HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **34**; March 1995
ISSN 0783–5396
ISBN 951–22–2538–7

PrT-NET BASED ANALYSIS OF INFORMATION FLOW SECURITY NETS

Jari Juopperi

VTT Information Technology
P.O.Box 1203
02044 VTT
FINLAND

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports — ISSN 0783–5396
No. **34**; March 1995 — ISBN 951–22–2538–7

PrT-net Based Analysis of Information Flow Security Nets

Jari Juopperi

Abstract: Information Flow Security Nets (IFS-nets) provide a framework for the definition of security models. IFS-nets employ a class of low-level Petri nets as their model of computation while the notion of security is based on the extension of Denning's information flow model. An inconvenient property of IFS-nets is that the direct analysis of the nets without specialized tools is difficult. The aim of this work is to show how the security properties of an IFS-net can be inspected indirectly by constructing a PrT-net and by examining the reachability graph of the constructed PrT-net. The crux of this approach is the mapping which defines how the PrT-net corresponding to an IFS-net can be created. It is shown that there exists an one-to-one correspondence between the reachable markings of the IFS-net and the reachable markings of the corresponding PrT-net. It is also shown that there exists an one-to-one correspondence between the firing sequences of an IFS-net and the firing sequences of the corresponding PrT-net. Several results linking the security properties of IFS-nets to the properties of the reachability graphs of the corresponding PrT-nets are provided. Finally, the use of the proposed approach is illustrated in an example where the upgrading and downgrading of data must be properly done. The IFS-net and the corresponding PrT-net of the example system are created. After this the security properties of the system are analysed with the reachability analysis tool `PROD`.

Keywords: security model, information flow model, security analysis, Information Flow Security Nets, Predicate/Transition nets, Petri nets, reachability analysis

Printing: TKK Monistamo; Otaniemi 1995

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

1 Introduction

Information security deals with the use and abuse of computer and communication systems. It has three separate but interrelated goals [35]

- secrecy,
- integrity,
- availability.

Secrecy (or confidentiality) is related to the disclosure of information. For example, in a payroll system secrecy is concerned with preventing an employee from finding out his superior's salary. Usually it is also forbidden for an employee to change his salary. Such a modification of information is an issue of integrity. Finally, availability is related to the denial of access to information. In the context of a payroll system availability is concerned with ensuring that paychecks are printed on time.

The needs of military applications have traditionally been the driving force for the development in information security. Currently the research on information security is also motivated by the interests of non-military organizations. This is due to the fact that these organizations are more and more dependent on computer and communication facilities in their daily work and at the same time the data processed by these systems is considered to be very valuable in terms of money.

A beautiful example of theoretical research emerging from non-military needs is described in [1]. Brewer and Nash discuss confidentiality in financial sector of commercial segment where consultants provide services for competing companies. The objective of the work is to formally describe the policies for preventing information flows which could cause conflict of interests for individual consultants.

Another non-military source of motivation for the research on information security are the recent applications of distributed computing and communication technology. For example, the advent of computing systems dedicated to assist the group oriented solving of problems, i.e., systems for *computer supported cooperative work (CSCW)* has risen a number of new questions related to information security [21, 36].

Although the analysis of security properties is the main theme of this work, the importance of integrity and availability should not be underestimated. Good starting points for the study of integrity and availability are, for example, [20, 23] and [43, 26].

1.1 Formal Models of Security

In order to conduct a formal study of any interesting property of a concrete computing system, a well defined abstraction of the system is needed. The abstrcation describes all the relevant features of the actual system and lays the basis for the machinery which will be used to draw further conclusions. *Security model* is an abstraction which is created for the research of security properties.

Although the detailed structure of security models vary considerably, two basic elements, *the model of computation* and *the notion of security*, can always be found from the models. The model of computation summarizes, as usual, the computational characteristics of the real-world system under inspection. The notion of security, on the other hand, consists of two conceptually different parts. *Security primitives* are the grips for the reasoning about security attached to the model of computation. *The definition of security* tells which individual computations are considered to be secure.

From the engineering point of view a security model should support the construction and verification of secure systems. In practical terms this means, for example, that the models can be easily interfaced with the standard formalisms used in software engineering. Furthermore, there should exist tools to aid the effective analysis of the instances of security models. The problems related to this aspect of information security has been studied in [14].

In the following subsections, the basic ideas of a few classical security models are reviewed. The models described in the sequel have been selected to act as the standard points of comparison and therefore the list of reviewed models is by no means comprehensive. A more complete discussion of classical security models can be found from [22].

1.2 Access Matrix Model

Harrison, Ruzzo and Ullman formalized ideas about the description of security properties in their work [15]. This *access matrix model* was inspired by the views from the database and operating systems research.

There are three principal components in the access matrix model:

- a set S of *subjects* which are the active entities of the system,
- a set O of *objects* which are the entities to be controlled,
- an *access matrix* A with rows corresponding to subjects and rows corresponding to objects.

An entry $A[s, o]$ of the access matrix contains all *access rights* of subject s for object o. Furthermore, the subjects are also considered to be objects, i.e., $S \subseteq O$. In operating systems the objects are typically files, segments of memory and activations of programs while users and processes are representatives of objects.

The access rights specify the ways subjects may access objects. The rights of files, for example, normally include read, write and execute permission while another set of rights may be applied to other types of objects. Some access rights are *generic* in a sense that they can be associated with several types of objects. The ownership of an object is a generic right frequently found in operating systems.

The state of the modelled system is described with *a configuration.* A configuration is a triple (S, O, A) where S, O and A are defined as above. Changes of the state of the system are modelled by a set of *commands.* Every command obey the following form

$$
\begin{array}{l}
\textbf{command } c(x_1, \ldots, x_k) \\
\quad \textbf{if } \; r_1 \text{ in } A[x_{s_1}, x_{o_1}] \text{ and} \\
\quad \phantom{\textbf{if }} \; r_2 \text{ in } A[x_{s_2}, x_{o_2}] \text{ and} \\
\quad \phantom{\textbf{if }} \; \vdots \\
\quad \phantom{\textbf{if }} \; r_m \text{ in } A[x_{s_m}, x_{o_m}] \\
\quad \textbf{then } \; op_1; \\
\quad \phantom{\textbf{then }} \; op_2; \\
\quad \phantom{\textbf{then }} \; \vdots \\
\quad \phantom{\textbf{then }} \; op_n; \\
\quad \textbf{end}
\end{array}
$$

where $x_1, \ldots, x_k$ are formal parameters of the command, $r_1, \ldots, r_m$ are access rights, $op_1, \ldots, op_n$ are primitive operations, and $s_1, \ldots, s_m$ and $o_1, \ldots, o_m$ are integers from the set $\{1, \ldots, k\}$. Each *primitive operation* is one of the following:

enter r **into** $A[s, o]$,
delete r **from** $A[s, o]$,
create subject s,
create object o,
delete subject s,
delete object o.

A command may have an empty set of conditions, i.e., $m = 0$ but it has to perform at least one primitive operation.

Suppose that c is a command, Q a configuration and $a_1, \ldots, a_k$ actual parameters to be used when c is executed. *Configuration Q yields configuration Q'*

under c, notation $C \vdash_{c(a_1,\ldots,a_k)} C'$, where

- $Q' = Q$, if some of the **if**–conditions of c are not satisfied, and
- $Q' = Q_n$, where $Q = Q_0 \vdash_{op'_1} Q_1, \ldots, \vdash_{op'_n} Q_n$, and $op'_1, \ldots, op'_n$ are the substituted versions of primitive operations of c .

As a notational matter, $Q \vdash_c Q'$ means that there exist actual parameters $a_1, \ldots, a_k$ such that $Q \vdash_{c(a_1,\ldots,a_k)} Q'$. Correspondingly, $Q \vdash^\star Q'$ denotes the situation where there exists a sequence of length $n \geq 0$ such that $Q = Q_0 \vdash_{c_1} Q_1 \ldots \vdash_{c_n} Q_n = Q'$ where $c_1, \ldots, c_n$ are commands.

The access matrix model defines precisely the model of computation in terms of the access matrix, commands, primitive operations and configurations. Furthermore, the access rights are the security primitives of this model. The definition of security is based on the concept of leak. Command *c leaks a right r from configuration Q* if c, when run on Q, executes a primitive operation which enters r into a cell of the access matrix which did not previously contain r. Leaks are not necessary a bad thing, as any system that allows sharing of resources will have many leaks. This means that many subjects intentionally leak their rights to *trustworthy subjects*, i.e., to those subjects which are not believed to abuse the given rights.

The initial configuration Q_0 of an access matrix is considered to be secure[1] for a right r if, for a reduced access matrix, there does not exists a configuration Q and a command c such that $Q_0 \vdash^\star Q$ and c leaks r from Q. *A reduced access matrix* can be constructed from the original access matrix by deleting the cells corresponding the trustworthy subjects from the system. As briefly mentioned above the trustworthy subjects may cause definitely legal leaks to the system. Because the definition of secure initial configuration does not distinguish legal and illegal leaks, the possibility of legal leaks must be eliminated before the security of the system is inspected.

