, \hat{i}, \varphi)$, i. e., we did not remove $\hat{p}$ or $\hat{i}$ from $\hat{v}$. Hence, case (ADAG iii c) still holds.

The last case, (ADAG iii d), holds analogously – if v is a goal with $\beta(v, \mathtt{I}) = (\hat{v}, \hat{p}, \hat{i}, \varphi)$, we know that $(\hat{v}, \hat{p}, \hat{i}) \rightsquigarrow (u, \mathtt{I})$, so we did not remove $\hat{p}$ from $\delta'(\hat{v})$ or $\hat{i}$ from $\gamma^\star(\hat{v}, \hat{p})$. This proves that by this transformation we get an ADAG.

6. This is a direct consequence of the fifth condition easily seen by contraposition: If $p \in \gamma(v) \cap \widetilde{R}$, then we have at least one index $i \in \gamma^\star(v, p)$ and can apply the lemma's fifth condition on (v, p, i), showing $\text{Goals}_{\mathcal{D}}(v) \neq \emptyset$.

This concludes the proof of our lemma, which will be used in the following proofs. □

Lemma 4.3 (Restriction of Nodes)**.** *For a protocol $\mathcal{P}$ we can assume that an ADAG for $\mathcal{P}$ contains only nodes w for which $\alpha(t) \rightarrow^* w$ holds for a term $t \in T_{\mathcal{P}}$.*

Proof. Let W be the set of nodes of an ADAG $\mathcal{D}$ such that $w \in W$ is no descendant of $\alpha(t)$ for any $t \in T_{\mathcal{P}}$. Furthermore, let $w \in W$ be a node without a parent – we can select such a node as any parent of w would also be in W.

We can remove this node without damaging the ADAG: First, w is not necessary to fulfill the ADAG conditions (ADAG i a) to (ADAG i c). For (ADAG ii) and (ADAG iii a) we have nothing to show. Then, it cannot be any v_j in condition (ADAG iii b) as w has no parents. Furthermore, by the previous lemma we know that $\delta(v)$ cannot contain any push symbols, as it would have to be descendant of a node $\alpha(t_n)$ by the previous lemma. Thus, it cannot be used as a justification in the witness function of (ADAG iii c) and (ADAG iii d).

Hence, if we define $\mathcal{D}'$ to be the same as $\mathcal{D}$ just with removing w from the set of nodes (and possibly from the functions), all ADAG conditions do still hold for $\mathcal{D}'$. We can repeat this procedure until we reach an ADAG $\mathcal{D}''$ for which we have $W = \emptyset$. □

Lemma 4.4 (Nodes of Variables)**.** *Let u be an n-goal of an ADAG $\mathcal{D}$. Then we have* $\mathrm{Var}_{\mathcal{D}}(u) \subseteq \mathrm{Steps}_{\mathcal{D}}(\leq n)$.

From the protocol point of view, this is plausible as it only states that the terms sent in a step n have to be known to the principals in that step or an earlier step than n.

This lemma also states that if d is the maximal depth of a term occurring in a send clause in the stage theory Ψ of $\mathcal{D}$, than each path of length greater than d from u or some of its descendants in the DAG will lead to a node in $\mathrm{Steps}_{\mathcal{D}}(\leq n)$.

Proof. Let u be an n-goal of an ADAG $\mathcal{D}$, and let $v \in \mathrm{Var}_{\mathcal{D}}(u)$. Then by the embedding condition in (ADAG iii d) we know that v is not only a descendant of u, but also of a node w with $\beta(u, \mathtt{I}) = (w, r^e, i, \varphi)$. We also know $r^e \in \delta(w)$ for a push symbol $r^e \in \widetilde{R}$. By looking at all possible send clauses we know $e \leq n$. Then Lemma 4.2 states there is a node $\hat{w} = \alpha(s_m)$ for some m with $\hat{w} \to^* w$. Again, we know $m \leq e$. But then, we have $m \leq n$, leading to $v \in \mathrm{Steps}_{\mathcal{D}}(\leq m) \subseteq \mathrm{Steps}_{\mathcal{D}}(\leq n)$. □

Lemma 4.5. *Let v be a node of an ADAG $\mathcal{D}$ with $v \in \mathrm{Steps}_{\mathcal{D}}(\leq n)$, but $v \notin \mathrm{Static}^{+}_{\mathcal{D}}$. Then there is an index $i \leq n$ with $\alpha(t_i) \to^* v$.*

Proof. Let v be a node of an ADAG $\mathcal{D}$. Then because of $v \in \mathrm{Steps}_{\mathcal{D}}(\leq n)$ there is a term $t \in \{\dot{\varsigma}, t_1, s_1, \ldots, t_n, s_n\}$ with $\alpha(t) \to^* v$. If $t = \dot{\varsigma}$, we would have $\alpha(\dot{\varsigma}) \to^0 v$, so $v \in \mathrm{Static}^{+}_{\mathcal{D}}$, this case is not possible. If $t = t_i$ for an $i \leq n$, there is nothing to show.

The only case left to consider is $t = s_i$ for an $i \leq n$. Then there is a path in the DAG of the form $\alpha(s_i) \to w_1 \to \cdots \to w_m \to v$. As $v \notin \mathrm{Static}^{+}_{\mathcal{D}}$, this path is longer than the depth of s_i, which means there is a node w_j on the path with $w_j = \alpha(x)$ for a variable $x \in \mathrm{sub}(s_i)$. But

then by definition of a protocol step we know that $x \in \mathrm{sub}(t_{i'})$ for an $i' \leq i$, and thus, there is a path $\alpha(t_{i'}) \to u_1 \to \cdots \to u_{m'} \to w_j$. Now we have a path

$$\alpha(t_{i'}) \to u_1 \to \cdots \to u_{m'} \to w_j \to w_{j+1} \to \cdots \to w_m \to v\,. \tag{4.19}$$

This shows $\alpha(t_{i'}) \to^* v$. □

4.2 Constructing Simple ADAGs

We now define *simple* ADAGs. Simple means that a goal with predicate symbol $\mathtt{I}^{e'}$ has no descendants in $\mathrm{Static}^{+}_{\mathcal{D}}$ with a predicate symbol $\mathtt{I}^{e}$ for any $e \gg e'$. The main idea is that these predicates are not necessary:

If a goal u with stage e is descendent of a goal u' with stage $e' \ll e$, then we cannot use the predicate symbol $\mathtt{I}^{e}$ to show that $\mathtt{I}^{e'}$ holds, as the intruder's knowledge in stage e cannot be used to show his knowledge in stage $e' \ll e$.

But there may be another ancestor of u in the DAG which needs the predicate symbol $\mathtt{I}^{e}$. Therefore, we just copy the node u to the node $\hat{u}$, leaving the original node as a descendent of v, but deleting it's symbol $\mathtt{I}^{e}$, and making the new copy $\hat{u}$ (which is annotated with the symbol $\mathtt{I}^{e}$) a descendent of all ancestors of u that need the symbol $\mathtt{I}^{e}$.

Definition 4.6. The ADAG $\mathcal{D}$ is called *simple* if for every $v \notin \mathrm{Static}^{+}_{\mathcal{D}}$ that is a descendent of $u \in \mathrm{Goals}_{\mathcal{D}}(n)$, the node v is not in $\mathrm{Goals}_{\mathcal{D}}(m)$ for any $m > n$. ◁

As we copy some nodes, there may be a situation where the definition of an ADAG no longer holds: Assume that we duplicate a node v which is in the set $\mathrm{Var}_{\mathcal{D}}(u)$ of a goal u (with $\beta(u, \mathtt{I}) = (u', i', \varphi)$). Further assume that the copy, call it $\hat{v}$, will only be a descendant of u (not u'), while v is only a descendant of u' (and not of u). Now the embedding condition in (ADAG iii d) is violated, i. e., a variable in t and t' is no longer mapped to the same node v. We will repair this by using *instances* of send clauses, as later shown in the definition of β_n. For this we use the following definition:

Definition 4.7. Let Φ_f be a stage theory for a protocol with N steps. Then $\widehat{\Phi}_f$ is the same stage theory unified with all instances φ' of send clauses $\varphi \in \Phi_f$, where the depth of φ' is not greater than $|\Phi| \cdot (N+1)$. ◁

It is easy to see that for a protocol $\mathcal{P} = (P, \Phi)$ the size of such an instantiated stage theory $\widehat{\Phi}_f$ is exponential in terms of the size of $\mathcal{P}$, as the maximal depth equals $|\Phi| \cdot (N+1) < |\mathcal{P}|^2$, i. e., is polynomial in terms of the protocol size.

Theorem 4.8 (Simple ADAGs)**.** *Let $\mathcal{P}$ be a protocol. If $\mathcal{D} = (D, \Phi_f, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ is an ADAG for a protocol $\mathcal{P} = (P, \Phi)$ and (π, f), then there exists a simple ADAG $\mathcal{D}'$ for $\mathcal{P}$ and (π, f) with $\mathcal{D}' = (D', \widehat{\Phi}_f, \widehat{\Gamma}, \alpha', \beta', \gamma', \delta', \lambda')$.*

4.2.1 Proof of Theorem 4.8

Proof. In order to prove this theorem, we will construct a simple ADAG $\mathcal{D}_N$ from an ADAG $\mathcal{D}_0$ step-by-step: We will construct a sequence $[\mathcal{D}_0, \dots, \mathcal{D}_N]$ of ADAGs where the following condition will hold for each ADAGs $\mathcal{D}_n$ in the sequence: For every $v \notin \text{Static}^+_{\mathcal{D}_n}$ that is a descendent of $u \in \text{Goals}_{\mathcal{D}_n}(m)$ for an $m \leq n$, the node v is not in $\text{Goals}_{\mathcal{D}_n}(m')$ for any $m' > m$. In short:

$$\begin{aligned} &\text{For all } v \notin \text{Static}^+_{\mathcal{D}_n} \\ &\quad \text{with } \text{Goals}_{\mathcal{D}_n}(m) \to^* v \text{ for an } m \leq n \\ &\text{we have } v \notin \text{Goals}_{\mathcal{D}_n}(>m)\ . \end{aligned} \tag{4.20 a}$$

We will later show that this condition holds for $n = N$, so we know that $\mathcal{D}_N$ is simple. We add another two conditions that will ease the argumentation:

Let $u \in \text{Goals}_{\mathcal{D}_n}(m)$ for an $m > n$. For all $v \in \text{Var}_{\mathcal{D}_n}(u)$ we either have $v \in \text{Var}^\star_{\mathcal{D}_n}(u)$, i. e., the variable that v corresponds to was not instantiated, or we have $v \in \text{Steps}_{\mathcal{D}_n}(\leq n)$. In short:

$$\begin{aligned} &\text{For all } u \in \text{Goals}_{\mathcal{D}_n}(>n) \\ &\quad \text{for all } v \in \text{Var}_{\mathcal{D}_n}(u) \setminus \text{Var}^\star_{\mathcal{D}_n}(u), \\ &\text{we have } v \in \text{Steps}_{\mathcal{D}_n}(\leq n)\ . \end{aligned} \tag{4.20 b}$$

And finally we need to state that the depth of send clauses used in the witness function β_n is bounded:

$$\begin{aligned} &\text{For all } \varphi = [\dots, p'(t', \kappa) \to p(t)] \in \widehat{\Phi}_f \\ &\quad \text{with } \beta_n(v, p, i) = (v', i', \varphi) \text{ for some } v, v', p, i, i' \\ &\text{we have } \text{depth}(t), \text{depth}(t') \leq |\mathcal{P}| \cdot (n+1)\ . \end{aligned} \tag{4.20 c}$$

For $\mathcal{D}_0$ all three conditions hold; the first one because there is no m-goal for $m \leq 0$; the second one because $\mathcal{D}_0$ uses no instances, so $\mathrm{Var}_{\mathcal{D}_n}(u) = \mathrm{Var}^\star_{\mathcal{D}_n}(u)$ for all goals u; and the last one, too, because $\mathcal{D}_0$ uses no instances, so the depth of terms in send clauses is smaller than $|\mathcal{P}|$.

Now let $n \in \{1, \ldots, N-1\}$. Let $\mathcal{D}_{n-1} = (D_{n-1}, \widehat{\Phi}_f, \widehat{\Gamma}, \alpha, \beta_{n-1}, \gamma_{n-1}, \delta_{n-1}, \lambda_{n-1})$ be an ADAG for $\mathcal{P}$ and (π, f), with V_{n-1} being the set of nodes in D_{n-1}. Assume that the inductive hypothesis – i. e., conditions (4.20 a) to (4.20 c) – holds for $\mathcal{D}_{n-1}$. We will now define $\mathcal{D}_n = (D_n, \widehat{\Phi}_f, \widehat{\Gamma}, \alpha, \beta_n, \gamma_n, \delta_n, \lambda_n)$, with V_n being the set of nodes in D_n. We will then show that all three conditions also hold for $\mathcal{D}_n$.

As explained above, the main step in this construction is to duplicate some nodes in V_{n-1}. We define the following set $W \subseteq V_{n-1}$ of nodes that will be duplicated. These are all nodes that are only belonging to protocol steps greater than n, and that are a descendant of an n-goal, but are no n-goal itself:

$$\begin{aligned} W = \{ w \in V_{n-1} \mid\ & \mathrm{Goals}_{\mathcal{D}_{n-1}}(n) \rightarrow^* w, \\ & w \notin \mathrm{Goals}_{\mathcal{D}_{n-1}}(n), \\ & w \notin \mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq n) \} \ . \end{aligned} \tag{4.21}$$

There may be some static nodes belonging to W, and we do not want to manipulate them, hence, we define the set $W_S \subseteq W$ of static nodes and their descendants. Formally, let W_S be the smallest set for which the following fixed point equation holds:

$$W_S = \left\{ w \in W \mid\ w \in \mathrm{Static}_{\mathcal{D}_{n-1}} \text{ or } W_S \rightarrow w \right\} \ . \tag{4.22}$$

We know $W_S \subseteq \mathrm{Static}^+_{\mathcal{D}_{n-1}}$: Each node $w \in W_S$ is either in $\mathrm{Static}_{\mathcal{D}_{n-1}}$, or it is a descendent of a node $\hat{w} \in \mathrm{Static}_{\mathcal{D}_{n-1}}$. By $W_S \subseteq W$ we know that in the latter case, $\hat{w}$ and w are the descendants of an n-goal u. But on the other hand both $\hat{w}$ and w are not in $\mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq n)$ (as they are in W). By Lemma 4.4 we know that the length of the path $\hat{w} \rightarrow^* w$ is bounded by $|\mathcal{P}|^2$ as that is the maximal depth of a send clause in $\widehat{\Phi}_f$. Hence, we know that w is in $\mathrm{Static}^+_{\mathcal{D}_{n-1}}$. In fact, the bound $|\mathcal{P}|^2$ in the definition of $\mathrm{Static}^+_{\mathcal{D}_{n-1}}$ originates in this step.

Now we are able to construct the ADAG $\mathcal{D}_n = (D_n, \widehat{\Phi}_f, \widehat{\Gamma}, \alpha, \beta_n, \gamma_n, \delta_n, \lambda_n)$, i. e., we will define the components of $\mathcal{D}_n$.

The Term DAG D_n

We assume that we have $D_{n-1} = (V_{n-1}, F_{n-1}, \mu_{n-1})$, and we will construct the term DAG $D_n = (V_n, F_n, \mu_n)$.

Let $W^{\triangleright}$ be a set of new nodes, i. e., $W^{\triangleright} \cap V_{n-1} = \emptyset$ with a bijection $f\colon W \to W^{\triangleright}$; then set $V_n = V_{n-1} \cup W^{\triangleright}$. Now each node $w \in W \subseteq V_{n-1}$ is split into two nodes w and $f(w) \in V_n$. As we need the corresponding nodes, we define two functions:

$$\triangleright\colon V_{n-1} \to V_n \qquad v^{\triangleright} = \begin{cases} f(v), & \text{if } v \in W\,, \\ v, & \text{if } v \notin W\,, \end{cases} \tag{4.23}$$

$$\triangleleft\colon V_n \to V_{n-1} \qquad v^{\triangleleft} = \begin{cases} f^{-1}(v), & \text{if } v \in W^{\triangleright}\,, \\ v, & \text{if } v \notin W^{\triangleright}\,. \end{cases} \tag{4.24}$$

Now we define the set of edges F_n by the following conditions for each $(v, w, l) \in F_{n-1}$:

$$(v^{\triangleright},\ w^{\triangleright},\ l) \in F_n \text{ if } v \in W\,, \tag{4.25 a}$$

$$\text{and either } (v,\ w^{\triangleright},\ l) \in F_n \text{ if } v \notin W, w \in W, v \notin \mathrm{Goals}_{\mathcal{D}_{n-1}}(\leq n) \tag{4.25 b}$$

$$\text{or } (v,\ w,\ l) \in F_n \text{ otherwise.} \tag{4.25 c}$$

The labeling function μ_n is defined by $\mu_n(v) = \mu_{n-1}(v^{\triangleleft})$. Note that we did not modify the nodes, edges or labeling in $\mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq n)$, so we have $\mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq n) = \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$.