Harrison, Ruzzo and Ullman proved important results about the computability of security. They showed that it undecidable whether a configuration of an arbitrary access matrix is secure for a given access right. The authors also proved a theorem about models containing only mono-operational commands. A command is *mono-operational* if it consists of one primitive operation. The theorem says that there exists an algorithm which decides the security of the initial configuration of a given mono-operational access matrix model in respect of a given right.

[1] For the sake of uniformity, terms "secure" and "insecure" are used instead of the original terms "safe" and "unsafe".

A conceptual difficulty related to the access matrix model is that a subject may have a total control of objects. For instance, if a process of an operating system has a right to read the contents of a file then the process may distribute the data of the file to every other object it happens to have write access, no matter how sensitive the data is. The mechanisms enforcing the security regulations of the system have no way to prevent this as long as the process has right to read the sensitive data. The crux of this issue is the fact that the model concentrates on the control of the containers of data regardless the sensitivity of the data, i.e., there are no tools to relate the sensitivity of the data to the allowed set of operations. This observation motivates the research on Denning's information flow model whose basic principles will be introduced in the next subsection.

A lot of work has been done to develop a decidable derivate of the access matrix model. Snyder surveys the classical results of this research in [37] while more recent work on this issue is described in [32, 34, 33].

1.3 Denning's Information Flow Model

As mentioned in the previous subsection, the access matrix model describes the accessing of objects, but not what subjects might do with the information contained in them. The security model introduced by Denning in her work [3] establishes a framework for the analysis of information flows. The following condensed representation of the model is adapted form [3, 6, 4].

An information flow model FM is a structure

$$FM = \langle N, P, SC, \oplus, \leq \rangle \tag{1}$$

where N is a set of *storage objects*, P is a set of *processes*, SC is a set of *security classes*, $\oplus$ is a associative and commutative *class combining operator* on SC, and $\leq$ is a binary *flow relation* on SC.

Processes are the active entities responsible for all information flow in the modelled system. Information *flows* from object a to object b, denoted by $a \Rightarrow b$, whenever information stored in a is transferred, or used to derive information transferred to object b by some process. Each object is bound to a security class of SC; the security class of object a is denoted by $\underline{a}$. The binary class combining operator $\oplus$ models the transformation of security rating in any binary function. Let f be a binary function on the set N. If $c = f(a, b)$, where $a, b, c \in N$ then the security class associated with c is defined to be $\underline{c} = \underline{a} \oplus \underline{b}$. Consequently, the security class of the value of an n-ary function $f(a_1, \ldots, a_n)$ is $\underline{a}_1 \oplus, \ldots, \oplus \underline{a}_n$. The flow relation $\leq$ defines the permitted information flows in the system.

A flow model FM is said to be *secure* if for any objects a, b and for every execution of any sequence of processes the following hold:

$$a \Rightarrow b \text{ only if } \underline{a} \leq \underline{b}. \tag{2}$$

In order to apply the model to the inspection of real systems, the nature of the binding of objects and security classes must be described, the semantics of the processes must be specified, and the flow relation must be accurately defined.

The binding of objects and security classes can be *static* in a sense that the security class associated with an object does not change during the execution of the processes or the binding may be *dynamic*, i.e., the binding may change during the execution of processes. The nature of the binding has a remarkable influence to the way the mechanisms regulating the information flows of a practical system will be implemented.

The key observation presented in [3] is that, under the assumptions to be given below, relation $\leq$ together with two operations forms a lattice on SC. This kind of flow relation was found to adequately describe the information flow policies used in military systems. Furthermore, a flow relation which generates lattice has a properties, notably the transitivity of the relation, which makes the verification and on-line checking of information flow easy. Suppose that for set SC, relation $\leq$ and operator $\oplus$ the following conditions hold:

- $\langle SC, \leq \rangle$ is a partially ordered set,
- SC is finite,
- SC has a lower bound L such that for all $A \in SC$ $L \leq A$,
- $\oplus$ is a least upper bound operator on SC.

These assumptions imply the existence of a greatest lower bound operator on SC, denoted by $\otimes$. The existence of $\otimes$ in turn implies the existence of unique upper bound H on SC. Thus, the structure $\langle SC, \leq, \oplus, \otimes \rangle$ is a lattice. The basic theory of partial orders and lattices can be found, for example, from [11].

Assume that a, b and c are objects of some information flow model FM, there exist information flows $a \Rightarrow b$, $b \Rightarrow c$ in the model, and $\underline{a} \leq \underline{b}$ and $\underline{b} \leq \underline{c}$ holds for the statically bound security classes. Clearly there exists an information flow from a to c because there exist information flow from b to c and the information contained in b is affected by the information in a. Suppose now that the task is to verify whether the information flow from a to c is secure. According to formula (2) $a \Rightarrow c$ is secure only if $\underline{a} \leq \underline{c}$. The answer to the security of this flow can always be found out by inspecting the definition of the relation $\leq$. If, however, the relation $\leq$ induces a lattice as sketched above, the inspection of the definition of $\leq$ is not needed. This is due to the fact that $\leq$ is a partial order and therefore $\leq$ is a transitive relation. Now, because $\underline{a} \leq \underline{b}$ and $\underline{b} \leq \underline{c}$ holds, the transitivity implies that also $\underline{a} \leq \underline{c}$ holds. The significance of this observation becomes evident when the the processes are interpreted as statements of a programming language and the security of sequences of statements is considered.

In the context of Denning's information flow model the security classes, the $\leq$-partial order and the $\oplus$-operator constitute the notion of security whereas the computation is modelled with the processes of the set P.

In her work [6] Denning described how the information flows of a Pascal-like programming language can be analyzed and regulated. A security class is associated with each object of the language, i.e, with every constant, scalar variable, array or file, during the compilation of the program. Every program statement corresponds to a process of the abstract information flow model and is linked with a semantic rule which defines the conditions for the flow caused by the statement to be secure. If the compiler detects an insecure information flow during the compilation, the compilation is aborted. As an example, let's look at the selection statement of the language and the semantic rule associated with it:

$$\begin{array}{lll} \langle stmt \rangle \;\rightarrow\; \textbf{if}\ \langle exp \rangle\ \textbf{then}\ \langle stmt \rangle_1 & \underline{\langle stmt \rangle} = \underline{\langle stmt \rangle}_1 \otimes \underline{\langle stmt \rangle}_2, \\ \qquad\qquad\qquad\qquad\ \ \textbf{else}\ \ \langle stmt \rangle_2 & \text{if not } (\underline{\langle exp \rangle} \leq \underline{\langle stmt \rangle}) \text{ then } \textbf{error}. \end{array}$$

When the condition of the **if**-statement is evaluated there occurs an information flow from the variables of $\langle exp \rangle$ either to the variables of $\langle stmt \rangle_1$ or to the variables $\langle stmt \rangle_2$ depending on the result of the evaluation. This is due to the fact that values of some variables are potentially affected by the result of the evaluation of the condition. The information flow is accepted if the security class associated with $\langle exp \rangle$ is in $\leq$-relation with security classes associated with both $\langle stmt \rangle_1$ and $\langle stmt \rangle_2$. The $\otimes$-operator is used to return the greatest security class c such that both $c \leq \underline{\langle stmt \rangle}_1$ and $c \leq \underline{\langle stmt \rangle}_2$ hold. If now $\underline{\langle exp \rangle} \leq c$ then by the transitivity of the partial order also $\underline{\langle exp \rangle} \leq \underline{\langle stmt \rangle}_1$ and $\underline{\langle exp \rangle} \leq \underline{\langle stmt \rangle}_2$ hold. Otherwise an error is reported.

The compile time mechanism sketched above does not remove the need for the run-time information flow control. For example, a malfunctioning piece of hardware may cause illegal information flows during the execution of verified piece of code. Another major concern is the control of convert channels. *A convert channel* is a path which is not normally intended for information transfer at all, but which could be used to signal some information. It is possible, for instance, to construct a program, which is verified to be secure but which signals some information by using patterns of exceptions which are caused by repeated information flow violations. The difficulties of the control of convert channel information flows are mainly due to the fact that the channels are not explicitly known when the analysis of the system is done. The need for an uniform treatment of both explicit and convert channels was the primary motivation for the introduction of the model based of noninterference which will be studied in the next subsection.

1.4 The Model Based on Noninterference

Goguen and Meseguer introduced a security model based on *noninterference* in their work [9]. In this model the conceptual building block of security is the process's inability to observe the actions of other processes rather than the detection and classification of information flow channels. In principle this approach leads to the uniform treatment of both explicit and convert information flow channels. Apart from the original work, the following presentation of the model is partly based on the description in [10].

Goguen and Meseguer use *a state machine* as the model of computation. A state machine consists of a set S of internal states containing an initial state s_0, a set U of *users*, a set O of outputs, a set C of *state changing commands*, a set R of *read commands* together with a *next state function* $do : S \times U \times C \rightarrow S$ and an *output function* $out : S \times U \times R \rightarrow O$. The defined functions have obvious meanings: $do(s, u, c)$ gives the next state of the system when an user u executes command c in state s and $out(s, u, r)$ returns the result of an user u executing a read command r in state s. An interesting feature of this model of computation is that sets S, U, C and R are not required to be finite.

A computation of a state machine is described by a history. *A history* is a sequence $(u_1, c_1) \dots (u_n, c_n)$ where each $u_i \in U$ and $c_i \in C \cup R$, i.e., a history is a member of the set $(U \times (C \cup R))^*$. At the beginning of each history the system is supposed to be in its distinguished initial state s_0. Let $[[w]]$ denote the system state after the execution of the history w.

The security properties of a state machine are expressed with a set of noninterference assertions. Each *noninterference assertion* expresses the fact that a group of users executing a subset of state transition commands does not interfere with, i.e., can not be detected by, another group of users executing a set of read commands. More specifically, let G, G' be sets of users, A a set of state transition commands, and B a set of read commands. Then the noninterference assertion $G, A :| G', B$ denotes the situation where for each sequence $w \in (U \times C)^*$, for each $v \in G'$ and for each $r \in B$

$$out([[w]], v, r) = out([[p_{G,A}(w)]], v, r), \tag{3}$$

where $p_{G,A}(w)$ is a sequence obtained from w by eliminating all occurrences of pairs (u, c) with $u \in G$ and $c \in A$. Intuitively speaking, formula (3) means that as far as any user $v \in G'$ can tell by executing read commands from B, everything looks as if the users in G had never executed any commands from A.

To illustrate the usage of the scheme sketched above, let's consider a system where the users are divided into three disjoint sets U, S and T. The associated security policy states that information may flow from U to S, S to T and U to T but not in any other direction. This flow policy can expressed by stating that assertions $S \cup T, C :| U, R$ and $T, C :| S, R$ must hold where C is the set of state transition commands and R is the set of read commands. Intuitively this means that the users of group U can not "see" the steps of computations of the users of groups S and T, and consequently the users from group S can not "see" the steps of computations of the users from group T.

Sutherland presented in his work [38] a model which defines the notion of security more precisely than the original noninterference model. Another important contribution of Sutherland's work is that an ordinary finite-state automaton is used as the model of computation. This is certainly an advantage in the context of practical applications. McCullough describes how CSP instead the formalism presented by Goguen and Meseguer can be used to express the noninterference assertions [25].

1.5 A Petri Net Based Information Flow Model

Varadharajan described in his work [40] a security model in which the notion of security resembles the notion of security in Denning's information flow model. This model, which was later called *an information flow secure net model* or IFS-net model [41], uses a class of Petri nets as its model of computation.

Petri nets are a mathematical model of computation in which a particular attention has been paid to the representation of concurrently occuring atomic events [28, 30, 19]. The static structure of a Petri net can be viewed as a bipartite graph. In the context of P/T-nets [30], the elements of the two disjoint sets of are called *places* and *transitions*. The purpose of the places is to preserve the distributed state of a net whereas the transitions model atomic actions. A state of a net is described by a *marking* which associates zero or more *tokens* with each place of the net. *The initial marking* is the initial state of the net. A marking of the net is changed via *firings of transitions*. A necessary condition for a transition firing is that every *input place*, i.e., every place adjacent to the transition, contains at least one token. When transition fires it removes tokens from its input places and submits tokens to its *output places*, i.e., places adjacent from the transition.

One way to classify different versions of Petri nets is to consider them as *low-level nets* or *high-level nets*. The difference of the nets of these classes is very much like the difference between an assembly language and a high-level programming language, say Pascal. There is not necessarily difference in the computational power of the two languages, but due to the advanced control structures and data-abstraction mechanisms the use of the high-level programming language is much effective and less error-prone the the use of an assembly language. The high-level Petri nets typically offer the notion of structured tokens which means that tokens are able to carry data items. Furthermore, the firing of a transition is not only regulated by the existence of tokens in the input places, but also by a logical formula linked with the transition.

Elementary net systems [39] and and P/T-nets [30] are representatives of low-level nets whereas PrT-nets [7] and Coloured Petri nets [17] are examples of high-level Petri net classes.

In Varadharaja's IFS-net model the places of nets represent *channels* which transfer *messages*, that is tokens, between *atomic processes* represented by transitions. Every message carries *a data item*. Furthermore, *a security class* is linked to every message. A set of security classes is also associated with every channel of the model. When a transition fires it emits an output token whose data item is a function of the data items of the input tokens and whose security class is also a function of the security classes of input tokens. *An information flow policy* is associated with every IFS-net in the form of partial order on the set of security classes.

The notion of security in the IFS-net model is based on the concepts of *a secure marking* and *a secure information flow*. A marking of an IFS-net is secure if the security classes of tokens conform with the sets of security classes associated with the places in which the token resides. An information flow may occur between transitions. The definition of the secure information flow is based on the notion of a *secure transition*. Roughly speaking, a transition is secure if its firing, at every secure marking reachable from the initial marking, obeys the information flow policy and leads to another secure marking.

While the IFS-net formalism is an interesting alternative for the modelling and verification of security properties, there exists a major problem which makes the practical evaluation of this approach difficult. The IFS-net formalism is not based on any pure class of Petri nets described in the literature, but on a class of nets which is created by equipping a class of low level Petri nets, Peterson's Petri nets [28], with security specific primitives. This approach has two serious implications. Firstly, the theoretical results and the practical experiences of the standard classes of Petri nets are not applicable to IFS-nets. Secondly, the existing software tools are not directly suitable to assist the analysis of these nets.

1.6 The Aim and the Structure of the Work

The ultimate goal of this work is to develop a methodology which helps one to evaluate the practical significance of IFS-nets in the modeling and verification of secure systems. To achieve this goal, it will be shown how a PrT-net corresponding to an IFS-net model can be constructed and how the various security features of the IFS-net model are reflected in the reachability graph of the PrT-net.

The rest of the work is structured as follows. In section 2 the formal definition of IFS-nets is given together with the description of the notion of security. Section 3 is devoted to the review of PrT-nets. Section 4, which is the core of the work, describes the mapping which is used to transform an IFS-net into a corresponding PrT-net. This section also contains results which bind the notion of security to the properties of the reachability graph of the constructed PrT-net. Section 5 begins by a discussion on the upgrading and downgrading of data. After this a simplified yet practically relevant example related to databases is described. The ideas presented in the earlier sections are exploited and the IFS- and PrT-net models of the example system are constructed. At the end of the section results concerning the security of the modelled system, obtained with a PrT-net software tool, are described. Finally, section 6 is reserved for concluding remarks.

2 Information Flow Security Nets

The following formal introduction to IFS-nets, obtained from [40, 41], is divided into two parts. In the first part the static structure of nets, the dynamic behavior of nets and the notion of secure information flow are introduced. A detailed example is also included. In the second part several technical details of IFS-nets are reviewed and their differences from the solutions found in the more traditional approaches are considered.

2.1 The Formal Definition of the Model

Before we proceed to the primary topic of this subsection, it is advisable to throw a glance at the foundations of the IFS-net formalism. A more complete treatment of the basics can be found from [28]. *A bag (or multi-set) over a domain D* is a function $B_{\mathrm{D}} : D \rightarrow \mathbf{N}$ where D is a non-empty set and $\mathbf{N}$ is the set of natural numbers. Intuitively speaking, the concept of bag extends the notion of set by allowing multiple occurrences of the same items. The standard predicates and operations on sets, such as $\in$, $=$, $\subset$, $\cap$ and $\cup$, can be extended to bags in obvious way. Furthermore, the extension of the concept of power set, i.e., the set of all bags over the domain D is denoted by $\mathcal{P}(B_{\mathrm{D}})$. The cardinality of a bag is defined as $\#B_{\mathrm{D}} = \sum_{d \in D} B_{\mathrm{D}}(d)$. As an example of the notation to be used to described concrete bags in the subsequent chapters consider a domain $A = \{a, b, c\}$ and a bag $G = B_{\mathrm{A}}$ such that $G(a) = 1$, $G(b) = 2$ and $G(c) = 1$. The contents of the bag can be expressed conveniently by expanding the notion of set with integer coefficients: $G = \{a, 2b, c\}$.

A Petri net structure C is a quadruple $C = \langle P, T, I, O \rangle$ where $P = \{p_1, \ldots, p_n\}$, $n \geq 0$ is a finite set of *places* and $T = \{t_1, \ldots, t_m\}$, $m \geq 0$ is a finite set of *transitions* such that $P \cap T = \emptyset$. Furthermore, $I : T \rightarrow \mathcal{P}(B_P)$ is the *input function* and $O : T \rightarrow \mathcal{P}(B_P)$ is the *output function* of the Petri net structure. *A marking* $\bar{M}$ *of a Petri net structure* $C = \langle P, T, I, O \rangle$ is a function $\bar{M} : P \rightarrow \mathbf{N}$. Intuitively, the marking describes how many *tokens* have been assigned to the places of the Petri net structure. *A Petri net* N is a pair $N = \langle C, \bar{M} \rangle$ where C is a Petri net structure and $\bar{M}$ is a marking.

A transition t of a Petri net structure C with a marking $\bar{M}$ is *enabled* iff for all $p \in I(t)$ $\bar{M}(p) \geq (I(t))(p)$, that is, the number of tokens in each input place of t is equal or greater than the number of arcs connecting the place and the transition t. A transition t may *fire* at $\bar{M}$ iff t is enabled at $\bar{M}$. When a transition fires it removes all of its enabling tokens from input places and deposits one or more tokens into each output place. The firing of the enabled transition t at a marking $\bar{M}$ leads to a new marking $\bar{M}_1$ in which the following equation holds: $\forall p \in P : \bar{M}_1(p) = \bar{M}(p) - (I(t))(p) + (O(t))(p)$.