The Register Sequence Function γ_n and the Freshness Function λ_n

The register sequences assigned to a node and a push symbol will just follow the paths on which they were constructed. Hence, take a node $v \in V_{n-1}$ and a push symbol $r \in \delta_{n-1}(v) \cap \widetilde{R}$. Now for each $i \in \gamma^{\star}_{n-1}(v, r)$ we set define the node $(v, r, i)^{\triangleright} \in V_n$ to be either v or $v^{\triangleright}$:

We will set $(v, r, i)^{\triangleright} = v^{\triangleright}$ if there are nodes $\hat{v}, w \in V_{n-1}$ with $\hat{v} \rightsquigarrow_{\mathcal{D}_{n-1}} w \rightsquigarrow^{*}_{\mathcal{D}_{n-1}} (v, r, i)$ and with $\hat{v} \notin W$, but $w \in W$ such that (4.25 b) holds for $\hat{v}$ and w, i. e., we have $\hat{v} \to_{\mathcal{D}_n} w^{\triangleright}$. Otherwise we set $(v, r, i)^{\triangleright} = v$.

Now for each $v \in V_{n-1}$, each $r \in \delta_{n-1}(v) \cap \widetilde{R}$ and each $i \in \gamma^{\star}_{n-1}(v, r)$ we define:

$$\gamma_n((v, r, i)^{\triangleright},\ r, i) = \gamma_{n-1}(v, r, i)\,. \tag{4.26}$$

Accordingly, for each $\mathtt{c} \in \widehat{\Gamma}$ we set:

$$\lambda_n(\mathtt{c}) = ((v,r,i)^{\triangleright},\ r,i,z) \qquad \text{for } \lambda_{n-1}(\mathtt{c}) = (v,r,i,z)\ . \tag{4.27}$$

The Predicate Symbol Labeling Function δ_n

Let $v \in V_{n-1}$ be a node. Let $\delta_Q = \delta_{n-1}(v) \cap Q$, as well as $\delta_R = \delta_{n-1}(v) \cap \widetilde{R}$, and $\delta_{\mathtt{I}} = \delta_{n-1}(v) \cap \widetilde{\mathtt{I}^+}$. Then we have $\delta_{n-1}(v) = \delta_Q \uplus \delta_R \uplus \delta_{\mathtt{I}}$. Now we have to assign these symbols to v and $v^{\triangleright}$ if the latter exists; so we will have $\delta_{n-1}(v) = \delta_n(v) \cup \delta_n(v^{\triangleright})$.

Hence, for now assume that $v \in W$. The pop symbols (except for $q_{\mathtt{I}\downarrow}$) can just be copied to both v and $v^{\triangleright}$. The $\mathtt{I}$-symbols we want to remove from v are all the symbols $\mathtt{I}^e$ with a stage that is greater than n, so we define $\delta_{\mathtt{I}}^{\gg} = \left\{ \mathtt{I}^e \text{ or } q^e_{\mathtt{I}\downarrow} \in \delta_{\mathtt{I}} \;\middle|\; e \gg n \right\}$. Finally we define the sets $\delta_R(v), \delta_R(v^{\triangleright}) \subseteq \delta_R$ in the following way:

$$\delta_R(v) = \left\{ r \in \delta_R \mid (v,r,i)^{\triangleright} = v \quad \text{for an index } i \in \gamma^{\star}_{n-1}(v,r) \right\}\ , \tag{4.28}$$

$$\delta_R(v^{\triangleright}) = \left\{ r \in \delta_R \mid (v,r,i)^{\triangleright} = v^{\triangleright} \text{ for an index } i \in \gamma^{\star}_{n-1}(v,r) \right\}\ . \tag{4.29}$$

Now we can define δ_n, so for each $v \in V_{n-1}$ we set:

$$\delta_n(v) = \delta_{n-1}(v) \qquad \text{if } v \notin W\ , \tag{4.30 a}$$

$$\delta_n(v) = \delta_Q \cup \delta_R(v) \cup \begin{cases} \delta_{\mathtt{I}} \setminus \delta_{\mathtt{I}}^{\gg}, & \text{if } v \notin W_S \\ \delta_{\mathtt{I}}, & \text{if } v \in W_S \end{cases} \qquad \text{if } v \in W\ , \tag{4.30 b}$$

$$\delta_n(v^{\triangleright}) = \delta_Q \cup \delta_R(v^{\triangleright}) \cup \delta_{\mathtt{I}} \qquad \text{if } v \in W\ . \tag{4.30 c}$$

The Witness Function β_n

This is the most technical part of the construction. In most cases, we will use a definition similar to $\beta_n((v,p,i)^{\triangleright},\ p,i) = \beta_{n-1}(v,p,i)$. But in some cases we have to modify the witness for the application of a send clause.

Hence, we distinguish between the following cases for $v \in V_n$ and $p \in \delta_n(v)$ if neither (ADAG iii a) nor (ADAG iii b) applies for $v^{\triangleleft}$ and p in $\mathcal{D}_{n-1}$:

1. For $p \in \delta_n(v) \cap \widetilde{R}$ we define

$$\beta_n(v,p,i) = ((v',p',i')^{\triangleright},\ p',i',\varphi) \qquad \text{for } \beta_{n-1}(v^{\triangleleft},\ p,i) = (v',p',i',\varphi)\ . \tag{4.31 a}$$

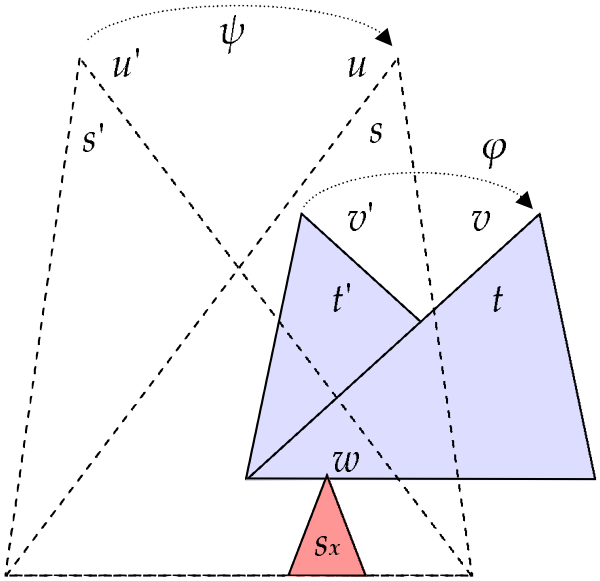

Figure 4.1: Illustration of case 2. c) of the definition of β_n in the proof of Theorem 4.8

2. For $p \in \delta_n(v) \cap \widetilde{\mathtt{I}}$ we have the following cases:

   a) If we have $\mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W = \emptyset$, or

   b) if we have $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \subseteq \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$, we do nearly the same as in (4.31 a):

$$\beta_n(v,p,0) = ((v',p',i')^{\triangleright}_{?},\ p',i',\varphi) \quad \text{for } \beta_{n-1}(v^{\triangleleft}_{?},\ p,0) = (v',p',i',\varphi)\,. \tag{4.31 b}$$

   c) Otherwise, we have $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \nsubseteq \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$, but we know that $v^{\triangleleft}$ is a goal (let e be the stage with $\mathtt{I}^e \in \delta_{n-1}(v^{\triangleleft})$), so by Lemma 4.4 we have $e \gg n$. But then by (4.20 b) we know $\beta_{n-1}(v^{\triangleleft}_{?},\ \mathtt{I}) = (v',p',i',\varphi)$ for a clause $\varphi \in \Phi_f$ with $\varphi = [\ldots, p'(t',\kappa) \to \mathtt{I}^e(t)]$.

   As case a) did not apply, we know $\mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W \neq \emptyset$, so we know that a node $w \in \mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft})$ is in W, i. e., a descendant of an n-goal u. We will define a substitution σ which maps each regular variable x to a term if and only if x is corresponding to such a node $w \in W$.

   Hence, we need the embedding θ_v which embeds $(\sigma_\kappa(t), t')$ to (v, v') as defined in (ADAG iii d) for $\mathcal{D}_{n-1}$. Now we define σ to be a substitution, the domain of σ will be the set $X_\sigma = \{x \in \mathrm{var}_X(t) \mid \theta_v(x) \in W\}$. Hence, for each $x \in X_\sigma$ we do the following steps, as illustrated in Figure 4.1:

Let u be an n-goal such that $w = \theta_v(x)$ is a descendant of u in D_{n-1}. Now let ψ be a send clause such that $\beta_{n-1}(u, \mathtt{I}) = (u', r', i', \psi)$. We assume that $\psi = [\dots r'(s', \kappa') \to \mathtt{I}^{e'}(s)]$, so let θ_u be the embedding which embeds $(\sigma_{\kappa'}(s), s')$ to (u, u') as defined in (ADAG iii d) for $\mathcal{D}_{n-1}$.

Because of Lemma 4.4 there is a term $s_x \in \mathrm{sub}(s)$ with $\theta_u(s_x) = w$: We know that u is an n-goal, w is a descendant of u, and $w \notin \mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq i)$ because of the definition of W.

Hence, we now set $\sigma(x) = s_x$. We assume that $\mathrm{var}(s_x) \cap (\mathrm{var}(t) \cup \mathrm{var}(\kappa)) = \emptyset$, else we have to rename the variables in the intersection.

After repeating this for each $x \in X_\sigma$, we set

$$\beta_n(v, p, 0) = ((v', p', i')^{\triangleright}_{?},\ p', i', \sigma(\varphi)) \quad \text{for } \beta_{n-1}(v^{\triangleleft}_{?},\ p, 0) = (v', p', i', \varphi)\ . \tag{4.31 c}$$

Note that we may have an *anonymous* variable in the term s_x, i. e., we took s_x from $\mathrm{sub}(s)$ and not, for example, from $\mathrm{sub}(\sigma_{\kappa'}(s))$, but – as shown later – this does not raise any problems.

This completes our ADAG $\mathcal{D}_n = (D_n, \widehat{\Phi}_f, \widehat{\Gamma}, \alpha, \beta_n, \gamma_n, \delta_n, \lambda_n)$, we now have to show that this actually is an ADAG and that the three conditions (4.20 a), (4.20 b), and (4.20 c) hold.

Proof of the ADAG Conditions

We will show that $\mathcal{D}_n$ actually is an ADAG. To do so, we first have to check the type of each component.

The graph D_n is acyclic: If there is a cycle $v_1, \dots, v_m$ in D_n, we have a cycle $v_1^{\triangleleft} \dots, v_m^{\triangleleft}$ in D_{n-1}. The type of the other components is easy to check from the definition of each component; and when later proving condition (4.20 c) we will show that in the definition of β_n we used instances of send clauses that actually are in $\widehat{\Phi}_f$.

Now we show that (ADAG i a) to (ADAG iii d) do still hold.

(i) For (ADAG i a) to (ADAG i c) note that we did not alter α. Hence, for each node $v = \alpha(t)$ for a term $t \in \{t_n \mid n \in \{1, \dots, N\}\} \cup \{k \mid k \in \mathrm{dom}(f)\} \cup \{\$\}$, we know $\mathtt{I}^{e_t} \in \delta_{n-1}(v)$ for an appropriate stage e_t. We show that $\mathtt{I}^{e_t} \in \delta_n(v)$, i. e., that $\mathtt{I}$ was not removed: We know that if $v \in W$, then v is also in W_S. Thus, in the definition of δ_n we did not remove $\mathtt{I}^{e_t}$, as only (4.30 a) or the second case of (4.30 b) are possible.

(ii) For (ADAG ii) just assume that $\mathtt{I}^{e_1}, \mathtt{I}^{e_2} \in \delta_n(v)$ for $e_1 \neq e_2$, then we would have $\mathtt{I}^{e_1}, \mathtt{I}^{e_2} \in \delta_{n-1}(v^{\triangleleft})$, which contradicts the inductive hypothesis that (ADAG ii) holds for $\mathcal{D}_{n-1}$.

(iii) Let $v \in V_n$ be a node, $p \in \delta_n(v)$ be a predicate symbol. By construction and inductive hypothesis, one of the for cases (ADAG iii a) to (ADAG iii d) holds for $v^{\triangleleft}$ and $p \in \delta_{n-1}(v^{\triangleleft})$. Hence, we can distinguish between this four cases:

(iii a) We know (ADAG iii a) holds for $v^{\triangleleft}$ and p in $\mathcal{D}_{n-1}$. First assume that $v^{\triangleleft} = \alpha(\dot{\varsigma})$. As $\alpha(\dot{\varsigma}) \in \mathrm{Steps}_{\mathcal{D}}(\leq 1)$ for each ADAG $\mathcal{D}$ (the first message of the intruder can only contain terms constructed from $\dot{\varsigma}$), we know $v^{\triangleleft} \notin W$, so $v^{\triangleleft} = v$. This means $v = \alpha(\dot{\varsigma})$ and $p = \mathtt{I}^{(0,0)} \in \delta_n(v)$, so (ADAG iii a) holds for v and p.

Now assume that $v^{\triangleleft} = \alpha(s_m)$ for an $m \in \{1, \ldots, N\}$. Then we have $p = r_m^{(m,0)} \in \delta_{n-1}(v^{\triangleleft})$. We defined $(v^{\triangleleft}, r_m^{(m,0)}, 0)^{\triangleright} = v^{\triangleleft}$, so we did neither change $\gamma(v, p, 0)$ nor $\lambda(v, p, 0, z)$ for any z, so (ADAG iii a) holds for v and p.

(iii b) If (ADAG iii b) holds for $v^{\triangleleft}$ and p in $\mathcal{D}_{n-1}$, the same pop clause ψ will also be sufficient as a justification for v and p in $\mathcal{D}_n$. We will show this for multiple cases:

For the case $p \in \widetilde{Q}$, we know that the necessary pop symbols $p_1, \ldots, p_m$ in the labeling of the descendants of $v^{\triangleleft}$ were copied from $\delta_{n-1}(w)$ to both $\delta_n(w)$ and $\delta_n(w^{\triangleright})$ for each descendant w of $v^{\triangleleft}$.

Take a look at $p \in \left\{ \mathtt{I}^e, q^e_{\mathtt{I}\downarrow} \;\middle|\; e \gg n \right\}$. Let $v \in V_n, w \in V_{n-1}$ be nodes with $p \in \delta_{n-1}(v^{\triangleleft})$ and $p' \in \delta_{n-1}(w)$, such that p was produced by an intruder pop clause using p'. We know $v^{\triangleleft} \to_{\mathcal{D}_{n-1}} w$, and to show that (ADAG iii b) still holds, we have to show that $v \to_{\mathcal{D}_n} w'$ for a node $w' \in \{w, w^{\triangleright}\}$ with $p' \in \delta_n(w')$.

- If we have $v \in W^{\triangleright}$, we constructed an edge $v \to_{\mathcal{D}_n} w'$ by (4.25 a), distinguish between two cases:
  - For $w \in W$ we have $w' = w^{\triangleright}$ and $p' \in \delta_n(w^{\triangleright})$ by (4.30 c).
  - For $w \notin W$ we have $w' = w$ and $p' \in \delta_n(w)$ because of (4.30 a).
- Otherwise we have $v \notin W^{\triangleright}$, i. e., $v \in V_{n-1}$. We constructed an edge $v \to_{\mathcal{D}_n} w'$ as we applied either condition (4.25 b) or condition (4.25 c).
  - If we used (4.25 b), we know $w \in W$, then again we set $w' = w^{\triangleright}$ and have $p' \in \delta_n(w^{\triangleright})$ by (4.30 c).