Definition 2.1 *An information flow security net structure* is a 10-tuple

$$IC = \langle C, Tokens, Data, SC, OP, CLSFN, TF, transop, transcls, psec\rangle$$

where

1. $C = \langle P, T, I, O\rangle$ is a Petri net structure, where $\forall t \in T \; \forall p \in I(t)$: $(I(t))(p) = 1$,
2. *Tokens* is a set of *tokens*,
3. *Data* is a set of *data items*,
4. *SC* is a partially ordered set of *security classes*,
5. *OP* is a set of functions $\mathcal{P}(Data) \rightarrow Data$,
6. *CLSFN* is a set of functions $\mathcal{P}(SC) \rightarrow SC$,
7. *TF* is a set of functions such that $tf_t \in TF$ iff $tf_t : Tokens^n \rightarrow Tokens$, $t \in T$, and $n = \#I(t)$,
8. *transop* is a function $T \rightarrow OP$,
9. *transcls* is a function $T \rightarrow CLSFN$,
10. *psec* is a function $P \rightarrow \mathcal{P}(SC)$.

□

The Petri net structure in definition 2.1 describes the static configuration of the modelled system. The set *Tokens* contains all the tokens to be met when the dynamic behavior of the model is inspected and the set *Data* contains the data items which can be linked to the tokens. The elements of the set *SC* will describe the sensitivity of the data carried by tokens and the sensitivity of the places. In addition to the partial order requirement, the original definition of IFS-nets states that the set *SC* and the associated partial order $\leq$ forms a lattice. This lattice requirement has not been included in the formal definition here because there may exist interesting partially ordered sets of security classes which are not lattices. The set *OP* contains all possible operations dedicated to the processing of data items and the set *CLSFN* contains all possible operations for the processing of the security classes.

For a transition t, the functions of the set *TF* and the functions *transop* and *transcls* define the relation between the input and output tokens as well as the relation between the data and security classes of these tokens. The set *TF* contains exactly one function for each transition of an IFS-net. Each function defines the token emitted by the transition when a particular combination of tokens is given as an input. The function *transop* relates an operation which modifies the data of incoming tokens in a firing process to each transition. Respectively, the function *transcls* associates a function which determines the security class of emitted tokens with each transition. The definitions of *transop* and *transcls* given in the definition 2.1 are actually not those to be used in the following chapters. The problems and redefinitions of these functions are represented in subsection 2.2.

A set of security classes is linked to each place of a net; formally this is done by function *psec*. The security classes of a place describe the sensitivity of the channel. They provide a grip for reasoning when the sensitivity issues are involved in the functioning of a system. The security policy of a system can be specified by defining the exact nature of the partial order of the security classes and the set of functions which describe how the security classes of emitted tokens are computed from the security classes of incoming tokens in transitions.

As with original Petri nets, the IFS-net structure defines only the static nature of the inspected system. In order to describe the dynamics of the system, the concept of marking of IFS-net structures is needed. In addition to the distribution of tokens in the net, the marking must describe the association of the data and security classes with tokens. To accomplish this task tree more functions must be introduced: function $tk : P \rightarrow \mathcal{P}(B_{Tokens})$ describes the distribution of tokens, $tokdata : Tokens \rightarrow Data$ the data associated with tokens and $toksec : Tokens \rightarrow SC$ the security classes of tokens, where $\mathcal{P}(B_{Tokens})$ is the set of all bags of tokens. These functions are dynamic in a sense that their extensions may change from time to time according to the rules to be described below. Now, *the marking $\bar{M}$ of an IFS-net structure* is a function from the set of places and its value is a quadruple

$$\begin{aligned} \bar{M}(p) \quad = \quad & \langle tk(p), \#tk(p), \{toksec(tok) | tok \in tk(p)\}, \\ & \{tokdata(tok) | tok \in tk(p)\}\rangle \end{aligned} \tag{4}$$

for all $p \in P$. Whenever the connection between the dynamic functions *tk*, *tokdata* and *toksec* and the associated marking is not clear, suitable sub- or superscripts are added to the marking and to these functions. It should be noted that the given definition of the marking has a problem which is discussed in subsection 2.2.

Let IC be an IFS-net structure and $\bar{M}$ a marking of IC. A transition t is *enabled at marking $\bar{M}$* iff all its input places have at least one token, i.e.,

$$\forall p \in I(t) : \#tk(p) \geq (I(t))(p) \Leftrightarrow \forall p \in I(t)\ \exists tok \in Tokens : tok \in tk(p). \tag{5}$$

Before the formal definition of the dynamics of IFS-nets can be given, the concept of securely enabled transition must be given. In order to ease this step some preliminary definitions are needed.

In the IFS-net formalism, the notion of security is based on the secure receiving, secure transforming and secure sending of tokens. To simplify the formal presentation of these activities three predicates are defined. Let $lub(S)$ denote the least upper bound of the set $S \subseteq SC$. The least upper bound of S is an element $s \in SC$ such that s is an upper bound of S and for any other upper bound s' of S, $s \leq s'$ holds. Consequently, an upper bound of S is an element e of SC such that for all $e' \in S$, $e' \leq e$. Now, let IC be an IFS-net structure and $\bar{M}$ marking of it. Then, at the marking $\bar{M}$:

$$\begin{aligned} & securereceive(tok, t, p) = TRUE\ iff \\ & p \in I(t) \wedge tok \in tk(p) \wedge toksec(tok) \in psec(p), \end{aligned} \tag{6}$$

$$\begin{aligned} & securesend(tok, p, t) = TRUE\ iff \\ & p \in O(t) \wedge toksec(tok) \in psec(p), \end{aligned} \tag{7}$$

$$\begin{aligned} & securetransform(tok, t) = TRUE\ iff \\ & \exists tok_1 \in tk(p_1) \ldots \exists tok_n \in tk(p_n) : tok = tf_t(tok_1, \ldots, tok_n) \wedge \\ & toksec(tok) = (transcls(t))(\{toksec(tok_1), \ldots, toksec(tok_n)\}) \wedge \\ & lub(\{toksec(tok_1), \ldots, toksec(tok_n)\}) \leq toksec(tok), \\ & \cup_{i=1}^{n} p_i = I(t). \end{aligned} \tag{8}$$

An atomic process may securely receive a message if the security class of the message conforms with the sensitivity of the conveying channel. This intuition is formalized by predicate *securereceive*. Respectively, predicate *securesend* states the fact that an atomic process may securely send a message via a channel if the security rating of the message follows the sensitivity of the channel. The model at hand allows also situations where an atomic process modifies the security classes of messages. Predicate *securetransform* describes the cases where the altering is considered to be secure. Formula (8) essentially says that the modification of the security class is secure if the security class of the emitted token is greater than the least upper bound of the security classes of the incoming tokens.

Transition t is *securely enabled at* $\bar{M}$ iff t is enabled at $\bar{M}$ and

$$\begin{aligned} & \forall p \in I(t)\ \forall tok \in tk(p) : \\ & \quad securereceive(tok, t, p) = TRUE \wedge \\ & \forall tok_1 \in tk(p_1), \ldots, \forall tok_n \in tk(p_n) : \\ & \quad securetransform(tf_t(tok_1, \ldots, tok_n), t) = TRUE \end{aligned} \tag{9}$$

where $\cup_{i=1}^{n} p_i = I(t)$.

Definition 2.2 Let IC be an IFS-net structure, $\bar{M}_1$ a marking and t a transition of IC. Let t be enabled at $\bar{M}_1$. *The marking $\bar{M}_2$ is reachable from $\bar{M}_1$ by firing t* if the following constraints hold:

1. $\forall p \in I(t) : tk_2(p_i) = tk_1(p_i) - \{tok_i\},\ 1 \leq i \leq n,$
2. $\forall p \in O(t) : tk_2(p) = tk_1(p) \cup \{tok\},$
3. $\forall p \notin (I(t) \cup O(t)) : tk_2(p) = tk_1(p),$

where $tok = tf_t(tok_1, \ldots, tok_n)$, $tok_i \in tk_1(p_i)$, $\cup_{i=1}^{n} p_i = I(t)$ and furthermore $toksec(tok) = (transcls(t))(\{toksec(tok_1), \ldots, toksec(tok_n)\})$, $tokdata(tok) = (transop(t))(\{tokdata(tok_1), \ldots, tokdata(tok_n)\})$. □

A marking $\bar{M}_2$ is immediately reachable from $\bar{M}_1$ iff there exists a transition t such that $\bar{M}_2$ is reachable from $\bar{M}_1$ by firing t (notation $\bar{M}_1 \xrightarrow{t} \bar{M}_2$). *A marking $\bar{M}$ is reachable from $\bar{M}$* iff there exists a sequence of transitions $s = t_1 t_2 \ldots t_k$ and a sequence of markings $\bar{M}_1 \bar{M}_2 \ldots \bar{M}_{\mathrm{k}+1}$, $k \geq 1$ such that $\bar{M}_1 = \bar{M}$, $\bar{M}_{\mathrm{k}+1} = \bar{M}'$ and every marking $\bar{M}_{\mathrm{i}+1}$ is reachable from $\bar{M}_{\mathrm{i}}$ by firing transition t_i, $1 \leq i \leq k$ (notation $\bar{M} \xrightarrow{s} \bar{M}'$).