– If (4.25 c) was applied, we have $v \in W$, $w \in W$ or $v \in \text{Goals}_{\mathcal{D}_{n-1}} n$.

If the latter one applies, we have nothing to show, as we will handle the case (ADAG iii d) later.

For $w \notin W$ we again set $w' = w$ and know $p' \in \delta_n(w)$ because of (4.30 a). Assume that $w \in W$.

For $v \in W$ we know $v \in W_S$, else, we would have removed p as defined in (4.30 b). But then by (4.22) we would also have $w \in W_S$ because $w \in W$ and $v \to_{\mathcal{D}_{n-1}} w$. Then for $w' = w$ we know $v \to_{\mathcal{D}_n} w$ as well as $p' \in \delta_n(w)$ because of (4.30 b).

Furthermore, we still have $v_{j_1} = v_{j_2}$ for all $j_1, j_2 \in \{1, \ldots, m\}$ with $x_{j_1} = x_{j_2}$, as conditions (4.25 a) to (4.25 c) modify all edges between two nodes in exactly the same way.

(iii c) In this case, too, we will show that (ADAG iii c) still holds. Hence, let $i \in \gamma_n^\star(v, p)$, and let $\beta_n(v, p, i) = (v', p', i', \varphi)$. We know by construction that we have $\beta_{n-1}(v^\triangleleft, p, i) = (v'^\triangleleft, p', i')$, so we have to show that v' is a father of v, that $p' \in \delta_n(v')$ and that $i' \in \gamma_n^\star(v', p')$. Then we know that the register sequences are still compatible and the freshness function still points to the right register sequences because we did modify these functions accordingly.

Distinguish between two cases:

- First, assume that $v = v^\triangleleft$, i. e., $(v, p, i)^\triangleright = v$. Then by the definition of $(v, p, i)^\triangleright$ we know that there are no nodes $\hat{v}, w \in V_{n-1}$ with $\hat{v} \rightsquigarrow_{\mathcal{D}_{n-1}} w \rightsquigarrow^*_{\mathcal{D}_{n-1}} (v, r, i)$, with $\hat{v} \notin W$, and $w \in W$ such that (4.25 b) holds for $\hat{v}$ and w. But then the same applies for (v', p', i'), leading to $(v', p', i')^\triangleright = v'$. This means $p' \in \delta_n(v')$ and $i' \in \gamma_n^\star(v', p')$, and this also means $v' \to_{\mathcal{D}_n} v$ because of condition (4.25 c) (remember that (4.25 b) did not apply).

- Now assume that $v \neq v^\triangleleft$, this means $v \in W^\triangleright$. We again use the definition of $(v^\triangleleft, p, i)^\triangleright$, but this time we know that there are nodes $\hat{v}, w \in V_{n-1}$ with $\hat{v} \rightsquigarrow_{\mathcal{D}_{n-1}} w \rightsquigarrow^*_{\mathcal{D}_{n-1}} (v, p, i)$ and with $\hat{v} \notin W$, but $w \in W$ such that (4.25 b) holds for $\hat{v}$ and w.

  We then can either have $w = v^\triangleleft$, then we know $\hat{v} = v'$ and we have $v' \to_{\mathcal{D}_n} v$ because of (4.25 b). Then $v' = \hat{v} \notin W$ also implies $p' \in \delta_n(v')$ and $i' \in \gamma_n^\star(v', p')$.

Or we have $w \neq v^{\triangleleft}$, but then we know $v' \neq v^{\triangleleft}$: There are nodes $\hat{v}, w \in V_{n-1}$ with $\hat{v} \rightsquigarrow_{\mathcal{D}_{n-1}} w \rightsquigarrow^*_{\mathcal{D}_{n-1}} (v'^{\triangleleft}_{,}\ p', i') \rightsquigarrow_{\mathcal{D}_{n-1}} (v^{\triangleleft}_{,}\ p, i)$ with $\hat{v} \notin W$, but $w \in W$ such that (4.25 b) holds for $\hat{v}$ and w. Then we know $v' \rightarrow_{\mathcal{D}_n} v$ by condition (4.25 a), and we also know $p' \in \delta_n(v')$ and $i' \in \gamma^\star_n(v', p')$ because $(v'^{\triangleleft}_{,}\ p', i')^{\triangleright} = v'$.

(iii d) In this case we know $v^{\triangleleft}$ was a goal, and we will show that v is also a goal with $p = \mathtt{I}^e$. Let $\beta_n(v, \mathtt{I}) = (v', p', i', \varphi)$ with $\varphi = [\ldots, p'(t', \kappa') \rightarrow p(t)]$. Then we know by definition of β_n that $p' \in \delta_n(v')$, and that $i' \in \gamma^\star_n(v', i')$. We only have to show that the embeddings are compatible, i. e., that $(\sigma_\kappa(t), t') \mapsto (v, v')$ for the substitution $\sigma_\kappa = \mathrm{subst}(\kappa', \gamma_n(v', p', i'))$. As we did neither modify κ' nor $\gamma_n(v', p', i')$, we will still use the same substitution σ_κ.

We can now distinguish between the cases we had when defining $\beta_n(v, \mathtt{I})$ by conditions (4.31 b) and (4.31 c).

In the first case, we have $\mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W = \emptyset$. Then we know that there are no nodes in $\mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft})$ that are duplicated, and as we know $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \subseteq \mathrm{Fix}_{\mathcal{D}_{n-1}}(v^{\triangleleft})$, all nodes of variables of v are left unchanged.

The same argumentations applies for the case $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \subseteq \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$, because this also leads to $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W = \emptyset$, so the nodes of variables of v were left unchanged.

But in the last case, we may have copied some nodes of variables of v because we cannot ensure that $\mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W$ is empty. Again, see Figure 4.1 for illustration. Take a node $w \in \mathrm{Var}_{\mathcal{D}_{n-1}}(v^{\triangleleft}) \cap W$ corresponding to the variable x. Then there is a node $w^{\triangleright} \neq w$, which could contradict the embedding condition: We may have

$$v \rightarrow w^{\triangleright}\,, \qquad v \not\rightarrow w\,, \qquad \text{but} \tag{4.32 a}$$

$$v' \not\rightarrow w^{\triangleright}\,, \qquad v' \rightarrow w\,. \tag{4.32 b}$$

But we replaced x by a term s_x, where the term s_x was taken from the set of subterms of a term s which embeds to an n-goal u. Thus, let x' be a variable in $\mathrm{sub}(s_x)$. Then there is a node w' corresponding to x' in the embedding of s to u, and by applying Lemma 4.4 to $\mathcal{D}_{n-1}$ we know that w' is in $\mathrm{Steps}_{\mathcal{D}_{n-1}}(< n)$, so we know $w' \in \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$. But then w' was not split, and there are two paths $w \rightarrow^* w'$ and $w^{\triangleright} \rightarrow^* w'$. This leads to $v \rightarrow^* w'$ and $v' \rightarrow^* w'$; thus for

this w' we have repaired the embedding such that $x' \in \mathrm{var}(t)$ and $x' \in \mathrm{var}(t')$ both embed to w'.

Proof of Condition (4.20 a)

Take a node $v \in V_n$ that is a descendant of an m-goal $u \in \mathrm{Goals}_{\mathcal{D}_n}(m)$ with $m \leq n$. Let $v \notin \mathrm{Static}^{+}_{\mathcal{D}_n}$ (and thus, $v^{\triangleleft} \notin \mathrm{Static}^{+}_{\mathcal{D}_{n-1}}$). We have to show that $v \notin \mathrm{Goals}_{\mathcal{D}_n}(> m)$, so we assume that $v \in \mathrm{Goals}_{\mathcal{D}_n}(> m)$ and show that each of the following cases leads to a contradiction:

1. If m is smaller than n, we get a contradiction with the inductive hypothesis (4.20 a) because of $v^{\triangleleft} \in \mathrm{Goals}_{\mathcal{D}_{n-1}}(> m)$ and $u^{\triangleleft} \in \mathrm{Goals}_{\mathcal{D}_{n-1}}(m)$.

2. If m equals n, we have three cases:

   a) First assume that $v \in W^{\triangleright}$: In this case there would have to be an n-goal $\hat{u} \in \mathrm{Goals}_{\mathcal{D}_n}(n) = \mathrm{Goals}_{\mathcal{D}_{n-1}}(n)$ and a path of the following kind with $w_1, \ldots, w_l \in W$ (and thus, $w_i \neq w_i^{\triangleright}$):

   $$\hat{u} \rightarrow_{\mathcal{D}_n} w_1^{\triangleright} \rightarrow_{\mathcal{D}_n} \cdots \rightarrow_{\mathcal{D}_n} w_l^{\triangleright} \rightarrow_{\mathcal{D}_n} v\,. \tag{4.33}$$

   But by (4.25 b) there is no edge $\hat{u} \rightarrow_{\mathcal{D}_n} w^{\triangleright}$, there is only an edge $\hat{u} \rightarrow_{\mathcal{D}_n} w$.

   b) Now assume that $v \in W$. As $v = v^{\triangleleft} \notin \mathrm{Static}^{+}_{\mathcal{D}_{n-1}}$, we have $v \notin W_S$. Then by the definition of δ_n (4.30 b) there is no $\mathtt{I}^e \in \delta_n(v)$ for any $e \gg n$.

   c) Finally, consider $v \notin W \cup W^{\triangleright}$, i. e., $v^{\triangleleft} \notin W$. Take look at the definition of W in (4.21) and distinguish between the different possibilities why we have $v \notin W$: We know that $v^{\triangleleft}$ cannot be an n-goal (then we would have nothing to show, see (ADAG ii)), and by assumption about u we have $u^{\triangleleft} \in \mathrm{Goals}_{\mathcal{D}_{n-1}}(n)$ and $u^{\triangleleft} \rightarrow^* v^{\triangleleft}$, so by definition of W we know $v = v^{\triangleleft} \in \mathrm{Steps}_{\mathcal{D}_{n-1}}(\leq n) = \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$.

   Lemma 4.5 states there is an $m \leq n$ with $\alpha(t_m) \rightarrow^*_{\mathcal{D}_n} v$. Let $[v_1, \ldots, v_l]$ be a path in the DAG D_n from $v_1 = \alpha(t_m)$ to a leaf v_l with $v = v_k$ for a $k \leq l$, as shown in Figure 4.2.

   By the definition of an ADAG (ADAG i a) there is a stage $e_1 \ll m$ with $\mathtt{I}^{e_1} \in \delta_n(\alpha(t_m))$. And there is a minimal index i such that v_i is a goal for a stage $e_i \leq e_1$. Now assume that v is a goal for a stage $e_v \gg n$. Then we know:

   $$e_v \gg n \geq m \gg e_1 \geq e_i\,. \tag{4.34}$$

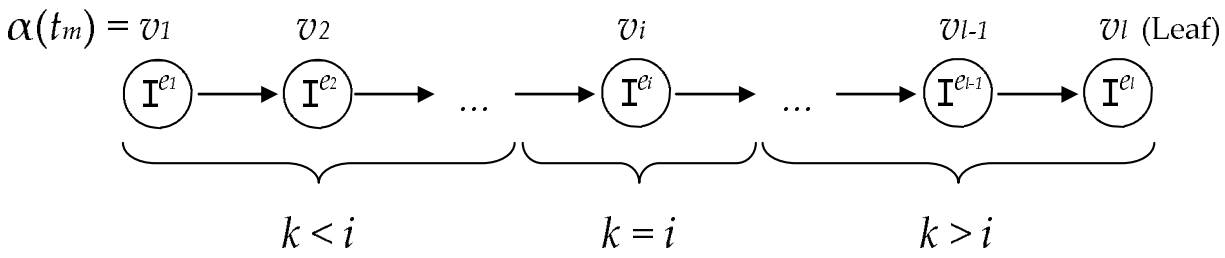

Figure 4.2: Illustration of case 2. c) of the proof of condition (4.20 a) for Theorem 4.8

Finally, we again consider three cases:

i. In the case $k < i$ note that by choice of i we have $\mathtt{I}^{e_j} \in \delta_n(v_j)$ for a stage e_j with $e_1 \geq e_j \geq e_i$ for all $j \in \{1, \ldots, i\}$. But then we would have $e_k \in \delta_n(v)$ with $e_k \leq e_1 \ll e_v$, which contradicts (ADAG ii) for v.

ii. The case $k = i$ directly contradicts (ADAG ii), as this would give $\mathtt{I}^{e_0} \in \delta_n(v_k)$ and $\mathtt{I}^{e_i} \in \delta_n(v_k)$.

iii. The case $k > i$ contradicts the inductive hypothesis (4.20 a): v would be descendant of a goal v_i with stage $e_i \ll n \ll e_0$.

Proof of Conditions (4.20 b) and (4.20 c)

Now we have to show that for every $u \in \mathrm{Goals}_{\mathcal{D}_n}(m)$ for an $m > n$ and for every $v \in \mathrm{Var}_{\mathcal{D}_n}(u) \setminus \mathrm{Var}^\star_{\mathcal{D}_n}(u)$ we have $v \in \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$. Therefore, let u be an m-goal, and let $v \in \mathrm{Var}_{\mathcal{D}_n}(u) \setminus \mathrm{Var}^\star_{\mathcal{D}_n}(u)$.

We know that v is a node that corresponds to a variable that was introduced by an instantiation during the definition of $\beta_{n'}$ for an n'. We have to show that $v \in \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$.

If $n' < n$, by inductive hypothesis (4.20 b) we already have $v \in \mathrm{Steps}_{\mathcal{D}_n}(< n)$. Now assume that $n' = n$, i. e., during the definition of $\mathcal{D}_n$ we replaced a variable x corresponding to a node w with a subterm s_x, such that there is a variable $x' \in \mathrm{var}_X(s_x)$ which v corresponds to. The term s_x was taken from the set of subterms of a term s which embeds to an n-goal u'. But then by Lemma 4.4 we know that the node corresponding to x' is in $\mathrm{Steps}_{\mathcal{D}_n}(< n)$, so we know $v \in \mathrm{Steps}_{\mathcal{D}_n}(< n) \subseteq \mathrm{Steps}_{\mathcal{D}_n}(\leq n)$.

As mentioned earlier, we still have to show that when defining β_n, we actually used instances that are in $\widehat{\Phi}_f$, i. e., that are not too deep; this is also necessary to show the third condition.

Note that we do only instantiate each variable once, so if we replaced x by s_x, we know that the node w that is corresponding to x is in $\mathrm{Var}^\star_{\mathcal{D}_n}(u)$. The resulting nodes $w'_1, w'_2, \ldots$ corresponding to the variables in s_x will then be in $\mathrm{Steps}_{\mathcal{D}_n}(< n)$. Thus, for $\mathcal{D}_{n+1}, \ldots, \mathcal{D}_N$, the nodes w'_i will not be in W, and hence, the variables occurring in s_x will not be substituted again.

Also note that each such x corresponding to a node $w \in \mathrm{Var}^\star_{\mathcal{D}_n}(u)$ is reachable from u with a path that is shorter than $|\mathcal{P}|$. We extend the length of that path by the depth of s_x, so to prove both conditions we only have to know that the depth of s_x is bounded by $|\mathcal{P}| \cdot n$, but this is the third condition (4.20 c) of the inductive hypothesis.

Now we have shown that our construction creates an ADAG $\mathcal{D}_n$ for which all three of our conditions hold. Thus, after constructing $\mathcal{D}_N$ from $\mathcal{D}_{N-1}$ in this way we get an ADAG which is simple. □

4.3 Bounding the Number of Goals

Now every ADAG can be transformed into a simple one. The next theorem states that a simple ADAG can be transformed into an ADAG with an exponentially bounded number of goals.

This transformation uses the observation that one part of the relevant information of a node v is the predicate symbols its labeled with, i. e., the value of $\delta(v)$. Hence, from the set of all goals u which have the same labeling $\delta(u)$, we can pick a specific one as a representative.

During this process we will not delete any nodes, but we will delete the predicate symbol $\mathtt{I}^e$ from some nodes, transforming them from goals to non-goals. The other components of the ADAG like the witness and the labeling function are adjusted accordingly.

In fact there are a goals we do not want to alter at all, for example, the goals in the static part of the protocol. Nevertheless, after this transformation we have an exponentially bounded number of goals without deleting any nodes.

Theorem 4.9 (Bounded Number of Goals)**.** *Let $\mathcal{P}$ be a protocol. If $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ is a simple ADAG for $\mathcal{P}$ and (π, f) where each term occurring in Ψ has at most polynomial depth in terms of the size of the protocol, then there exists an ADAG $\mathcal{D}' = (D', \Psi, \widehat{\Gamma}, \alpha, \beta', \gamma', \delta', \lambda)$ for $\mathcal{P}$ and (π, f), where $\mathrm{Goals}_{\mathcal{D}'}$ is exponentially bounded in terms of the size of $\mathcal{P}$.*

4.3.1 Proof of Theorem 4.9

Proof. This proof, too, will be done by constructing a sequence $[\mathcal{D}_{N+1}, \dots, \mathcal{D}_1]$ of ADAGs, this time starting with $\mathcal{D} = \mathcal{D}_{N+1}$ and ending with an ADAG $\mathcal{D}_1$ with a bounded number of goals. The inductive hypothesis is straightforward: For each $n \in \{1, \dots, N\}$ we will construct $\mathcal{D}_n$ from $\mathcal{D}_{n+1}$ such that the following condition holds for $\mathcal{D}_n$:

$$\text{Goals}_{\mathcal{D}_n}(\geq n) \text{ is exponentially bounded} \quad . \tag{4.35}$$

Again, this condition holds for any ADAG $\mathcal{D}_{N+1}$ we start with; and after we constructed an ADAG $\mathcal{D}_1$ for which this condition holds, we have proven the theorem.

Let $n \in \{1, \dots, N\}$. Let $\mathcal{D}_{n+1} = (D_{n+1}, \Psi, \widehat{\Gamma}, \alpha, \beta_{n+1}, \gamma_{n+1}, \delta_{n+1}, \lambda)$ be a simple ADAG with $D_{n+1} = (V, F_{n+1}, \mu)$ being the DAG. Assume as inductive hypothesis that the condition holds for $\mathcal{D}_{n+1}$. We will construct $\mathcal{D}_n = (D_n, \Psi, \widehat{\Gamma}, \alpha, \beta_n, \gamma_n, \delta_n, \lambda)$ with $D_n = (V, F_n, \mu)$, i. e., use the same stage theory, the same set of anonymous variables, the same embedding and freshness functions and even the same set of nodes. Hence, we will only define the new set of edges F_n and the new predicate labeling, register sequence and witness functions δ_n, γ_n, and β_n.

Selecting the Protected Goals

Let $G = \text{Goals}_{\mathcal{D}_{n+1}}(n)$ be the set of n-goals, we will define a subset $G^{\triangleright} \subseteq G$ of protected goals, i. e., nodes we keep as goals, whereas for all nodes v in $G^{\triangleleft} = G \setminus G^{\triangleright}$ we will later remove $\mathtt{I}^e$ from $\delta_n(v)$.

First, we will define an equivalence relation on G according to the predicate labeling of the goals: For $u_1, u_2 \in G$ let $u_1 \equiv u_2$ if $\delta_{n+1}(u_1) = \delta_{n+1}(u_2)$. We will need a representative from each equivalence class of $\equiv$, so for each class G_u for a goal u, let $u^{\equiv}$ denote a unique node such that no goal $w \in G_u$ is a descendant of $u^{\equiv}$.

We will put $u^{\equiv}$ in $G^{\triangleright}$, but some other goals need to be protected, too; so we define $G^{\triangleright}$ to

be the least subset of G such that

$$\{u^{\equiv} \mid u \in G\} \subseteq G^{\triangleright}\,, \tag{4.36 a}$$

$$\mathrm{Fix}_{\mathcal{D}_{n+1}}(v) \cap G \subseteq G^{\triangleright} \qquad \text{for all } v \in \mathrm{Goals}_{\mathcal{D}_{n+1}}(>n)\,, \tag{4.36 b}$$

$$\mathrm{Static}^{+}_{\mathcal{D}_{n+1}} \cap G \subseteq G^{\triangleright}\,, \tag{4.36 c}$$

$$\left\{u \in G \mid u \rightsquigarrow^{*}_{\mathcal{D}_{n+1}} v\right\} \subseteq G^{\triangleright} \qquad \text{for all } v \in \mathrm{Goals}_{\mathcal{D}_{n+1}}(>n)\,, \tag{4.36 d}$$

$$\left\{u \in G \mid G^{\triangleright} \rightarrow^{*}_{\mathcal{D}_{n+1}} u\right\} \subseteq G^{\triangleright}\,. \tag{4.36 e}$$

The reasons why we need to protect all these goals are hidden in the details of the proof. The first line states that for each occurring value of δ, we need to keep one representative. We need the second line to protect all the fixed parts of m-goals for $m > n$ so the embeddings in (ADAG iii d) are still compatible – we will later show that this protection is also guaranteed for $m = n$. The third line protects nodes in the static part of the DAG, which eases our argumentation in the following steps because, e. g., the property of »simple« only holds for non-static goals. Condition (4.36 d) is necessary to prevent damage to β-paths from a node to a goal: The β-paths move along the normal edges of the DAG, and we will change some edges, so by this condition we guarantee that no β-path is damaged. Finally, (4.36 e) and the definition of $u^{\equiv}$, or more precisely the fact that $u^{\equiv}$ has no descendants in its equivalence class, are necessary to guarantee that we will get an DAG. All this parts will be referred to in the proof later.

We will now start with defining our ADAG $\mathcal{D}_n$, i. e., the functions δ_n, β_n, γ_n and the term DAG D_n.

The Predicate Symbol Labeling Function δ_n

Let $v \in V$ be a node. Again, let $\delta_Q = \delta_{n+1}(v) \cap Q$, as well as $\delta_{\mathtt{I}} = \delta_{n+1}(v) \cap \widetilde{\mathtt{I}}^{+}$. We will set $\delta_R = \delta_{n+1}(v) \cap \widetilde{R}$, but also define $\delta'_R \subseteq \delta_R$ to only contain the necessary push symbols, i. e., let δ'_R be the set of all $r \in \delta_R$ with $(v, r, i) \rightsquigarrow^{*}_{\mathcal{D}_{n+1}} u$ for an index i and a goal $u \notin G^{\triangleleft}$.

Now we can define δ_n for each $v \in V$:

$$\delta_n(v) = \delta_Q \cup \delta'_R \qquad \text{if } v \in G^{\triangleleft}\,, \tag{4.37 a}$$

$$\delta_n(v) = \delta_Q \cup \delta'_R \cup \delta_{\mathtt{I}} \qquad \text{if } v \notin G^{\triangleleft}\,. \tag{4.37 b}$$

The Witness Function β_n and the Register Sequence Function γ_n

For each node $v \in V$ and $p \in \delta_n(v) \cap (\widetilde{R} \cup \widetilde{\mathtt{I}})$ we will reduce β_n and γ_n to the necessary parts.

Let $i \in \gamma^\star_{n+1}(v,p)$. If $(v,p,i) \rightsquigarrow^*_{\mathcal{D}_{n+1}} u$ for a goal $u \notin G^\triangleleft$, we will set $\beta_n(v,p,i) = \beta_{n+1}(v,p,i)$ and $\gamma_n(v,p,i) = \gamma_{n+1}(v,p,i)$; otherwise let $\beta_n(v,p,i)$ and $\gamma_n(v,p,i)$ be undefined.

The Term DAG D_n

As stated above, we do neither remove nor add any nodes. But we will modify some edges: All $\mathtt{I}$-symbols that are created by intruder pop clauses which are based on a goal $u \in G^\triangleleft$ will now be based on $u^\equiv$. But as the pop clauses follow the edges of the DAG, we have to modify the DAG:

Let W be the set of all nodes w with $\mathtt{I}^e$ or $q^e_{\mathtt{I}\downarrow} \in \delta_{n+1}(w)$, but $w \notin \mathrm{Goals}_{\mathcal{D}_{n+1}}$, i. e., we know that (ADAG iii a) or (ADAG iii b) holds for w and $\mathtt{I}^e$ or $q^e_{\mathtt{I}\downarrow}$ in $\mathcal{D}_{n+1}$, so the predicate symbol was derived by an intruder pop clause (or $w = \alpha(\dot{\varsigma})$). Now define the set of edges F_n such that for all $(v,u,l) \in F_{n+1}$ we have

$$(v, u^\equiv, l) \in F_n \qquad \text{if } v \in W \text{ and } u \in G^\triangleleft\,, \tag{4.38 a}$$

$$(v, u, \ \ l) \in F_n \qquad \text{if } v \notin W \text{ or } \quad u \notin G^\triangleleft\,. \tag{4.38 b}$$

This completes our ADAG $\mathcal{D}_n = (D_n, \Psi, \widehat{\Gamma}, \alpha, \beta_n, \gamma_n, \delta_n, \lambda)$, we now show that we constructed a simple ADAG and condition (4.35) holds.

Proof of the ADAG Conditions

We will show that $\mathcal{D}_n$ actually is an ADAG. As we kept most of the components, there is not much to show for the types – the only interesting part here is to see that D_n is acyclic. To do so, assume that there is a cycle $v_1, \ldots, v_l$ in D_n.

As D_{n+1} is acyclic, there must be an edge $v_j \to_{\mathcal{D}_n} v_{j+1}$, that is modified, i. e., in F_n but not in F_{n+1}. By (4.36 e) we know that $G^\triangleright$ is closed under $\to^*$, so no descendant of v_{j+1} is in $G^\triangleleft$, which means that (4.38 a) cannot be applied for any descendant of v_{j+1}.

Hence, $v_j \to_{\mathcal{D}_n} v_{j+1}$ is the only edge in the cycle that was modified, i. e., $v_{j+1} \to^*_{\mathcal{D}_{n+1}} v_j$. By (4.38 a) there is a node u with $u^\equiv = v_{j+1}$ and $(v_j, u, l) \in F_{n+1}$. But then we have

$$u^\equiv = v_{j+1} \to^*_{\mathcal{D}_{n+1}} v_j \to_{\mathcal{D}_{n+1}} u\,. \tag{4.39}$$

But $u^{\equiv} \to^* u$ contradicts the selection of the representative $u^{\equiv}$; so our assumption that there is a cycle must be wrong.

Now we have to check if (ADAG i a) to (ADAG iii d) do still hold.

(i) Because of (4.36 c) all nodes in the range of α are elements of $G^{\triangleright}$, so we did not modify their predicates.

(ii) We did at most remove predicates $\mathtt{I}^e$, so condition (ADAG ii) still holds.

(iii) Let $v \in V$ and $p \in \delta_n(v)$. As $\mathcal{D}_{n+1}$ is an ADAG and $\delta_n(v) \subseteq \delta_{n+1}(v)$, we know one of the four conditions (ADAG iii a) to (ADAG iii d) holds in $\mathcal{D}_{n+1}$. We will distinguish between these four cases and show that each case still holds for v and p in $\mathcal{D}_n$:

(iii a) For this case there is nothing to show.

(iii b) In this case we may have modified an edge $v \to_{\mathcal{D}_{n+1}} v_j$ to $v \to_{\mathcal{D}_n} v_j^{\equiv}$. But as $\delta_{n+1}(v_j) = \delta_{n+1}(v_j^{\equiv})$ and $\delta_{n+1}(v_j^{\equiv}) \cap \widetilde{Q} = \delta_n(v_j^{\equiv}) \cap \widetilde{Q}$, the condition (ADAG iii b) still holds for v and p.

(iii c) In this case, too, we show that (ADAG iii c) still holds, mainly because we did not change most of the components. We have to show this for each $i \in \gamma_n^\star(v,p)$, i. e., each i for which $\beta_n(v,p,i)$ is defined. Therefore, let $\beta_n(v,p,i) = \beta_{n+1}(v,p,i) = (v', p_m, i', \psi)$. During the definition of δ'_R we might have removed the predicate p_m from δ'_R for v', or we might have removed the edge(s) between v' and v, both would be a problem. The predicates $p_1, \ldots, p_{m-1}$ are pop symbols so we did not modify them.

But because $\beta_n(v,p,i)$ is defined we know that there is a goal $u \notin G^{\triangleleft}$ with $(v,p,i) \rightsquigarrow^*_{\mathcal{D}_{n+1}} u$. As we also know $(v', p_m, i') \rightsquigarrow_{\mathcal{D}_{n+1}} (v,p,i)$, we know that $(v', p_m, i') \rightsquigarrow^*_{\mathcal{D}_{n+1}} u$, and thus, we have $p_m \in \delta'_R$ for v'.

Now we might still have damaged the DAG such that v' no longer is a parent of v, but of another node $v^{\equiv}$. For this to be the case, v would have to be an n-goal in $G^{\triangleleft}$. Again look at the goal u with $(v,p,i) \rightsquigarrow^*_{\mathcal{D}_{n+1}} u$, and let e_u be the stage of u. In Lemma 4.10 we will later show that $e_u \gg n$. Then we have $v \in G^{\triangleright}$ by (4.36 d), so we have not altered the edge from v' to v.

Again, the condition $v_{j_1} = v_{j_2}$ for all $j_1, j_2 \in \{1, \ldots, m\}$ with $x_{j_1} = x_{j_2}$ still holds, as the modification of edges in (4.38 a) and (4.38 b) does not depend on l.

(iii d) This case is the most technical one. Again, we will show that (ADAG iii d) still holds. Note that for each goal $u \notin G^{\triangleleft}$ we have $(u, p, 0) \rightsquigarrow^{0}_{\mathcal{D}_{n+1}} (u, p, 0)$ for $p \in \widetilde{\mathtt{I}}$, so the witness function for v, p and 0 is defined.

Let the witness be $\beta_n(v, p, 0) = \beta_{n+1}(v, p, 0) = (v', p_m, i', \psi)$ with the send clause $\psi = [p_0(t), p_1(t'), \dots, p_m(t', \kappa') \to p(t)]$. Again, the definition of δ'_R might have removed the predicate p_m from δ'_R for v', or we might have modified edges in $\mathrm{Fix}_{\mathcal{D}_{n+1}}(v)$ (or between v' and v).

The predicate p_0 would only be removed from $\delta_n(v)$ if p was also removed from $\delta_n(v)$, which is not the case.

The predicates $p_1, \dots, p_{m-1}$ are pop symbols which we did not modify. And as v was a goal in $\mathcal{D}_{n+1}$, we set $p_m \in \delta'_R$, so we have $p_m \in \delta_m(v')$.

In $\mathcal{D}_{n+1}$, the pair $(\sigma_\kappa(t), t')$ embeds to (v, v'), hence, it also embeds in $\mathcal{D}_n$ as long as we did not modify any edge in $\mathrm{Fix}_{\mathcal{D}_{n+1}}(v)$, i. e., as long as we do not have any $u \in \mathrm{Fix}_{\mathcal{D}_{n+1}}(v)$ with $u \in G^{\triangleleft}$. Assume that there is such an u, we will show that this leads to a contradiction. First, we know that u is an n-goal as it is in $G^{\triangleleft}$ and it is either a descendant of v or of v', or both.

- If $v' \to^* u$ holds, note that $u \in G^{\triangleleft}$ means $u \notin \mathrm{Static}^{+}_{\mathcal{D}_{n+1}}$ by (4.36 c). We then either have $u \in \mathrm{Steps}_{\mathcal{D}_{n+1}}(\leq n)$ or $u \notin \mathrm{Steps}_{\mathcal{D}_{n+1}}(\leq n)$. In the first case, we use Lemma 4.11, so we have $u \notin \mathrm{Goals}_{\mathcal{D}_{n+1}}(\geq n)$, so $u \notin G^{\triangleleft}$.

  In the latter case, $u \notin \mathrm{Steps}_{\mathcal{D}_{n+1}}(\leq n)$ implies $v' \notin \mathrm{Steps}_{\mathcal{D}_{n+1}}(\leq n)$, so $v \notin \mathrm{Goals}_{\mathcal{D}_{n+1}}(\leq n)$. But then we have $v \in \mathrm{Goals}_{\mathcal{D}_{n+1}}(> n)$, which means $u \in \mathrm{Fix}_{\mathcal{D}_{n+1}}(v) \cap G \subseteq G^{\triangleright}$ by (4.36 b). This contradicts $u \in G^{\triangleleft}$.