An IFS-net structure describes the static configuration of the system at hand. The dynamic nature of the system is described by the notion of reachability and by fixing the initial state of the inspected system. The static and dynamic parts are formally combined in the definition of the IFS-net.

Definition 2.3 *An IFS-net* is a pair $IN = \langle IC, \bar{M}_0 \rangle$, where IC is an IFS-net structure and $\bar{M}_0$ is a marking of IC. □

$\bar{M}$ is said to be a marking of $IN = \langle IC, \bar{M}_0 \rangle$ iff $\bar{M}$ is a marking of IC. The reachability set and the set firing sequences represent explicitly the dynamics of an IFS-net.

Definition 2.4 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net. *The reachability set of IN is the set $RS_{IN} = \{\bar{M} \mid \bar{M}_0 \xrightarrow{s} \bar{M},\ s = t_1 t_2 \ldots t_k,\ t_i \in T,\ |s| \geq 0\}$. The set of firing sequences of IN is the set $FS_{IN} = \{s \mid \bar{M}_0 \xrightarrow{s} \bar{M},\ \bar{M} \in RS_{IN}\}$.* □

In the IFS-net formalism the notion of security is based on the concepts of secure marking and secure transition.

Definition 2.5 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net and $\bar{M}$ a marking of IN. *A marking $\bar{M}(p)$ of place p is secure* iff $\forall tok \in tk(p) : toksec(tok) \in psec(p)$. Consequently, *the marking M is secure* iff $\forall p \in P : \bar{M}(p)$ is secure. □

Definition 2.6 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net, $\bar{M}_1$ and $\bar{M}_2$ markings of IN and t transition of IN. Furthermore, let $\bar{M}_2$ be immediately reachable from $\bar{M}_1$ by firing t. *The firing of t at $\bar{M}_1$ is secure* iff $\forall p \in \ O(t)$: *securesend*$(tok, p, t) = TRUE$ at $\bar{M}_2$ where $tok = tf_t(tok_1, \ldots, tok_n)$, $tok_i \in tk_i(p_i)$, $\cup_{i=1}^{n} p_i = I(t)$. □

Definition 2.7 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net and t its transition. *Transition t is secure* iff $\forall \bar{M} \in RS_{IN}$: If t is enabled at $\bar{M}$ then it is securely enabled at $\bar{M}$ and the firing of t is secure. Let $\bar{M}_2$ be reachable from $\bar{M}_1$ by firing the transition sequence $s = t_1 t_2 \ldots t_k$, $k \geq 1$. *The sequence s is secure* iff each transition t_i is secure. □

A marking of a place is secure if the security classes of the tokens occuring in the place conform with the sensitivity of the place. The security class of a token conforms with the sensitivity of the place if the security class is a member of the set which is associated with the place by the function *psec*. A transition is considered to be secure if, at every marking reachable from the initial marking, whenever the transition is enabled, the markings of the input places are secure, the transition computes the security class of the output token according to the predefined security policy, and the new markings of the output places are secure.

The notion of security is characterized by two interrelated concepts in the connection of IFS-nets. When an IFS-net is information flow secure (or simply secure), a general statement about the security of all reachable markings and all possible firing sequences can be made. On the other hand, if an information flow from a transition to another transition is secure then only conclusions related to the transition sequences transferring tokens between the transition are justified. Let's consider the expression of information flow security first.

Definition 2.8 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net. *IN is information flow secure* iff $\bar{M}_0$ is secure and every transition sequence $s \in FS_{IN}$ is secure. □

This definition itself does not seem to be very practical aid for the verification of the security of IFS-nets. The definition does not directly bind the security of the net to the security properties of reachable markings. The following theorem clarifies this connection.

Theorem 2.9 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net. If IN is information flow secure then every marking $\bar{M} \in RS_{IN}$ is secure.

Proof: Let $IN = \langle IC, \bar{M}_0 \rangle$ be an information flow secure IFS-net. It follows from definition 2.8 that $\bar{M}_0$ is secure and each transition sequence $s \in FS_{IN}$ is secure. If $RS_{IN} = \{\bar{M}_0\}$ then the claim trivially follows, so let's suppose that the cardinality of set RS_{IN} is ≥ 2. Let's suppose that $\bar{M}_1, \bar{M}_2 \in RS_{IN}$

are markings such that $\bar{M}_1 \xrightarrow{t} \bar{M}_2$ and $\bar{M}_2$ is not secure. By definition 2.7 t can not be secure. This is implies that there exists a transition sequence $s \in FS_{IN}$ such that s is not secure. But by the initial assumption every transition sequence of FS_{IN} is secure $\Rightarrow$ Contradiction $\Rightarrow$ Every marking of RS_{IN} is secure. $\square$

Having defined the meaning of security for IFS-nets it is time to examine the idea of information flow. The notion of information flow is based on the structural properties of the underlying net as well as its dynamics. Under certain circumstances, information is considered to flow between transitions via paths. *A path of an IFS-net* is an alternating sequence of places and transitions $p_1t_1 \ldots p_kt_kp_{k+1}$, $k \geq 1$ such that $p_i \in I(t_i)$, and $p_{i+1} \in O(t_i)$, $1 \leq i \leq k$. Moreover, *the transition sequence s corresponding to path c* is the sequence $s = t_1 \ldots t_k$, $k \geq 1$.

Definition 2.10 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net and $t, \bar{t}$ transitions of IN. *There exists an information flow from t to $\bar{t}$ along path c* iff $c = p_1t_1 \ldots p_kt_kp_{k+1}$, $t_1 = t$, $t_k = \bar{t}$, $k \geq 1$, and there exist markings $\bar{M}, \bar{M}' \in RS_{IN}$ such that $\bar{M} \xrightarrow{s} \bar{M}'$ where s corresponds to path c. *There exists an information flow from t to $\bar{t}$* iff there exists an information flow from t to $\bar{t}$ along some path c. $\square$

Definition 2.11 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net and $t, \bar{t}$ transitions of IN such that there exists an information flow from t to $\bar{t}$ along path c. *The information flow from t to $\bar{t}$ along path c is secure* iff for every pair of markings $\bar{M}, \bar{M}' \in RS_{IN}$ such that $\bar{M} \xrightarrow{s} \bar{M}'$, $s = t_1t_2 \ldots t_k$ corresponds to path c, $k \geq 1$, $t_1 = t$, $t_k = \bar{t}$, and for every transition t_i of s the following conditions hold:

1. Transition t_j is securely enabled at $\bar{M}_{\mathrm{j}}$ and marking $\bar{M}_{\mathrm{j+1}}$ is reached by firing transition t_j at marking $\bar{M}_{\mathrm{j}}$, $1 \leq j \leq k$.

2. The firing of transition t_i is secure.

Furthermore, *the information flow from t to $\bar{t}$ is secure* iff the information flow from t to $\bar{t}$ is secure along every path c from t to $\bar{t}$. $\square$

As can be seen from definition 2.10 the possible information flow paths are explicitly defined by the Petri net structure of an IFS-net. If there is a path (in graph theoretical sense) between transition then there may occur an information flow. The information flow actually occurs only if there also exists a firing sequence which corresponds to the path.

The following lemma helps us to proof a theorem which clarifies the relation of the information flow security and the secure information flow between transitions.

Lemma 2.12 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net such that every transition sequence of FS_{IN} is secure and let $\bar{M}_1, \bar{M}_2 \in RS_{IN}$ be markings such that $\bar{M}_1 \xrightarrow{s} \bar{M}_2$. Then the transition sequence s is also secure.

Proof: Let's assume that s is not secure. By definition 2.7 s must contain a transition t which is not secure. Because $\bar{M}_1 \in RS_{IN}$ there exists a transition sequence s_1 such that $\bar{M}_0 \xrightarrow{s_1} \bar{M}_1$. Furthermore, $\bar{M}_2 \in RS_{IN}$ and $\bar{M}_1 \xrightarrow{s_1} \bar{M}_2$ imply that there exists a sequence $s_2 \in FS_{IN}$ of transitions such that $s_2 = s_1 s$. Because s_2 is a member of the set FS_{IN} it is also a secure transition sequence, i.e., every transition of s_2 is secure. But s contains at least one insecure transition so that s_2 can not be secure $\Rightarrow$ Contradiction $\Rightarrow$ s is secure. $\square$

With lemma 2.12 it is now easy to prove the following important theorem.

Theorem 2.13 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net and $t, \bar{t}$ transitions of IN such that there exists an information flow from t to $\bar{t}$. If IN is information flow secure then the information flow from t to $\bar{t}$ is secure.

Proof: Let IN, t and $\bar{t}$ be defined as above. By definition 2.8, $\bar{M}_0$ is secure and every transition sequence of FS_{IN} is secure. Consider a path c from t to $\bar{t}$. Let s be a transition sequence corresponding to c and $\bar{M}, \bar{M}' \in RS_{IN}$ markings such that $\bar{M}_1 \xrightarrow{s} \bar{M}_2$. The transition sequence s is secure by lemma 2.12. By definition 2.7, every transition of s is secure which in turn implies that if a transition of s is enabled at a marking reachable from $\bar{M}$ then it is also securely enabled and the firing of the transition is secure. But now, by definition 2.11, the information flow from t to $\bar{t}$ along c must be secure. Because c was an arbitrary path from t to $\bar{t}$, the information flow from t to $\bar{t}$ is secure along every path. Thus information flow from t to $\bar{t}$ is secure. $\square$

Example 2.14 Consider an IFS-net $IN = \langle IC, \bar{M}_0 \rangle$. Let the Petri net structure $C = \langle P, T, I, O \rangle$ of IC be characterized by the sets $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3\}$ and by the bags $I = \{(t_1, p_1), (t_2, p_1), (t_3, p_2)\}$ and $O = \{(t_1, p_2), (t_2, p_2), (t_3, p_3)\}$. Let the set of tokens be $Tokens = \{tok_1, tok_2\}$, the set of data items $Data = \{d_1, d_2, d_3, d_4\}$ and the set of security classes $SC = \{U, C, S\}$. Furthermore, let the partial ordering of SC be defined by $U \leq C$, $C \leq S$ and $U \leq S$. Let the set OP contain functions fd_1, fd_2, and fd_3 whose value for the singleton sets of $Data$ is $fd_i(\{d\}) = fd_i(d) = d_i$ iff $d = d_i$ and d_4 for other subsets of $Data$ $i \in \{1, 2, 3\}$. Respectively, the functions of the set $CLSFN = \{fs_1, fs_2, fs_3\}$ are defined as follows: $fs_i(\emptyset) = \emptyset$ and $fs_i(E) = S$ for any subset $E \subseteq \{U, C, S\}$ where $i \in \{1, 3\}$, and $fs_2(\{C\}) = C$, $fs_2(\{S\}) = S$, $fs_2(\{U, C\}) = C$, $fs_2(\{U, S\}) = S$, $fs_2(\{C, S\}) = S$, and finally $fs_2(\{U, C, S\}) = S$. Let the set TF contain functions tf_{t_1}, tf_{t_2} and tf_{t_3} such that for all $tok \in Tokens : tf_{t_i}(tok) = tok$, $i \in \{1, 2, 3\}$. Furthermore,

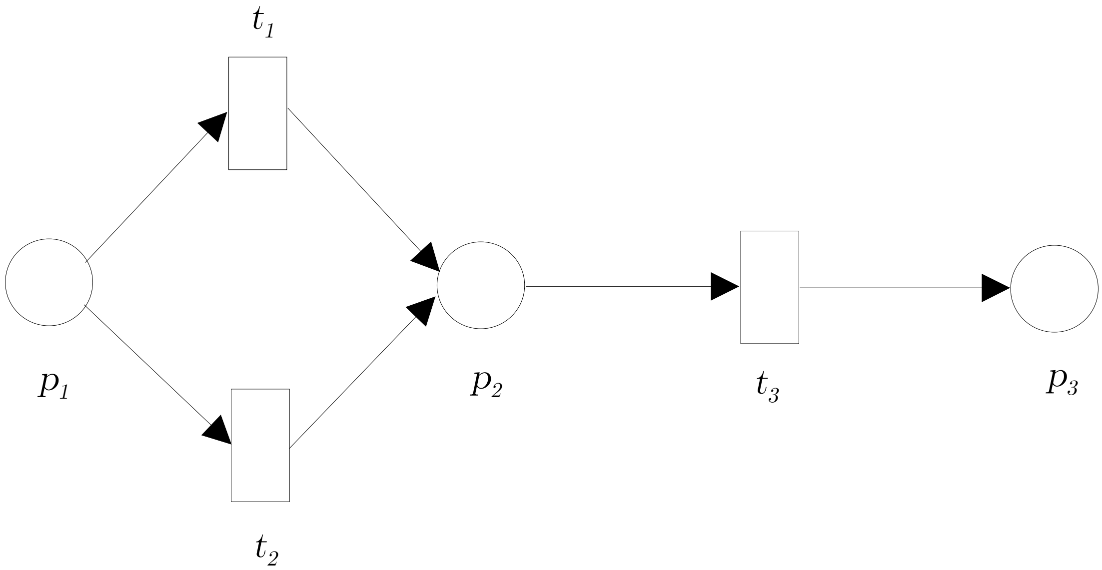

Figure 1: A sample IFS-net

let the function *transop* associate the functions fd_1, fd_2 and fd_3 to the transitions t_1, t_2 and t_3 respectively, and let the function *transcls* associate the functions fs_1, fs_2 and fs_3 to the corresponding transitions t_1, t_2 and t_3. To conclude the description of the static structure of the net, the security ratings of the places must be given. Let the function *psec* link the sets $\{C\}$, $\{S\}$ and $\{S\}$ to the places p_1, p_2 and p_3 respectively.

In order to define the initial marking $\bar{M}_0$ of the net, the data carried by the tokens, the security classes associated with tokens as well as the distribution of the tokens must be defined. To accomplish this task, let the function $tokdata_0$ link the data item d_1 to each token of the set *Tokens* and let the function $toksec_0$ associate the security class C with every token of the set *Tokens*. Moreover, let's suppose that the function tk_0 defines the distribution of tokens as follows: $tk_0(p_1) = \{tok_1\}$ and $tk_0(p_2) = tk_0(p_3) = \emptyset$. By equation (4) the initial markings of the places are $\bar{M}_0(p_1) = \langle\{tok_1\}, 1, \{C\}, \{d_1\}\rangle$, $\bar{M}_0(p_2) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$ and $\bar{M}_0(p_3) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$. The graphical representation of the net is given in figure 1.

Transitions t_1 and t_2 are clearly enabled at $\bar{M}_0$ since the place p_1 is the only input place for these transitions and p_1 contains a token. Consider a situation where t_1 fires. The firing of t_1 removes tok_1 from p_1 and injects a token to the place p_2. By the definition of tf_{t_1}, p_2 receives the token $tok_1 = tf_{t_1}(tok_1)$. The data item linked to the injected token is $d_1 = fd_1(d_1)$. Correspondingly, the security class of the tok_1 in p_2 is $fs_1(\{C\}) = S$. Thus, the firing of t_1 at $\bar{M}_0$ leads to marking $\bar{M}_1$ such that $\bar{M}_1(p_1) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$, $\bar{M}_1(p_2) = \langle\{tok_1\}, 1, \{S\}, \{d_1\}\rangle$ and $\bar{M}_1(p_3) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$. On the other hand, the firing of t_2 at $\bar{M}_0$ leads to marking $\bar{M}_2$, $\bar{M}_2(p_1) = \bar{M}_2(p_3) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$ and $\bar{M}_2(p_2) = \langle\{tok_1\}, 1, \{C\}, \{d_4\}\rangle$. Again, transition t_3 is enabled at $\bar{M}_1$. The firing of t_3 at $\bar{M}_1$ leads to marking $\bar{M}_3$ where $\bar{M}_3(p_1) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$, $\bar{M}_3(p_2) = \langle\emptyset, 0, \emptyset, \emptyset\rangle$ and $\bar{M}_3(p_3) = \langle\{tok_1\}, 1, \{S\}, \{d_4\}\rangle$. The firing of t_3 at $\bar{M}_2$ in turn leads to a marking in which p_1 and p_2 are associated with marking $\langle\emptyset, 0, \emptyset, \emptyset\rangle$ and p_3 is associated with a marking $\langle\{tok_1\}, 1, \{S\}, \{d_4\}\rangle$ i.e. the firing of t_3 at $\bar{M}_2$ leads to $\bar{M}_3$. The set of possible firing sequences of IN is $FS_{IN} = \{t_1t_3, t_2t_3\}$.

Consider the security properties of the inspected net. The examination of the reachable markings reveals that markings $\bar{M}_0$, $\bar{M}_1$ and $\bar{M}_3$ are secure whereas markings $\bar{M}_2$ is not secure. The insecurity of $\bar{M}_2$ is due to the fact that the security class of the token in the place p_2 does not conform with the security rating of p_2; the security class associated with the token is C but the sensitivity of the place p_2 is $\{\{S\}\}$ and $\{C\} \not\in \{\{S\}\}$.

Transitions t_1 and t_2 are securely enabled at $\bar{M}_0$ and t_3 is securely enabled at $\bar{M}_1$ but transition t_3 is not securely enabled at marking $\bar{M}_2$. Transition t_3 always transforms the security class of the incoming token so that the predicate *securetransform* is true but the insecurity of the marking of the input place indicated by the predicate *securereceive* prevents transition t_3 to be securely enabled at the $\bar{M}_2$. On the other hand, the firing of t_3 is secure at $\bar{M}_1$ and $\bar{M}_2$ but the firing of t_2 at $\bar{M}_0$ is not secure. The insecurity of the firing of t_2 is implied by the predicate *securesend* not being true. Because $\bar{M}_0, \ldots, \bar{M}_3$ are the only reachable markings of the net, the previous observations lead to the conclusion that t_1 is a secure transition while t_2 and t_3 are not. This is due to the fact that there exists markings reachable from $\bar{M}_0$ such that the firing of t_2 is not secure and such that t_3 is not securely enabled. By definition 2.7 these conditions prevent transition t_2 and t_3 of the net from being secure.