- If $v \to^* u$ holds, again note $u \in G^{\triangleleft}$ means $u \notin \mathrm{Static}^{+}_{\mathcal{D}_{n+1}}$ by (4.36 c). As $\mathcal{D}_{n+1}$ is simple, we know $v \notin \mathrm{Goals}_{\mathcal{D}_{n+1}}(< n)$. But as above, we have $v \notin \mathrm{Goals}_{\mathcal{D}_{n+1}}(> n)$, otherwise, we would have $u \notin G^{\triangleleft}$. Hence, we have $v \in \mathrm{Goals}_{\mathcal{D}_{n+1}}(n) = G$. But the I-symbol p is still in $\delta_m(v)$, so we have $v \in G^{\triangleright}$. Then by (4.36 e) we also have $u \in G^{\triangleright}$, i. e., $u \notin G^{\triangleleft}$.

Now we know there is no node in $\mathrm{Fix}_{\mathcal{D}_{n+1}}(v) \cap G^{\triangleleft}$, so the structure of $\mathrm{Fix}_{\mathcal{D}_{n+1}}(v)$ has not changed in $\mathcal{D}_n$. Then we also know $(\sigma_\kappa(t), t') \mapsto (v, v')$.

This shows that $\mathcal{D}_n$ is still an ADAG. Note that it is also a simple ADAG: Take some nodes $v, w \in V$ with $v \to^* w$, but $v \in \mathrm{Goals}_{\mathcal{D}}(j)$ for a j and $w \notin \mathrm{Static}^{+}_{\mathcal{D}}$. We have to show that $w \notin \mathrm{Goals}_{\mathcal{D}}(> j)$.

During the definition of δ_n we did only remove symbols, so because of the assumption that $\mathcal{D}_{n+1}$ was simple this could only happen if we changed an edge on the path from v to w. This means there is an n-goal u such that $v \to^*_{\mathcal{D}_n} u$ and $u^{\equiv} \to^*_{\mathcal{D}_n} w$, and we modified the edge pointing to u to an edge pointing to $u^{\equiv}$.

But then because of the modification and condition (4.36 c) we know that $u \notin \text{Static}^+_{\mathcal{D}_n}$. Then we have $j \geq n$, otherwise the nodes v and u would contradict the simplicity of $\mathcal{D}_{n+1}$. But we also know $u^{\equiv} \to^*_{\mathcal{D}_{n+1}} w$ (remember that on each path we modified only on edge), and the simplicity of $\mathcal{D}_{n+1}$ tells us that $w \notin \text{Goals}_{\mathcal{D}}(>n)$, so $w \notin \text{Goals}_{\mathcal{D}}(>j) \subseteq \text{Goals}_{\mathcal{D}}(>n)$.

Thus, we obtained a simple ADAG, and we will now have to show that condition (4.35) holds.

Proof of Condition (4.35)

We now show that $\text{Goals}_{\mathcal{D}_n}(\geq n)$ is exponentially bounded in terms of the size of the protocol. By inductive hypothesis, $\text{Goals}_{\mathcal{D}_{n+1}}(>n)$ is exponentially bounded, and as we did only modify n-goals, we also know $\text{Goals}_{\mathcal{D}_n}(>n)$ is exponentially bounded. Because we removed the `I`-symbol from all goals in $G^{\triangleleft}$, we only need to find a bound for $G^{\triangleright}$.

Therefore, we will take a look at the five conditions (4.36 a) to (4.36 e) defining $G^{\triangleright}$:

a) The number of goals in $\{u^{\equiv} \mid u \in G\}$ is bounded by the number of equivalence classes of $\equiv$. As $\equiv$ is defined by different values of δ_{n+1} with $\delta_{n+1} \colon V \dashrightarrow 2^{Q_f}$, we can have at least $2^{|Q_f|}$ different possible values, i. e., equivalence classes. This is exponential in terms of the size of the protocol.

b) As stated above, we know by inductive hypothesis that there are at most exponentially many goals in $\text{Goals}_{\mathcal{D}_n}(>n)$. As the depth of each send clause is polynomial, there can be an exponential number of goals in the fixed part of each goal. But exponentially many goals and exponentially many nodes for each of them still gives a total number of nodes that is exponentially bounded in terms of the size of the protocol.

c) The set $\text{Static}^+_{\mathcal{D}_{n+1}}$ is defined by all nodes which are reachable from a node $v = \alpha(t)$ for a term $t \in \text{sub}(T_{\mathcal{P}})$ in less than $|\mathcal{P}|^2$ steps. This equals all nodes which are reachable from a node $v' = \alpha(t')$ for a term $t' \in T_{\mathcal{P}}$ (note the missing »sub«) in less than $|\mathcal{P}|^3$ steps. As the maximum degree of the nodes is two, this leads to exponentially many nodes in $\text{Static}^+_{\mathcal{D}_{n+1}}$.

   For the following steps we can use Lemma 4.11, which shows that there are no additional goals in $S = \text{Steps}_{\mathcal{D}_{n+1}}(\leq n)$, so all goals that we will add in (4.36 d) and (4.36 e) are in

$G \setminus S$. Also note that Lemma 4.4 states that for each node $u \in G \setminus S$, the nodes of variables of u are in S.

d) By inductive hypothesis, there is only an exponentially bounded number of m-goals u for $m > n$. We will show that on each β-path to such a goal, there can be only a polynomial number of n-goals u not in S. Therefore, we will add a polynomially bounded number of goals v for each goal u greater than n, leading to a total number of n-goals added in this step that is exponentially bounded.

So let u be an m-goal for an $m > n$. Then by Lemma 4.2 there is a unique β-path $v_1, \dots, v_l$ from $v_1 = \alpha(t_{m'})$ to $v_l = u$ for an $m' \leq m$. By the lemma, we also know $v_i \rightarrow_{\mathcal{D}_{n+1}} v_{i+1}$ for all $i \in \{1, \dots, l-2\}$.

If no n-goal exists on the path, we have nothing to show. Therefore, assume that $i \in \{1, \dots, l-1\}$ is the minimal index of an n-goal on that path, i. e., v_i is an n-goal. Lemma 4.4 shows that v_j is in S for each $j > n + |P|^2$. Hence, for u we will only add $\{v_i, \dots, v_{i+|\mathcal{P}|^2}\} \cap G$ to $G^{\triangleright}$, which is polynomially bounded.

e) We have already shown that the set $G^{\triangleright}$ is exponentially bounded if we only use conditions (4.36 a) to (4.36 d).

We will now show that for each u added so far, there is only an exponentially bounded number of goals added by condition (4.36 e). This shows that the whole set $G^{\triangleright}$ is exponentially bounded.

As we have already shown, the goals added by condition (4.36 e) cannot be in S. They have to be descendant of an n-goal $u \in G^{\triangleright}$, but the nodes of variables of u are in S (Lemma 4.4). Hence, the goals added by this condition can only be in the fixed part $\mathrm{Fix}_{\mathcal{D}_{n+1}}(u)$.

Now as the number of fixed nodes of a goal is exponentially bounded, we add only an exponentially bounded number of goals for each goal already in $G^{\triangleright}$ prior to condition (4.36 e).

Thus, $G^{\triangleright}$ is exponentially bounded, which is the desired conclusion. □

4.3.2 Lemmas concerning Goals

We will show two lemmas we referred to in the proof above. The first states that an n-goal can only be used to send goals whose step is greater than n, i. e., that a β-path through an

n-goal leads to a goal greater than n:

Lemma 4.10. *Let v be an n-goal in a simple ADAG $\mathcal{D}$ with $v \notin \text{Static}^+_{\mathcal{D}}$. Assume that $v \rightsquigarrow^* (u, \mathtt{I}^e, 0)$. Then $n \ll e$ holds.*

Proof. For the proof let v be an n_1-goal in a simple ADAG $\mathcal{D}$ with $v \notin \text{Static}^+_{\mathcal{D}}$, and assume that we have $\mathtt{I}^{e_1} \in \delta(v)$ for $e_1 = (n_1, m_1)$ and $v \rightsquigarrow^* (u, \mathtt{I}^{e_7}, 0)$. We will show that $e_1 \ll e_7$ holds.

If we have $v \rightsquigarrow^* (u, \mathtt{I}^{e_7}, 0)$, we know that $r^{e_6} \in \delta(v)$ for a stage $e_6 = (n_6, m_6)$ with $e_6 \leq e_7$. But then we also have $(\alpha(s_{n_5}), r_{n_5}^{(n_5,0)}, 0) \rightsquigarrow^* (v, r^{e_6})$ for an $n_5 \leq n_6$. As $v \notin \text{Static}^+_{\mathcal{D}}$, we know by Lemma 4.5 that v is a descendant of $w = \alpha(t_{n_4})$ for an $n_4 \leq n_5$. By condition (ADAG i a) we have $\mathtt{I}^{e_3} \in \alpha(w)$ for a stage $e_3 \ll n_4$.

Now take the first goal on the path from w to v. Either v is that goal, then by the structure of the intruder pop clauses we know $e_1 \leq e_3$, or v is a descendant of another goal v' with $\mathtt{I}^{e_2} \in \delta(v')$ for a stage $e_2 \leq e_3$. In the latter case we have $e_1 \leq e_2$ as the ADAG is simple and $v \notin \text{Static}^+_{\mathcal{D}}$.

Hence, altogether we know $e_1 \ll e_7$ by

$$\left.\begin{array}{ll} & e_1 \\ \text{or} & e_1 \leq e_2 \end{array}\right\} \leq e_3 \ll n_4 \leq n_5 \leq n_6 \leq e_7 \,. \tag{4.40}$$

□

We now show another lemma used above to prove the ADAG conditions. This lemma states that a node belonging to a step is no goal of that step or of a later step. Intuitively this should hold for a normal ADAG, as any term processed by a principal in a step has to be known to the intruder earlier than that step, so is makes no sense to send that information again.

Lemma 4.11. *Let v be a node in a simple ADAG $\mathcal{D}$ with $v \notin \text{Static}^+_{\mathcal{D}}$, but $v \in \text{Steps}_{\mathcal{D}}(\leq n)$. Then $v \notin \text{Goals}_{\mathcal{D}}(\geq n)$ holds.*

Proof. Let v be a node in a simple ADAG $\mathcal{D}$ such that $v \notin \text{Static}^+_{\mathcal{D}}$, but $v \in \text{Steps}_{\mathcal{D}}(\leq n)$ for an $n \leq N$. If v is no goal at all, we have nothing to show. Now assume that v is a goal with $\mathtt{I}^{e_v} \in \delta(v)$. Then we refer to Lemma 4.5 to show that there is an $m \leq n$ with $w \rightarrow^* v$ for $w = \alpha(t_m)$. Let $\mathtt{I}^{e_w} \in \delta(w)$, we know $e_w \ll m$. We will show that $e_v \leq e_w$ holds, which will lead to $e_v \leq e_w \ll m \leq n$, i. e., $v \notin \text{Goals}_{\mathcal{D}}(\geq n)$.

We now use the same argumentation as in the previous lemma: Take the first goal on the path from w to v. Either v itself is that goal, then by the structure of the intruder pop clauses we know $e_v \leq e_w$. Or v could be a descendant of another goal u with $\mathtt{I}^{e_u} \in \delta(v)$ for a stage $e_u \leq e_w$. In the latter case we have $e_v \leq e_u \leq e_w$ as the ADAG is simple and $v \notin \mathrm{Static}^{+}_{\mathcal{D}}$. □

4.4 Bounding the Size of an ADAG

Finally, the last theorem states that if we have such an ADAG with a bounded number of goals, we can reduce the total number of nodes in the ADAG (and the other relevant components) so that the whole ADAG is exponentially bounded.

Theorem 4.12 (Bounded ADAGs)**.** *Let $\mathcal{P}$ be a protocol. If $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ is an ADAG for $\mathcal{P}$ and (π, f) with an exponentially bounded set of goals in terms of the size of $\mathcal{P}$ and where each term occurring in Ψ has at most polynomial depth in terms of the size of the protocol. Then there is an ADAG $\mathcal{D}' = (D', \Psi, \widehat{\Gamma}', \alpha', \beta', \gamma', \delta', \lambda')$ for $\mathcal{P}$ and (π, f) which is exponentially bounded.*

As above, this is done by merging nodes with the same labeling – but now we respect the goals and the register sequences, i. e., we merge all nodes which have the same labeling, lead to the same goals and which have undistinguishable register sequences from a certain point of view. After this mergence the ADAG has only an exponential number of nodes, while the additional structures like the labeling function are also exponentially bounded in terms of the protocol size.

For the ease of understanding we will present the main proof with the help of four lemmas that will be proven afterwards Sections 4.4.1 to 4.4.4.

Proof. Let $\mathcal{P}$ be a protocol. Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be an ADAG for $\mathcal{P}$ and (π, f) with the restrictions mentioned in the theorem, and let $D = (V, F, \mu)$.

We will first define a set $\mathrm{Fixed}_{\mathcal{D}}$ of nodes that we will protect during the rest of the proof, and the complement $\mathrm{Free}_{\mathcal{D}}$:

$$\mathrm{Fixed}_{\mathcal{D}} = \mathrm{Static}^{+}_{\mathcal{D}} \cup \bigcup_{u \in \mathrm{Goals}_{\mathcal{D}}} \mathrm{Fix}_{\mathcal{D}}(u) \,, \tag{4.41}$$

$$\mathrm{Free}_{\mathcal{D}} = V \setminus \mathrm{Fixed}_{\mathcal{D}} \,. \tag{4.42}$$

For the next part of the proof, we need to know that each node in $\text{Free}_{\mathcal{D}}$ has exactly one parent. By Lemma 4.13 this assumption is possible whilst still meeting the restrictions mentioned in the theorem. Hence, we will assume that $\mathcal{D}$ is an ADAG where each $w \in \text{Free}_{\mathcal{D}}$ has exactly one parent.

To reduce the set of nodes V, we will split $\text{Free}_{\mathcal{D}}$ into two sets of paths: push paths, which are labeled with push symbols originating in $\text{Fixed}_{\mathcal{D}}$ and leading to a goal, and pop paths, which lead to a pop symbol at the top of that path. We will first define the set of push paths:

$$\text{Paths}^{\triangledown}_{\mathcal{D}} = \Big\{ [v_1, \dots, v_m] \;\Big|\; (v_1, r_n^{(n,0)}, 0) \rightsquigarrow v_2 \rightsquigarrow \cdots \rightsquigarrow v_m \rightsquigarrow (v, \mathtt{I}^e, 0) \text{ for an } n \in \{1, \dots, N\} \text{ with } v_1 = \alpha(s_n) \Big\} \quad (4.43)$$

Now the set of pop paths is defined – and according to the fifth condition of Lemma 4.2, we can assume that the nodes on these paths are not labeled with any push symbols:

$$\text{Paths}^{\triangle}_{\mathcal{D}} = \Big\{ [v_1, \dots, v_m] \;\Big|\; w \to v_1 \to v_2 \to \cdots \to v_m \text{ for a node } w \in \text{Fixed}_{\mathcal{D}} \cup \text{Paths}^{\triangledown}_{\mathcal{D}} \text{ and } v_1, \dots, v_m \notin \text{Fixed}_{\mathcal{D}} \cup \text{Paths}^{\triangledown}_{\mathcal{D}} \text{ such that for all } v_m \to w' \text{ we know } w' \in \text{Fixed}_{\mathcal{D}} \cup \text{Paths}^{\triangledown}_{\mathcal{D}} \Big\}. \quad (4.44)$$

By abuse of notation we will write $\text{Paths}^{\triangledown}_{\mathcal{D}}$ or $\text{Paths}^{\triangle}_{\mathcal{D}}$ for the set of all nodes occurring in a path in $\text{Paths}^{\triangledown}_{\mathcal{D}}$ or $\text{Paths}^{\triangle}_{\mathcal{D}}$, respectively – as we already did in (4.44).

Note that we can assume that each node in $\text{Free}_{\mathcal{D}}$ is an element of either $\text{Paths}^{\triangle}_{\mathcal{D}}$ or $\text{Paths}^{\triangledown}_{\mathcal{D}}$: Let $v \in V$ be a node. If $\text{Goals}_{\mathcal{D}}(v)$ is not empty, v is element of a β-path $\vec{v} \in \text{Paths}^{\triangledown}_{\mathcal{D}}$ by Lemma 4.2. Otherwise, we note that v leads to a leaf v' (possibly $v = v'$). By Lemma 4.3 we can assume that there is an ancestor $w \in \text{Fixed}_{\mathcal{D}}$, hence, take a path from w to v' through v, then there is a subpath $\vec{v}$ through v without nodes in $\text{Fixed}_{\mathcal{D}}$ or in $\text{Paths}^{\triangledown}_{\mathcal{D}}$, such that $\vec{v} \in \text{Paths}^{\triangle}_{\mathcal{D}}$.

Now Lemma 4.15 shows that by modifying the paths in $\text{Paths}^{\triangledown}_{\mathcal{D}}$ the number of nodes on these paths may be bounded to a double exponential number. Afterwards we can apply Lemma 4.18 which proves the same for pop paths. This yields that the total set of nodes V can be double exponentially bounded because $V = \text{Fixed}_{\mathcal{D}} \cup \text{Paths}^{\triangledown}_{\mathcal{D}} \cup \text{Paths}^{\triangle}_{\mathcal{D}}$.

Finally we can transform the ADAG such that not only the set of nodes, but the whole structure is double exponentially bounded as shown in Lemma 4.19. This concludes the proof of our theorem. □

4.4.1 Splitting Nodes with Multiple Different Parents

Lemma 4.13 (Single Parents)**.** *Let $\mathcal{D}$ be an ADAG for a protocol $\mathcal{P}$ with the same assumptions and restrictions as in Theorem 4.12. Then there exists an ADAG $\mathcal{D}'$ still fulfilling these restrictions, but where each node in* $\text{Free}_{\mathcal{D}}$ *has exactly one parent node.*

Remark 4.14. Note that there can still be multiple edges between two nodes, i. e., it is possible to have two nodes v, v' and edges $(v, v', 1)$ and $(v, v', 2)$. This is necessary to fulfill the ADAG conditions (ADAG iii b) and (ADAG iii c). But even if a node has multiple connections to its parent node, we will say that it has *only* or *exactly one parent (node)*.

Proof. Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$. We will split the nodes in $\text{Free}_{\mathcal{D}}$ which have more than one parent – this can be done without harm to the ADAG because the nodes that must not be split are in $\text{Fixed}_{\mathcal{D}}$.

Let $w \in \text{Free}_{\mathcal{D}}$ be a node with multiple different parents, i. e., the nodes $v_1, \dots, v_m$ are the parents of w for an $m > 1$ with $v_{j_1} \neq v_{j_2}$ for all $j_1, j_2 \in \{1, \dots, m\}$. Then we will define the ADAG $\mathcal{D}' = (D', \Psi, \widehat{\Gamma}, \alpha, \beta', \gamma', \delta', \lambda')$.

We begin with the DAG D': Take new nodes $w_1, \dots, w_m$. Let $V' = \{w_1, \dots, w_m\} \cup V \setminus \{w\}$. Let F' be the same as F but where each edge (v_j, w, l) is replaced by (v_j, w_j, l). And finally let μ' be the same as μ but with $\mu'(w_j) = \mu(w)$ for all $j \in \{1, \dots, m\}$. Then we set $D' = (V', F', \mu')$.

For each predicate symbol $r \in \widetilde{R}$ and each $i \in \mathbb{N}$ we again define $(w, r, i)^{\triangleright} = w_j$ if $\beta(w, r, i) = (v_j, r', i', \varphi)$. Similar to the previous constructions we define $\delta_Q = \delta(w) \cap Q$, $\delta_R = \delta(w) \cap \widetilde{R}$, as well as $\delta_{\mathtt{I}} = \delta(w) \cap \widetilde{\mathtt{I}}^{+}$ and

$$\delta_R^j = \left\{ r \in \delta_R \mid (w, r, i)^{\triangleright} = w_j \text{ for an index } i \in \mathbb{N} \right\} . \tag{4.45}$$

Note that the new nodes will not be goals as there are no goals in $\text{Free}_{\mathcal{D}}$, so we do not have to care for the witnesses of I-symbols. Now we can define δ', β', γ' and λ' for the new

nodes:

$$\delta'(w_j) = \delta_Q \cup \delta_R^j \cup \delta_I \qquad \text{for all } j \in \{1, \dots, m\}\,, \tag{4.46}$$

$$\beta'((w,p,i)^{\triangleright}_{;}\ p,i) = \beta(w,p,i) \qquad \text{for all } p \in \delta(w), i \in \gamma^\star(w,p)\,, \tag{4.47}$$

$$\gamma'((w,p,i)^{\triangleright}_{;}\ p,i) = \gamma(w,p,i) \qquad \text{for all } p \in \delta(w), i \in \gamma^\star(w,p)\,, \tag{4.48}$$

$$\lambda'(\mathtt{c}) = ((w,p,i)^{\triangleright}_{;}\ p,i,j) \qquad \text{if } \lambda(\mathtt{c}) = (w,p,i,j)\,. \tag{4.49}$$

For all other nodes in $V \cap V'$, for all related register sequences, indices and constants, let $\delta' = \delta$, $\beta' = \beta$, $\gamma' = \gamma$ and $\lambda' = \lambda$. Then $\mathcal{D}' = (D', \Psi, \widehat{\Gamma}, \alpha', \beta', \gamma', \delta', \lambda')$ is an ADAG; and $\mathcal{D}'$ has an exponential number of goals, as we did not duplicate any goals.

If there is a node $w \in \mathrm{Free}_{\mathcal{D}'}$, we will apply this transformation on $\mathcal{D}'$, until we get an ADAG $\mathcal{D}''$ that still fulfils the restrictions mentioned in Theorem 4.12, but in which each node $w \in \mathrm{Free}_{\mathcal{D}''}$ has at most one parent. Now by Lemma 4.3 we can assume that each node in $w \in \mathrm{Free}_{\mathcal{D}''}$ has *exactly* one parent. □

4.4.2 Shortening the Push Paths

Lemma 4.15 (Bounding Push Paths)**.** *Let $\mathcal{D}$ be an ADAG for a protocol $\mathcal{P}$ with the same assumptions and restrictions as in Theorem 4.12 as well as the restriction that each node in* $\mathrm{Free}_{\mathcal{D}}$ *has only one parent. Then there exists an ADAG $\mathcal{D}'$ still fulfilling these restrictions, but where the number of nodes in* $\mathrm{Paths}^{\triangledown}_{\mathcal{D}}$ *is bounded double exponentially in terms of the size of the protocol.*

Proof. Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be an ADAG as mentioned in the lemma. To prove this lemma, we define *sections* of push paths which will be special subpaths of push paths:

For each $[v_1, \dots, v_i, \dots, v_j, \dots, v_m] \in \mathrm{Paths}^{\triangledown}_{\mathcal{D}}$ the set $\mathrm{Sections}_{\mathcal{D}}$ shall contain each non-empty sequence $[v_i, \dots, v_j]$ of maximal length, such that $\mathrm{Goals}_{\mathcal{D}}(v_i) = \mathrm{Goals}_{\mathcal{D}}(v_j)$ and that $v_i, \dots, v_j \notin \mathrm{Fixed}_{\mathcal{D}}$. Maximal length means that neither $[v_{i-1}, \dots, v_j]$ nor $[v_i, \dots, v_{j+1}]$ could be in $\mathrm{Sections}_{\mathcal{D}}$.

Lemma 4.16 restricts the number of sections, i. e., the size of the set $\mathrm{Sections}_{\mathcal{D}}$ is exponentially bounded. Then Lemma 4.17 shows that $\mathcal{D}$ can be modified such that each section has a length which is double exponentially bounded, without losing the restrictions in the lemma. □

4.4.2.1 Bounding the Number of Sections

Lemma 4.16 (Bounding the Number of Sections)**.** *Let $\mathcal{D}$ be an ADAG fulfilling the restrictions mentioned in Lemma 4.15. Then the number of paths in* $\text{Sections}_{\mathcal{D}}$ *is exponentially bounded in terms of the size of the protocol.*

Proof. Let $\mathcal{D}$ be an ADAG with the necessary restrictions. Examine the size of $\text{Sections}_{\mathcal{D}}$: First, Lemma 4.2 states that the size of $\text{Paths}^{\triangledown}_{\mathcal{D}}$ equals the number of goals in $\mathcal{D}$. We look at each push path $\vec{v} = [v_1, \ldots, v_m] \in \text{Paths}^{\triangledown}_{\mathcal{D}}$, and we show that the number of sections that can be constructed from this path is bounded.

First, let $i_1, \ldots, i_l$ be the set of indices with $v_{i_j} \in \text{Fixed}_{\mathcal{D}}$ (and let $i_0 = 0$ as well as $i_{l+1} = m + 1$). This leads to the following set of subsequences of $\vec{v}$, where the size of the set is equal to or less than $l + 1 \leq |\,\text{Fixed}_{\mathcal{D}}\,| + 1$, i. e., exponential:

$$\text{Paths}^{\triangledown}_{\mathcal{D}}(\vec{v}) = \left\{ [v_{i_j+1}, \ldots, v_{i_{j+1}-1}] \;\middle|\; j \in \{0, \ldots, l\} \right\} . \tag{4.50}$$

Now let $\vec{w} = [w_1, \ldots, w_m] \in \text{Paths}^{\triangledown}_{\mathcal{D}}(\vec{v})$. Let $i_1, \ldots, i_l$ be all indices with $\text{Goals}_{\mathcal{D}}(w_{i_j}) \neq \text{Goals}_{\mathcal{D}}(w_{i_j+1})$, and so $\vec{w} \notin \text{Sections}_{\mathcal{D}}$. But we know $\text{Goals}_{\mathcal{D}}(w_{i_j+1}) \subseteq \text{Goals}_{\mathcal{D}}(w_{i_j})$ because of Lemma 4.2 and the fact that $w_{i_j+1} \in \text{Free}_{\mathcal{D}}$ has only one parent, i. e., each goal reachable from w_{i_j+1} is also reachable from w_{i_j}. Therefore, l is bounded by $|\,\text{Goals}_{\mathcal{D}}(w_1)|$, which is bounded by the total number of goals. Now we can define the following set (with $i_0 = 0$ and $i_{l+1} = m$):

$$\text{Sections}_{\mathcal{D}}(\vec{w}) = \left\{ [w_{i_j+1}, \ldots, w_{i_{j+1}}] \;\middle|\; j \in \{0, \ldots, l\} \right\} . \tag{4.51}$$

Then each sequence $\vec{u} \in \text{Sections}_{\mathcal{D}}(\vec{w})$ is a section, as $\vec{u}$ contains no nodes from $\text{Fixed}_{\mathcal{D}}$ and the set of reachable goals equals for all nodes in $\vec{u}$. On the other hand, for each section $\vec{u}$ we find a sequence $\vec{w}$ which is a subsequence of a β-path $\vec{v} \in \text{Paths}^{\triangledown}_{\mathcal{D}}$, so altogether we have:

$$\text{Sections}_{\mathcal{D}} = \left\{ \vec{u} \;\middle|\; \vec{v} \in \text{Paths}^{\triangledown}_{\mathcal{D}}, \vec{w} \in \text{Paths}^{\triangledown}_{\mathcal{D}}(\vec{v}), \vec{u} \in \text{Sections}_{\mathcal{D}}(\vec{w}) \right\} . \tag{4.52}$$

As for each $\vec{v} \in \text{Paths}^{\triangledown}_{\mathcal{D}}$ we have a set $\text{Paths}^{\triangledown}_{\mathcal{D}}(\vec{v})$ of exponential size, and because for each $\vec{w} \in \text{Paths}^{\triangledown}_{\mathcal{D}}(\vec{v})$ we have a set $\text{Sections}_{\mathcal{D}}(\vec{w})$ of exponential size, we altogether have an exponentially bounded set $\text{Sections}_{\mathcal{D}}$. □

4.4.2.2 Shortening Sections

Lemma 4.17 (Bounding the Length of Sections)**.** *Let $\mathcal{D}$ be an ADAG fulfilling the restrictions mentioned in Lemma 4.15. Then we can construct an ADAG $\mathcal{D}'$ still meeting the restrictions, but where the length of paths in* $\text{Sections}_{\mathcal{D}}$ *is exponentially bounded in terms of the size of the protocol.*

Proof. Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be an ADAG as mentioned in the lemma. We will reduce the number of nodes in a section by a pumping method.

Comparing Register Sequences and Nodes

Let $\vec{v} = [v_0, v_1, \ldots, v_m] \in \text{Sections}_{\mathcal{D}}$. We will compare each v_j with v_0 by a function Δ_{v_0} which notes the differences in the labeling δ and the register sequences γ between v_0 and v_j. We will search for nodes v_{j_1}, v_{j_2} with $\Delta_{v_0}(v_{j_1}) = \Delta_{v_0}(v_{j_2})$ and $j_2 > j_1$, and then we will replace each edge $v_{j_1-1} \to v_{j_1}$ by an edge $v_{j_1-1} \to v_{j_2}$.

As each node on the path may have a register sequence with fresh anonymous constants, we cannot simply compare the values of the register sequences – we need to abstract from the concrete constants as explained below. Thus, we first define a function $\Delta^\star$ for comparing register sequences.

We need a fixed set of Z anonymous variables $y_1, \ldots, y_Z \in Y$ and Z fresh variables $y_1^\star, \ldots, y_Z^\star \in Y^\star$. Then we define the following function on two ground register sequences:

$$\Delta^\star((\mathsf{c}_1, \ldots, \mathsf{c}_Z), (\mathsf{c}'_1, \ldots, \mathsf{c}'_Z)) = (\chi(\mathsf{c}'_1), \ldots, \chi(\mathsf{c}'_Z)) \tag{4.53}$$

$$\text{with a function } \chi(\mathsf{c}') = \begin{cases} y_j & \text{if } \mathsf{c}' = \mathsf{c}_j \text{ and } j = \min\left\{ j' \mid \mathsf{c}' = \mathsf{c}_{j'} \right\} , \\ y_j^\star & \text{if } \mathsf{c}' \notin \{\mathsf{c}_1, \ldots, \mathsf{c}_Z\} . \end{cases}$$

To clarify this definition we will give some examples. The function compares each c'_j with c_j: It uses y_j as a variable for c_j (and for all $j' > j$ with $\mathsf{c}_{j'} = \mathsf{c}_j$), and the fresh anonymous constants not occurring in $\mathsf{c}_1, \ldots, \mathsf{c}_Z$ are mapped to the fresh variables $y_i^\star$.

Note that the results may be ambiguous as we did not care for the ordering of the fresh variables $y_i^\star$; in fact, we do not need these details as we only show a rough complexity

bound. Therefore, the following results would be possible

$$\Delta^\star((a,b,c,d),\ (b,c,c,d)) = (y_2, y_3, y_3, y_4)\ , \tag{4.54}$$
$$\Delta^\star((a,b,c,c),\ (c,c,e,d)) = (y_3, y_3, y_1^\star, y_1^\star)\ , \tag{4.55}$$
$$\Delta^\star((a,b,c,d),\ (e,f,e,a)) = (y_2^\star, y_3^\star, y_2^\star, y_1)\ , \tag{4.56}$$
$$\Delta^\star((a,b,c,d),\ (e,f,e,a)) = (y_4^\star, y_1^\star, y_4^\star, y_1)\ . \tag{4.57}$$

This enables us to compare the nodes $v' \in \vec{v}$ with the node v_0 which is the first node of $\vec{v}$, but first, just for one push symbol and one of the register sequences of v_0. Therefore, let $p \in \delta(v_0)$ and $i \in \gamma^\star(v_0, p)$. Then we set:

$$\Delta_{v_0}(v', p, i) = \left\{ \Big(p',\ \Delta^\star\big(\gamma(v_0, p, i),\ \gamma(v', p', i')\big)\Big) \;\middle|\; (v_0, p, i) \rightsquigarrow^* (v', p', i') \right\}\ . \tag{4.58}$$

Now let $[(p_1, i_1), \ldots, (p_l, i_l)]$ be a sequence of all $p \in \delta(v_0)$ and $i \in \gamma^\star(v_0, p)$. Then we define:

$$\Delta_{v_0}(v') = \Big(\delta(v'),\ \Delta_{v_0}(v', p_1, i_1),\ \ldots,\ \Delta_{v_0}(v', p_l, i_l)\Big)\ . \tag{4.59}$$

We will count the possible values for Δ_{v_0}: In the first component, we have a subset of Q_f, so there are $2^{|Q_f|}$ possible values. In each of the next l components, we have a set of tuples as defined in (4.58). Each of these tuples consists of a push symbol and a register sequence containing variables from two sets of size Z, so the size of each tuple is bounded by $|Q_f| \cdot (2Z)^Z$.