Because there exist paths from the transition t_1 to t_3 and t_2 to t_3 and

$$\bar{M}_0 \xrightarrow{t_1} \bar{M}_1 \xrightarrow{t_3} \bar{M}_3 \text{ and } \bar{M}_0 \xrightarrow{t_2} \bar{M}_2 \xrightarrow{t_3} \bar{M}_3, \tag{10}$$

there certainly exists information flows from t_1 to t_3 and t_2 to t_3. By utilizing the formal machinery established above, it is possible to see that at least one of the information flows is insecure. Because there exist insecure markings reachable from $\bar{M}_0$, the counterpositive interpretation of theorem 2.9 indicates that the net at hand is not information flow secure. Definition 2.8 then implies that there exists at least one insecure transition sequence in the set RS_{IN}. There exists indeed only one insecure transition sequence, namely t_2t_3; this is due to the fact that t_2 is the only insecure transition of the net. But then the information flow from t_2 to t_3 must also be insecure. On the other hand the information flow from t_1 to t_3 is secure because the transition sequence t_1t_3 is secure. □

2.2 Some Observations on IFS-nets

The definition of IFS-nets restricts the multiplicity of the arcs to one, i.e., at most one arc may link a place and a transition of an IFS-net. This means that the firing of a transition removes only one token from every input place. Varadharajan relates this restriction to the modelling of multilevel channels [40, pp. 54]. *A multilevel channel* is a channel which is able to transmit messages with different security ratings. In practical engineering the need for multiple inputs or outputs for atomic processes is so common that extra effort spent for the search of a roundabout methods would be acceptable. Unfortunately, it is not very straightforward to see the exact connection between the modeling of multilevel channels and the restriction related to the multiplicity of arcs.

A brief examination of the function *transcls* in definition 2.1 reveals that the security rating of a token emitted in a firing process only depends on the security classes of the incoming tokens. More specifically, the security rating of the emitted token is not affected neither the identities nor the data carried by the incoming tokens. Furthermore, the model does not establish explicit connection between the operation manipulating the incoming data and the operation changing the security rating of the incoming tokens. In this respect the IFS-net model differs from the classical information flow based security models. For example, Denning's information flow model [3, 6] gives exact rules for the security class of the result of every atomic computation which described by the statements of a programming language.

The technical approach adopted in the binding of data to tokens differs from the approaches found in the mainstream Petri net formalisms. For example, PrT-nets, which will be described later in this work, and Coloured Petri nets [17, 18] consider tokens merely as tuples of data items while IFS-net formalism gives an unique identity to every token regardless the data carried by tokens. In the first approach, two tokens are considered to be identical if the data items in the corresponding positions of tuples are same. In the second approach the notion of identity can be based either on the identities of tokens or data items carried by tokens or both. Despite the introduction of the identities for tokens, the identities are not used in the description of the dynamics of IFS-nets.

The formal definition of IFS-nets contains technical ambiguities which must be solved before proceeding to the PrT-net based analysis of IFS-nets. The first of the problems is related to the definition of the functions of sets OP and $CLSFN$ while the second problem is related to the uniqueness of the notion of marking as defined in equation (4).

The intuitive purpose of the function *transop* is to link a function from set

OP to each transition of an IFS-net. When a transition fires, the associated function computes the data item to be carried by the output token. The data items of the incoming tokens are the input of this computation.

Let's suppose that a transition t of an IFS-net has three input places and *transop* links function fd to t. Obviously fd should have three input arguments as well as a general rule which defines how the incoming data items are assigned to the formal arguments of fd. Unfortunately items 5 and 8 of definition 2.1 do not formally relate the number of arguments of the function fd to the number of input places of t nor define how the input data items should be used as arguments of fd.

To resolve the ambiguity related to the functions of the set OP, the set OP is supposed to contain one or more functions of the type $Data^i \rightarrow Data$ for each $i \in \{1, \dots, |P|\}$ and the *transop* is supposed to associate a function $fd : Data^n \rightarrow Data$ with each transition t, where n is the number of input arcs of transition t and $|P|$ denotes the number of places of the net. Furthermore, the set of input arcs of every transition having n input places is supposed to have an index from the set $\{1, \dots, n\}$ and the data item coming along the ith input arc is assigned to ith argument of the function associated with the transition by *transop*. Clearly it is sufficient to include functions up to $|P|$ arguments in OP because any transition of the net may have at most $|P|$ input places and therefore at most $|P|$ incoming arcs.

The purpose of function *transcls* is analogous to purpose of function *transop*. As already mentioned above *transcls* associates a function which calculates the security class of the token to be emitted with each transition of an IFS-net. Because the ambiguity related to the functions of set $CLSFN$ is similar to the ambiguity related to the functions of OP, the solution also resembles the solution given in the context of OP. This means that set $CLSFN$ contains one or more functions of the type $SC^i \rightarrow SC$ for each $i \in \{1, \dots, |P|\}$, *transcls* relates a function which has the same number of arguments as the transition has incoming arcs to each transition, and the assignment of the incoming data items is driven by the indexing of the incoming arcs.

The main problem of the definition of marking is that formula (4) does not define an unique marking for a given distribution of tokens. Consider a place p such that $\bar{M}(p)$ contains exactly three tokens: one tok_1 and two pieces of tok_2. Assume that function *tokdata* assigns a data item d_1 and the function *toksec* assigns a security class C to the token tok_1 while a data item d_2 and a security class S are linked to both the pieces of tok_2. According to formula (4) the marking of place p is

$$\bar{M}(p) = \langle \{tok_1, 2tok_2\}, 3, \{C, S\}, \{d_1, d_2\} \rangle. \tag{11}$$

A closer study of equation (11) shows that it is impossible without further information to say how the data items and the security classes are linked to the tokens at $\bar{M}(p)$. To resolve this ambiguity, the notion of marking is redefined as follows. Let E be a set $E = \mathit{Tokens} \times \mathit{SC} \times \mathit{Data}$ where *Tokens*, *SC* and *Data* are the sets of an IFS-net. Then the marking the IFS-net is

$$\bar{M} : P \rightarrow \mathcal{P}(B_{\mathrm{E}}), \tag{12}$$

i.e, the marking associates a bag over E with each place of the net. For a place p value of the marking is defined as

$$\bar{M}(p) = \bigcup_{tok \in tk(p)} \langle tok, toksec(tok), tokdata(tok) \rangle \tag{13}$$

where the union operator refers to the union of bags. According to this definition the marking given in equation (11) must be revised to form

$$\bar{M}(p) = \{\langle tok_1, C, d_1 \rangle, 2\langle tok_2, S, d_2 \rangle\}. \tag{14}$$

It is quite obvious that the definition of the marking is now unique.

3 Predicate/Transition Nets

Genrich introduced a class of high-level Petri-nets called *Predicate/Transition nets* (or PrT-nets) in his work [7]. A generalized version of the formalism, so called *abstract PrT-nets* were introduced in [8]. The main difference between the abstract and the basic PrT-nets is that an abstract net describes a family of interrelated dynamic systems whereas a basic net describes only one specific system.

The following condensed presentation describes PrT-nets as they were defined in Genrich's original work except that a minor part of the notation is borrowed from [8, 24]. Firstly, the basic form of the PrT-nets is defined. Secondly, a practical extension of the basic formalism, the weak interpretation of the nets, is informally described and its influence to the structure and dynamics of the basic nets is discussed.

3.1 The Basic Form of PrT-nets

Let Ω be a finite set of *operators* such that $\Omega = \bigcup_{n\in\mathbf{N}} \Omega^{(\mathrm{n})}$ and $\Pi = \bigcup_{n\in\mathbf{N}} \Pi^{(\mathrm{n})}$ a finite set of *predicates*, where $\Omega^{(\mathrm{n})}$ is a set of n-ary operators and $\Pi^{(\mathrm{n})}$ is a set of n-ary predicates. Moreover, let V be an infinite set of *variables* such that V and $\Omega \cup \Pi$ are disjoint.

Definition 3.15 Let Ω, Π and V be a set of operators, predicates and variables respectively. The language $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{T} \cup \mathbf{F} \cup \mathbf{S}$ where $\mathbf{T}$ is the set of *individual terms*, $\mathbf{F}$ is the set of *first-order logical formulae* and $\mathbf{S} = \bigcup_{k\in\mathbf{N}} \mathbf{S}^{(\mathrm{k})}$ is the set of *symbolic sums*. The sets $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{S}^{(\mathrm{k})}$ for each $k \in \mathbf{N}$ are defined as follows:

1. Set $\mathbf{T}$:
   (a) Every variable is in $\mathbf{T}$.
   (b) If $f \in \Omega^{(\mathrm{n})}$ and $v_1, \ldots, v_{\mathrm{n}} \in \mathbf{T}$ then $f(v_1, \ldots, v_{\mathrm{n}}) \in \mathbf{T}$, $n \in \mathbf{N}$.
2. Set $\mathbf{F}$:
   (a) If $v_1, v_2 \in \mathbf{T}$ then $v_1 = v_2 \in \mathbf{F}$.
   (b) If $P \in \Pi^{(\mathrm{n})}$ and $v_1, \ldots, v_{\mathrm{n}} \in \mathbf{T}$ then $Pv_1 \ldots v_{\mathrm{n}} \in \mathbf{F}$, $n \in \mathbf{N}$.
   (c) If $p_1, p_2 \in \mathbf{F}$ then $\neg p_1 \in \mathbf{F}$ and $(p_1 \vee p_2) \in \mathbf{F}$.
   (d) If $x \in$ V and $p \in \mathbf{F}$ then $(\exists x)p \in \mathbf{F}$.
3. Set $\mathbf{S}^{(\mathrm{k})}$, $k \geq 1$:

(a) $\mathbf{0} \in \mathbf{S}^{(\mathrm{k})}$.