Now analyze how many tuples can occur in $\Delta_{v_0}(v', p_1, i_1),\ \ldots,\ \Delta_{v_0}(v', p_l, i_l)$: Each node can contain no more register sequences than the number of goals reachable from that node, which is a consequence of Lemma 4.2. Hence, the total number of tuples in all l sets is bounded by the total number of goals. This leads to the following bound for the possible values of Δ_{v_0}:

$$2^{|Q_f|} \cdot \Big(|Q_f| \cdot (2Z)^Z\Big)^{|\,\mathrm{Goals}_\mathcal{D}\,|} \tag{4.60}$$
$$< 2^n \cdot (n \cdot (2n)^n)^{|\,\mathrm{Goals}_\mathcal{D}\,|} \qquad \text{with } n = c_1 \cdot |\mathcal{P}|^{c_0} \text{ for some } c_0, c_1 \in \mathbb{N} \tag{4.61}$$
$$\leq (2^{n \cdot c_2})^{|\,\mathrm{Goals}_\mathcal{D}\,|} \qquad \text{for some } c_2 \in \mathbb{N} \tag{4.62}$$
$$\leq 2^{n \cdot c_2 \cdot (2^{n \cdot c_3})} \qquad \text{with } |\,\mathrm{Goals}_\mathcal{D}\,| \leq 2^{n \cdot c_3} \text{ for some } c_3 \in \mathbb{N} \tag{4.63}$$
$$\leq 2^{2^{n \cdot c_4}} \qquad \text{for some } c_4 \in \mathbb{N} \tag{4.64}$$

Now we can define the transformation on the ADAG that bounds the length of a section $\vec{v} = [v_0, v_1, \ldots, v_m]$ to $2^{2^{n \cdot c_4}}$. If the length of a section is already less than $2^{2^{n \cdot c_4}}$, we have nothing to do. Else, there have to be two nodes v_{j_1}, v_{j_2} with $\Delta_{v_0}(v_{j_1}) = \Delta_{v_0}(v_{j_2})$ and $j_1 < j_2$.

We will now replace each edge $v_{j_1-1} \to v_{j_1}$ by $v_{j_1-1} \to v_{j_2}$ and further modify the ADAG. Note that by repeating this procedure we are able to bound the length of the section $\vec{v}$ to be less than $2^{2^{n \cdot c_4}}$. Then, by iterating over all sections, we reduce the number of nodes in each section.

Transforming the ADAG to Shorten a Section

We will define the term DAG D' based on D, with the modified edge as explained above and without the nodes $v_{j_1}, \ldots, v_{j_2-1}$ (and without their relevant descendants):

$$V' = V \setminus \left\{ w \in V \mid v_{j_1} \to^* w, \text{ but not } v_{j_2} \to^* w \right\} , \tag{4.65}$$

$$F' = (F \cap (V' \times V' \times \mathbb{N})) \cup \left\{ (v_{j_1-1}, v_{j_2}, l) \mid (v_{j_1-1}, v_{j_1}, l) \in F \right\} . \tag{4.66}$$

We now have to change the witness function for v_{j_2} to point to v_{j_1-1}, and we have to modify some register sequences as the constants in the register sequences of v_{j_1} and v_{j_2} may differ: Each fresh anonymous constant c that is generated between v_0 and v_{j_1-1} will be replaced by $\hat{\chi}(\mathsf{c})$. This may mean that it is not changed at all if $\hat{\chi}(\mathsf{c}) = \mathsf{c}$, but in this way we will construct β-paths which are valid for the register sequences of v_{j_2}.

Note that in this step we may have removed an unbounded number of anonymous constants: All the constants that had been generated along the path from v_{j_1} to v_{j_2} are deleted if they do not occur in any register sequence of v_{j_2}.

From $\Delta_{v_0}(v_{j_1}) = \Delta_{v_0}(v_{j_2})$ we know $\delta(v_{j_1}) = \delta(v_{j_2})$. We also know that for each $p \in \delta(v_{j_1})$ and each $i_1 \in \gamma^\star(v_{j_1}, p)$ we have a matching index $i_2 \in \gamma^\star(v_{j_2}, p)$. For simplification we just assume that $i_1 = i_2$; this is possible because the actual values of the indices i_1, i_2 do not matter, so we could modify the values of $\gamma^\star(v_{j_2}, p)$ to match $\gamma^\star(v_{j_1}, p)$ without damaging the ADAG.

Now we know that for each register sequence κ_{v_0} of v_0, for each $p \in \delta(v_{j_1})$ and each $i \in \gamma^\star(v_{j_1}, p)$ we have a register sequence κ_Δ with

$$\kappa_\Delta = \Delta^\star(\kappa_{v_0}, \gamma(v_{j_1}, p, i)) = \Delta^\star(\kappa_{v_0}, \gamma(v_{j_2}, p, i)) . \tag{4.67}$$

Let χ_1^{-1} be a mapping from constants to variables which replaces each anonymous constant with the corresponding variable in the register sequence of variables κ_Δ, i. e., this mapping – extended on register sequences – would give $\chi_1^{-1}(\gamma(v_{j_1}, p, i)) = \kappa_\Delta$. Let χ_2 be the mapping from variables to anonymous constants such that – extended on register sequences – the mapping χ_2 would give $\chi_2(\kappa_\Delta) = \gamma(v_{j_2}, p, i)$. By (4.53) we know both χ_1^{-1} and χ_2 exist and that when combining them to $\chi_{(p,i)} = \chi_1^{-1} \circ \chi_2$ we have a function with $\chi_{(p,i)}(\gamma(v_{j_1}, p, i)) = \gamma(v_{j_2}, p, i)$.

Let $\hat{\chi}$ be the unification of all functions $\chi_{(p,i)}$ for each $p \in \delta(v_{j_1})$ and each $i \in \gamma^\star(v_{j_1}, p)$. This unification is a well-defined function as each constant in its domain is either fresh or in a register sequence of v_0, and $\hat{\chi}$ is the identity on the latter ones. We will define the functions of the ADAG $\mathcal{D}' = (D', \Psi, \widehat{\Gamma}, \alpha, \beta', \gamma', \delta', \lambda')$, beginning with δ', which is just the restriction of δ to V'. Now define the register sequence and the witness function for all nodes $v \in V'$, symbols $p \in \delta'(v)$, and indices $i \in \gamma^\star(v, p)$ by

$$\beta'(v, p, i) = \begin{cases} \beta(v, p, i), & \text{if } v \neq v_{j_2}\ , \\ \beta(v_{j_1}, p, i), & \text{if } v = v_{j_2}\ , \end{cases} \tag{4.68}$$

$$\gamma'(v, p, i) = \begin{cases} \hat{\chi}(\gamma(v, p, i)), & \text{if } v \in \{v_0, \ldots, v_{j_1 - 1}\}\ , \\ \gamma(v, p, i), & \text{otherwise}\ . \end{cases} \tag{4.69}$$

We can now define the freshness function – this looks a bit technical because we may have removed nodes and their register sequences, and because other register sequences may have been affected by $\hat{\chi}$. Note that λ is not defined for constants $\mathsf{c} \in \mathrm{dom}(\hat{\chi}) \setminus \mathrm{range}(\hat{\chi})$, as they are replaced and removed from the ADAG.

$$\lambda'(\mathsf{c}) = \begin{cases} \lambda(\mathsf{c}), & \text{if } \mathsf{c} \notin \mathrm{dom}(\hat{\chi}) \text{ and } \lambda(\mathsf{c}) = (v, p, i, z) \text{ for a node } v \in V'\ , \\ \lambda(\hat{\chi}^{-1}(\mathsf{c})), & \text{if } \mathsf{c} \in \mathrm{range}\, \hat{\chi}\ . \end{cases} \tag{4.70}$$

Now $\mathcal{D}'$ is an ADAG still fulfilling all restrictions mentioned in Theorem 4.12, but with a shortened section. As explained above, we can repeat this procedure until each section has a bounded length.

Proving the ADAG Conditions and Restrictions

We will now prove that $\mathcal{D}'$ is still an ADAG. First, me modified the DAG, but we only connected two nodes with an edge that were connected before by a sequence of edges, so

D' is still acyclic. We will now check conditions (ADAG i a) to (ADAG iii d). As we did not manipulate `I`-symbols, there is nothing to show for (ADAG i a) to (ADAG ii). Now let $v \in V'$ and $p \in \delta'(v, p)$. Then we have $v \in V$ and $p \in \delta(v, p)$ and one of the conditions (ADAG iii a) to (ADAG iii d) holds for v and p in $\mathcal{D}$. Again, distinguish between these four cases:

(iii a) In this case we have nothing to show because $v \in \text{Fixed}_{\mathcal{D}}$.

(iii b) For the application of pop clauses note that when manipulation edges, we did replace each edge $v_{j_1-1} \to v_{j_1}$ by an edge $v_{j_1-1} \to v_{j_2}$, but we know $\delta(v_{j_1}) = \delta(v_{j_2})$, so (ADAG iii b) still holds.

(iii c) This case is the interesting one. Distinguish three cases:

- First, let $v \notin \{v_0, v_1, \ldots, v_{j_2}\}$. Then nothing was changed for v, so condition (ADAG iii c) still holds.
- Now let $v = v_{j_2}$, so we connected v to a new parent node during the modification. First, we modified all edges between v and his parent consistently. We did not change the register sequences of v, so the freshness function is still correct, but we have to show that the modified witness function is correct, i. e., for each register sequence of v there is a compatible register sequence of the new parent.

  Let $i \in \gamma^\star(v, p)$. Then we know that not only $p \in \delta(v)$, but also $p \in \delta(v_{j_1})$ by equation (4.59). Hence, there was a witness $\beta(v_{j_1}, p, i)$ in $\mathcal{D}$, and we can use it as a witness $\beta'(v, p, i)$.

  Now for the embedding of the register sequences let $\beta(v_{j_1}, p, i) = (v', p', i', \varphi)$ (with $v' = v_{j_1-1}$), and let κ, κ' be the register sequences in φ. Then we can start with the compatibility of the embeddings in $\mathcal{D}$ and show:

$$(\kappa, \kappa') \mapsto (\gamma(v', p', i'),\ \ \gamma(v_{j_1}, p, i))\ , \tag{4.71}$$

$$\text{it follows } (\kappa, \kappa') \mapsto (\hat{\chi}(\gamma(v', p', i')),\ \ \hat{\chi}(\gamma(v_{j_1}, p, i))) \tag{4.72}$$

$$= (\hat{\chi}(\gamma(v', p', i')),\ \ \gamma(v_{j_2}, p, i)) \tag{4.73}$$

$$= (\gamma'(v', p', i'),\ \ \gamma'(v_{j_2}, p, i))\ . \tag{4.74}$$

  The first transformation is valid because $\hat{\chi}$ is a function that changes both single embeddings in the same way. The third step is valid by definition of $\hat{\chi}$ and the functions $\chi_{(p,i)}$, whereas the final step is valid by definition of γ' (4.69).

  Now we know that condition (ADAG iii c) still holds for this case.

- Finally, we assume that $v \in \{v_0, v_1, \ldots, v_{j_1-1}\}$, so v's register sequences may have been changed by χ, and so may the freshness function.

  As in case (ADAG iii d) we know the embeddings may have changed, but are still compatible, as we did change the register sequence of the parent node of v in the same way. This does even hold for $v = v_0$ as we do not change its register sequences (or that of its parent node) because by definition, $\mathrm{dom}(\hat{\chi})$ only contains constants that are in none of the register sequences of v_0, i. e., fresh.

  The freshness function was modified accordingly, i. e., when $\mathtt{c}$ was replaced by $\mathtt{c}'$, we have $\lambda'(\mathtt{c}') = \lambda(\mathtt{c})$. Thus, condition (ADAG iii c) also holds for this case.

(iii d) In this case we know that $v \in \mathrm{Fixed}_{\mathcal{D}}$, and also $v' \in \mathrm{Fixed}_{\mathcal{D}}$ for $\beta(v, \mathtt{I}) = (v', p', i', \varphi)$, so we have nothing to show.

This proves that our construction yields an ADAG. But this ADAG also fulfills all restrictions we mentioned before: It is still simple; we only shortened a path, so each set of nodes contradicting the simplicity would also show that the ADAG we started with would not have been simple. It still has only one parent for each node in $\mathrm{Free}_{\mathcal{D}}$; we only changed each edge $v_{j_1-1} \to v_{j_1}$ to $v_{j_1-1} \to v_{j_2}$ and removed the node v_{j_2-1}. The set of goals was not manipulated, this also holds for the stage theory, so both are still bounded. □

4.4.3 Shortening the Pop Paths

Lemma 4.18 (Bounding Pop Paths)**.** *Let $\mathcal{D}$ is an ADAG for a protocol $\mathcal{P}$, which fulfils the restrictions of Theorem 4.12 as well as the restrictions that each node in* $\mathrm{Free}_{\mathcal{D}}$ *has only one parent and that the number of nodes in the push paths is double exponentially bounded in terms of the size of the protocol. Then there exists an ADAG $\mathcal{D}'$ fulfilling these restrictions, while the number of nodes in* $\mathrm{Paths}^{\triangle}_{\mathcal{D}}$ *is double exponentially bounded as well.*

Proof. We will prove this lemma by first shortening each single path and then showing that the sum over all nodes in all paths is double exponentially bounded.

Shortening a Pop Path

Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be such an ADAG with the DAG $D = (V, F, \mu)$, and let $\vec{v} = [v_1, \ldots, v_m] \in \mathrm{Paths}^{\triangle}_{\mathcal{D}}$. By Lemma 4.2 we can assume that there are no push symbols in the

labeling of the nodes of $\vec{v}$ and that for all nodes $v_j \in \vec{v}$ we have $\text{Goals}_{\mathcal{D}}(v_j) = \emptyset$ (otherwise, v_j would be in $\text{Paths}^{\triangledown}_{\mathcal{D}}$).

Let $M = 2^{|Q_f|}$. Then we know that if $m > M$, there are two nodes v_{j_1}, v_{j_2} with $\beta(v_{j_1}) = \beta(v_{j_2})$ and $j_1 < j_2$. Like in Section 4.4.2.2, we will replace the edge $v_{j_1-1} \to v_{j_1}$ by $v_{j_1-1} \to v_{j_2}$. However, we do not have to remove $v_{j_1}, \ldots, v_{j_2-1}$ (and the relevant descendants) explicitly, as shown below.

Formally we define $\mathcal{D}'$ based on $\mathcal{D}$ just by replacing the set of edges F with the following set F':

$$F' = (F \setminus \{ (v_{j_1-1}, v_{j_1}, l) \mid l \in \mathbb{N} \}) \cup \{ (v_{j_1-1}, v_{j_2}, l) \mid (v_{j_1-1}, v_{j_1}, l) \in F \} \ . \tag{4.75}$$

This clearly leads to an ADAG: The only condition affected may be (ADAG iii b), but it still holds because of $\beta(v_{j_1}) = \beta(v_{j_2})$.

Now we can use Lemma 4.3 which shows that each node w which was on a pop path $\vec{w} = [v_1, \ldots, v_{j_1}, \ldots, w, \ldots] \in \text{Paths}^{\triangle}_{\mathcal{D}}$ and that is no descendant of v_{j_2} can be removed, as it is no longer a descendant of any $\alpha(t)$ for any $t \in T_{\mathcal{P}}$.

Then we get an ADAG with the necessary restrictions: The only thing we have to check is that each node in $\text{Free}_{\mathcal{D}}$ has only one parent. In $\text{Free}_{\mathcal{D}'}$, there was exactly one node with more than one parent: The node v_{j_2} was a descendant of v_{j_2-1} and v_{j_1-1}. But by applying Lemma 4.3 we removed v_{j_2-1}, so this restriction holds again. We can repeat this procedure until the length of each pop paths is bounded by M.

Bounding the Number of Pop Paths

We will show that in an ADAG $\mathcal{D}$ with a bounded length of the pop paths as described above, there is at most a double exponential number of pop paths, which proves our statement that there are at most double exponentially many nodes in the set $\text{Paths}^{\triangle}_{\mathcal{D}}$.

Let $w \in \text{Fixed}_{\mathcal{D}} \cup \text{Paths}^{\triangledown}_{\mathcal{D}}$ be a node. Let t_w be the tree rooted in w consisting of all paths $[v, \ldots] \in \text{Paths}^{\triangle}_{\mathcal{D}}$ with $w \to v$; we know this is a tree because the nodes in $\text{Paths}^{\triangle}_{\mathcal{D}}$ have exactly one parent. The grade of each node in this tree is bounded by the maximal degree of the signature, and by assumption about the length of the pop paths the depth of the tree is at most exponential. Hence, we know that there are at most double exponentially many nodes in t_w.

As each path in $\text{Paths}^{\triangle}_{\mathcal{D}}$ belongs to a tree t_w, we can bound the number of nodes in $\text{Paths}^{\triangle}_{\mathcal{D}}$ by the following expression which is double exponential in terms of the size of the protocol:

$$\sum_{w \in \text{Fixed}_{\mathcal{D}}} |\{v \in V \mid v \in t_w\}| \,. \tag{4.76}$$

This concludes the proof of this lemma and shows that the total number of nodes in an ADAG can be bounded to be double exponential. □

4.4.4 Reducing the ADAG Components

Lemma 4.19 (Bounding the ADAG)**.** *Let $\mathcal{D}$ is an ADAG for a protocol $\mathcal{P}$ with a double exponentially bounded number of nodes in terms of the size of $\mathcal{P}$, and with the bounds assumed in Theorem 4.12. Then we can assume that the whole ADAG with all of its components has a double exponentially bounded size.*

Proof. Let $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ be an ADAG with a double exponentially bounded number of nodes. We will now show bounds for all components or reduce them, leading to an ADAG that can be described in a double exponentially bounded amount of space.

We will use the assumption that the indices of register sequences $i \in \gamma^\star(v, p)$ for each node v and each symbol $p \in \delta(v)$ do not exceed an exponential size: By Lemma 4.2 the number of register sequences in a node v, i. e., $\sum_{p \in \delta(v)} |\gamma^\star(v, p)|$, is bounded by the number of goals that v leads to. As an upper bound, each node could lead to exponentially many goals and thus have exponentially many register sequences. As the values of the indices do not matter, we can rename each register sequence such that these indices are smaller than a certain bound $M \in O(2^{|\mathcal{P}|})$ where O is the Landau notation.

Now we will turn to the components of the ADAG. As the set of nodes is bounded, D is bounded as well. By the assumption transferred from Theorem 4.12, the stage theory Ψ is exponentially bounded.

Now look at the set of anonymous constants $\widehat{\Gamma}$: We know we have at most M register sequences at one node. Each of the register sequences could contain Z fresh anonymous constants. Thus, the total number of constants occurring in the ADAG is also double exponentially bounded, and we can set $\widehat{\Gamma}' \subseteq \widehat{\Gamma}$ to contain all anonymous constants in all register sequences as well as all constants used in the labeling of the DAG. Then, $\widehat{\Gamma}'$ is double exponentially bounded.

Finally, look the functions of $\mathcal{D}$:

$$\alpha\colon \operatorname{sub}(T_{\mathcal{P}}) \to V\,, \tag{4.77}$$
$$\beta\colon V \times Q_f \times \mathbb{N} \dashrightarrow V \times \mathbb{N} \times \Psi\,, \tag{4.78}$$
$$\gamma\colon V \times Q_f \times \mathbb{N} \dashrightarrow \widehat{\Gamma}'^{Z}\,, \tag{4.79}$$
$$\delta\colon V \dashrightarrow 2^{Q_f}\,, \tag{4.80}$$
$$\lambda\colon \widehat{\Gamma}' \dashrightarrow V \times Q_f \times \mathbb{N} \times \{1,\dots,Z\}\,. \tag{4.81}$$

We will analyze the size of each set occurring in the list of functions above:

- The sets $T_{\mathcal{P}}$, Q_f, and Z have polynomial size in terms of the size of the protocol.
- The sets $\operatorname{sub}(T_{\mathcal{P}})$, 2^{Q_f}, and Ψ are of exponential size.
- The cardinality of V, $\widehat{\Gamma}'$, and $\widehat{\Gamma}'^{Z}$ is double exponential.
- Finally, each occurrence of $\mathbb{N}$ can be substituted by M, which is double exponential.

Now we will use the fact that for finite sets A and B, the description of a function $f\colon A \to B$ can be done in space $O(|A| \cdot \log|B|)$ (with Landau notation O) by taking functions $\mu_A\colon \{1,\dots,|A|\} \to A$ and $\mu_B\colon B \to \{1,\dots,|B|\}$ and storing the $|A|$-tuple

$$\Big(\mu_B(f(\mu_A(1))),\dots,\mu_B(f(\mu_A(|A|)))\Big)\,. \tag{4.82}$$

Thus, all functions and hence the whole ADAG can be described in space that is at most double exponential. □

4.5 Proof of the Main Result

Proof of the Main Theorem 2.17. Let $\mathcal{P} = (P, \Phi)$ be a protocol. Theorems 4.12 allow us to develop the following algorithm:

Algorithm 4.20. *Let $\mathcal{P} = (P, \Phi)$ be a protocol. Guess an execution scheme π and a stage mapping f for $\mathcal{P}$. Construct the instance $\widehat{\Phi}_f$ of Φ_f of exponential size as shown in Definition 4.7. Take a double exponential set $\widehat{\Gamma}$ of anonymous constants. Then guess a DAG $D = (V, F, \mu)$ of double exponential size which uses only constants from $\widehat{\Gamma}$. Guess functions $\alpha, \beta, \gamma, \delta, \lambda$ of appropriate type. Then verify if the ADAG conditions (ADAG i a) to (ADAG iii d) hold for $\mathcal{D} = (D, \Psi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$, the protocol $\mathcal{P}$ and (π, f). If that is the case, return »insecure«, otherwise return »secure«.*

Our algorithm decides in nondeterministic double exponential time if there is an attack by constructing and guessing all necessary components in nondeterministic double exponential time and then checking the properties of an ADAG. The latter can be done by checking (ADAG i) – this is possible in less than $\mathcal{O}(|\beta|)$ steps – and then iterating over each node, checking conditions (ADAG ii) and (ADAG iii) for each node – this can mostly be done locally in that node and its direct ancestors and descendants, the most complex thing to check is the embedding in the case (ADAG iii d), but this only affects an exponential number of nodes, i. e., the goals.

This concludes the proof of our main theorem: We can check nondeterministically in double exponential time if there is an ADAG for a given protocol, i. e., if there is an attack on the protocol. □

5 Conclusion

The goal of this report was to extend the model of selecting theories used to analyze recursive cryptographic protocols; the extension now allows us to generate fresh tokens like nonces or keys. Our main theorem shows that security of protocols in this model is decidable with respect to a bounded number of sessions.

To obtain this result we extended the formalism and protocol model introduced in [Tru05a, Tru05b]: First, the signature of the protocol model was joined with an infinite set of constants like in [KW04], then we modified the notion of ADAGs to cope with the generation of fresh tokens and unbounded finite subsets of infinite signatures.

We have shown that in this model, security is equivalent to the absence of an ADAG, and so we were able to prove decidability by bounding the size of ADAGs. To do so, we described a method to scaled down any ADAG until its size is at most double exponential in terms of the size of the protocol.

Extensions and Open Problems

A major disadvantage of this model is the necessity for simple or flat push clauses and simple or flat left-hand-sides in send clauses. Another problem is to find a tighter complexity bound for the decision problem.

In addition, it would be interesting if anonymous constants could be used as keys in this model. This is not trivial as the definition of an ADAG relies on the bounded number of keys occurring in a protocol. Furthermore, allowing complex keys or algebraic properties of XOR leads to undecidability in the general case [KT07]; but for restricted classes of protocols it is possible to model properties of XOR while still maintaining decidability [KT07]. A future direction could be to combine the approach of [KT07] with anonymous constants.

List of Figures

Index

Bibliography

[AC02] Roberto M. Amadio and Witold Charatonik. On Name Generation and Set-Based Analysis in the Dolev-Yao Model. In *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2002.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In *Proceedings of the International Conference IFIP on Theoretical Computer Science (TCS 2000), Exploring New Frontiers of Theoretical Informatics*, pages 3–22, London, UK, 2000. Springer-Verlag.

[Bla01] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, 2001. IEEE Computer Society.

[BO97] John A. Bull and David J. Otway. The Authentication Protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, 1997.

[Boy90] Colin Boyd. Hidden Assumptions in Cryptographic Protocols. *Proceedings of the IEEE*, 137, Part E(6):433–436, 1990.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A Universally Composable Cryptographic Library. Cryptology ePrint Archive, Report 2003/015, 2003. `http://eprint.iacr.org/`.

[BR93] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[BS97] Jeremy Bryans and Steve Schneider. CSP, PVS and a Recursive Authentication Protocol. In *DIMACS Workshop on Design and Formal Verification of Crypto Protocols*, 1997.

[CBH05] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in computational complexity proofs for protocols. Cryptology ePrint Archive, Report 2005/351, 2005. http://eprint.iacr.org/.

[CDL+99] Iliano Cervesato, Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A Meta-Notation for Protocol Analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW 1999)*, pages 55–69, 1999.

[CJ97] John A. Clark and Jeremy L. Jacob. A Survey of Authentication Protocol Literature. Version 1.0. http://www-users.cs.york.ac.uk/~jac/papers/drareview.ps.gz, 1997.

[CJM98] Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. Using State Space Exploration and a Natural Deduction Style Message Derivation Engine to Verify Security Protocols. In *PROCOMET*, volume 125 of *IFIP Conference Proceedings*, pages 87–106. Chapman & Hall, 1998.

[CKRT03a] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

[CKRT03b] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science — 23rd Conference*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive Or. In *LICS*, page 271. IEEE Computer Society, 2003.

[CS02] Hubert Common and Vitaly Shmatikov. Is it Possible to Decide Whether a Cryptographic Protocol is Secure or not? *Journal of Telecommunications and Information Technology*, 4:5–14, 2002.

[CW05] Véronique Cortier and Bogdan Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[DK02] Hans Delfs and Helmut Knebl. *Introduction to Cryptography: Principles and Applications*. Springer-Verlag, Berlin, Germany, 2002.

[DLMS99] Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Undecidability of Bounded Security Protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols – FMSP, Trento, Italy*, 1999.

[DLMS04] Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[EG83] Shimon Even and Oded Goldreich. On the Security of Multi-Party Ping-Pong Protocols. In *FOCS*, pages 34–39. IEEE, 1983.

[FHG98] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand Spaces: Why is a Security Protocol Correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, 1998.

[FS00] Niels Ferguson and Bruce Schneier. A Cryptographic Evaluation of IPsec. `http://www.schneier.com/paper-ipsec.html`, 2000.

[GB01] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. Summer Course »Cryptography and Computer Security« at MIT, 1996–2001, 2001.

[HE95] Kipp E. B. Hickman and Taher Elgamal. The SSL Protocol, 1995. Internet-Draft draft-hickman-netscape-ssl-01.txt.

[Hor51] Alfred Horn. On Sentences Which are True of Direct Unions of Algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951.

[Kau05] Charlie Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), 2005.

[KKW05] Detlef Kähler, Ralf Küsters, and Thomas Wilke. Deciding Properties of Contract-Signing Protocols. In *STACS*, volume 3404 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2005.

[Kow79] Robert A. Kowalski. *Logic for Problem Solving*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1979.

[KT07] Ralf Küsters and Tomasz Truderung. On the Automatic Analysis of Recursive Security Protocols with XOR. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, volume 4393 of *Lecture Notes in Computer Science*. Springer, 2007.

[Küs03] Ralf Küsters. Tree Transducer-based Analysis of Cryptographic Protocols. Technical Report 0301, Institut für Informatik und Praktische Mathematik, CAU Kiel, Germany, 2003. Available from http://www.informatik.uni-kiel.de/reports/2003/0301.html.

[KW04] Ralf Küsters and Thomas Wilke. Automata-Based Analysis of Recursive Cryptographic Protocols. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, volume 2996 of *Lecture Notes in Computer Science*, pages 382–393. Springer-Verlag, 2004.

[Llo84] John W. Lloyd. *Foundations of logic programming*. Springer-Verlag, New York, NY, USA, 1984.

[Low95] Gavin Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[Low96] Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop (TACAS 1996)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[Mea95] Catherine Meadows. Formal Verification of Cryptographic Protocols: A Survey. In *Proceedings of the 4th International Conference on the Theory and Applications of Cryptology (ASIACRYPT 1994)*, pages 135–150, London, UK, 1995. Springer-Verlag.

[Mea00] Catherine Meadows. Extending Formal Cryptographic Protocol Analysis Techniques for Group Protocols and Low-Level Cryptographic Primitives. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS 2000)*, pages 1–4, 2000.

[Mea01] Catherine Meadows. Open Issues in Formal Methods for Cryptographic Protocol Analysis. In *Proceedings of the International Workshop on Information Assurance in Computer Networks (MMM-ACNS 2001)*, page 21, London, UK, 2001. Springer-Verlag.

[Mea03] Catherine Meadows. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *IEEE Journal on Selected Areas in Communications*, 21(1):44–54, January 2003.

[MMS97] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated Analysis of Cryptographic Protocols using Murφ. In *IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society, 1997.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, 1996.

[Nie05] Robert Nieuwenhuis, editor. *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*, volume 3632 of *Lecture Notes in Computer Science*. Springer, 2005.

[NS78] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, 21(12):993–999, 1978.

[OR87] David J. Otway and Owen Rees. Efficient and Timely Mutual Authentication. *Operating Systems Review*, 21(1):8–10, 1987.

[Pau97] Lawrence C. Paulson. Mechanized Proofs for a Recursive Authentication Protocol. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW 1997)*, pages 84–95, Washington, DC, USA, 1997. IEEE Computer Society Press.

[PQ01] Olivier Pereira and Jean-Jacques Quisquater. A Security Analysis of the Cliques Protocols Suites. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 73–81. IEEE Computer Society Press, 2001.

[Rob65] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[RS98] Peter Y. A. Ryan and Steve A. Schneider. An Attack on a Recursive Authentication Protocol. A Cautionary Tale. *Information Processing Letters*, 65(1):7–10, 1998.

[RT01] Michaël Rusinowitch and Mathieu Turuani. Protocol Insecurity with Finite Number of Sessions is NP-Complete. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations (CSFW 2001)*, page 174, Washington, DC, USA, 2001. IEEE Computer Society.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol Insecurity with a Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science*, 299(1-3):451–475, 2003.

[SB05] Graham Steel and Alan Bundy. Attacking group protocols by refuting incorrect inductive conjectures. *Journal of Automated Reasoning*, Special Issue on Automated Reasoning for Security Protocol Analysis:1–28, December 2005.

[SNS88] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the USENIX Winter 1988 Technical Conference*, pages 191–202. USENIX Association, 1988.

[Sti95] Douglas R. Stinson. *Cryptography: Theory and Practice (Discrete Mathematics and its Applications)*. CRC Press, 1995.

[Tru05a] Tomasz Truderung. Regular Protocols and Attacks with Regular Knowledge. In Nieuwenhuis [Nie05], pages 377–391.

[Tru05b] Tomasz Truderung. Selecting Theories and Recursive Protocols. In *Concurrency Theory, 16th International Conference (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2005.

[VSS05] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the Complexity of Equational Horn Clauses. In Nieuwenhuis [Nie05], pages 337–352.

[Ylö96] Tatu Ylönen. SSH – Secure Login Connections over the Internet. In *Proceedings of the 6th USENIX Security Symposium*, pages 37–42, San Jose, CA, USA, 1996. USENIX Association.

[Zho99] Jianying Zhou. Fixing a Security Flaw in IKE Protocols. *Electronic Letter*, 35(13):1072–1073, 1999.