(b) If $v_1, \ldots, v_{\mathrm{k}} \in \mathbf{T}$ then $\langle v_1, \ldots, v_{\mathrm{k}} \rangle \in \mathbf{S}^{(\mathrm{k})}$.

(c) If $l_1, l_2 \in \mathbf{S}^{(\mathrm{k})}$ then $(l_1 + l_2) \in \mathbf{S}^{(\mathrm{k})}$.

(d) If $l \in \mathbf{S}^{(\mathrm{k})}$ and $z \in \mathbf{Z}^+$ then $zl \in \mathbf{S}^{(\mathrm{k})}$.

There are no other expressions in $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{S}^{(\mathrm{k})}$, $k \geq 1$. □

The 0-ary operators or *constants* introduced by the rule 1b will be used as names of different individuals and the 0-ary predicates denote the propositional variables of the language. Furthermore, the formulae defined by rules 2a and 2b are called *atomic formulae.* The connectives $\wedge$, $\rightarrow$, $\leftrightarrow$ and $\forall$ are derived from $\neg$, $\vee$ and $\exists$ in the standard way. An occurrence of a variable x in an expression $E \in \mathbf{L}$ is a *free occurrence of* x if it is not in the range of a $(\exists x)$ or $(\forall x)$. An expression E is called *open* iff all the occurrences of variables in E are free and it is called *closed* iff no occurrence of its variables is free. Furthermore, the set of variables that have a free occurrence in E is called the *degree of* E. Intuitively, the free occurrences of a variable in an expression hold places for individuals, closed terms are the compound names of individuals and closed formulae are propositions.

Let E be an expression of $\mathbf{L}$, $x_1, \ldots, x_{\mathrm{k}} \in \mathrm{V}$ and $v_1, \ldots, v_{\mathrm{k}} \in \mathbf{T}$. The expression achieved by substituting simultaneously v_{i} for each free occurrence of x_{i}, $1 \leq i \leq k$, is denoted by $E : \{x_1 \leftarrow v_1, \ldots, x_{\mathrm{k}} \leftarrow v_{\mathrm{k}}\}$. The element $\alpha = \{x_1 \leftarrow v_1, \ldots, x_{\mathrm{k}} \leftarrow v_{\mathrm{k}}\}$ is *a substitution* and $E : \alpha$ is *an* α*-instance of* E. It should be noted that $E : \alpha$ is a term, a formula or a symbolic sum iff E is a term, a formula or a symbolic sum respectively.

Definition 3.16 The relational structure $\mathcal{R} = (D; F_1, \ldots, F_{\mathrm{k}}; R_1, \ldots, R_n)$ is a $\mathbf{L}$*-structure* iff every m-ary function of $\mathcal{R}$ is related to an operator $f \in \Omega^{(\mathrm{m})}$, and every m-ary relation of $\mathcal{R}$ is related to a predicate $P \in \Pi^{(\mathrm{m})}$. □

Let $\mathcal{R}$ be a $\mathbf{L}$-structure and $\mathbf{L}_{\mathcal{R}}$ a language $\mathbf{L}_{\mathcal{R}} = \mathbf{L} \cup U_{\mathcal{R}}$ such that the items of $U_{\mathcal{R}}$ and the 0-ary operators of $\mathbf{L}$ correspond the items of the domain of $\mathcal{R}$ in one-to-one fashion. The item of D associated with a constant u is denoted by $u_{\mathcal{R}}$. Consequently, the function of $\mathcal{R}$ associated with an operator f is denoted by $f_{\mathcal{R}}$ and the relation of $\mathcal{R}$ associated with a predicate P is denoted by $P_{\mathcal{R}}$. This means that $f_{\mathcal{R}}$ is a function $f_{\mathcal{R}} : D^m \rightarrow D$ and $P_{\mathcal{R}}$ is a subset $P_{\mathcal{R}} \subseteq D^m$. In order to inspect the properties of a $\mathbf{L}$-structure with language $\mathbf{L}$, a more detailed interpretation of the various constructs of $\mathbf{L}$ in respect of the given relational structure is needed. For this reason *the value of an expression* is defined for every expression of $\mathbf{L}_{\mathcal{R}}$: an element of the domain is linked to every closed term, a truth-value ($\top$ or $\perp$) is related to every closed formula, and a multi-set over the domain is linked to every symbolic sum.

Before the formal definition of the value of an expression can be stated, the notion of multi-set must be made precise.

Let D be a set. *A linear combination in D with integer coefficients* is a function $\lambda : D \to \mathbf{Z}$. The set of all linear combinations in D is denoted by $\mathcal{L}(D)$ and consequently the set of all linear combinations of any cartesian product $D_1 \times \ldots \times D_n$ is denoted by $\mathcal{L}(D_1 \times \ldots \times D_n)$. *A multi set (or a bag) over D* is a non-negative linear combination in D. Furthermore, the set of all multi sets over D is denoted by $\mathcal{L}^+(D)$. As an example of the notation to be used, the linear combination λ over set $D = \{a, b, c, d\}$ such that $\lambda(a) = 2$, $\lambda(b) = -3$, $\lambda(c) = 0$ and $\lambda(d) = 1$ will be written as $2\langle a\rangle - 3\langle b\rangle + \langle d\rangle$. Consequently, $\mathbf{0}$ and $\mathbf{1}$ denote the symbolic sums of $\mathcal{L}(D)$ whose coefficients are all 0 or all 1 respectively. The componentwise multiplication, multiplication by scalars and comparison are defined in obvious way. Let $\lambda, \lambda_1, \lambda_2 \in \mathcal{L}(D)$ and $z \in \mathbf{Z}$. Then for each $x \in D$

$$
\begin{aligned}
(\lambda_1 + \lambda_2)(x) &= \lambda_1(x) + \lambda_2(x), && (15)\\
(\lambda_1 \cdot \lambda_2)(x)) &= \lambda_1(x) \cdot \lambda_2(x), && (16)\\
(z\lambda)(x) &= z\lambda(x), && (17)\\
\lambda_1 \leq \lambda_2 &\Leftrightarrow \forall y \in D : \lambda_1(y) = \lambda_2(y). && (18)
\end{aligned}
$$

From the set-theoretic point of view, the subsets of D are represented by the linear combinations whose coefficients are either 0 or 1, the element $\mathbf{0} \in \mathcal{L}^+(\mathrm{D})$ corresponds to the empty set and the element $\mathbf{1} \in \mathcal{L}^+(\mathrm{D})$ corresponds to the set D. Moreover, the componentwise multiplication of these linear combinations conforms to the intersection of sets.

Definition 3.17 Let $\mathcal{R}$ be a relational structure and $E \in \mathbf{L}_{\mathcal{R}}$ a closed expression. *The value of E in respect of $\mathcal{R}$* denoted by $\mathcal{R}(E)$ is defined as follows:

1. (a) $\mathcal{R}(u) = u_{\mathcal{R}}$, where u is a constant.
   (b) $\mathcal{R}(f(v_1, \ldots, v_\mathrm{n})) = f_{\mathcal{R}}(\mathcal{R}(v_1), \ldots, \mathcal{R}(v_\mathrm{n}))$, where $f(v_1, \ldots, v_\mathrm{n}) \in \Omega$.
2. (a) $\mathcal{R}(v_1 = v_2) = \top$ iff $\mathcal{R}(v_1) = \mathcal{R}(v_2)$.
   (b) $\mathcal{R}(Pv_1 \ldots v_\mathrm{n}) = \top$ iff $\langle \mathcal{R}(v_1), \ldots, \mathcal{R}(v_\mathrm{n})\rangle \in P_{\mathcal{R}}$.
   (c) $\mathcal{R}(\neg p) = \top$ iff $\mathcal{R}(p) = \bot$, where p is a formula.
   (d) $\mathcal{R}((p_1 \vee p_2)) = \top$ iff $\mathcal{R}(p_1) = \top$ or $\mathcal{R}(p_2) = \top$.
   (e) $\mathcal{R}((\exists x)p) = \top$ iff there exists a constant u: $\mathcal{R}(p : \{x \leftarrow u\}) = \top$.
3. (a) $\mathcal{R}(\mathbf{0}) = \mathbf{0}$.

(b) $\mathcal{R}(\langle v_1, \ldots, v_n \rangle) \in \mathcal{L}^+(D^n)$ and $\mathcal{R}(\langle v_1, \ldots, v_n \rangle)(u_1, \ldots, u_n) = 1$ iff $\mathcal{R}(v_i) = u_i$, $1 \leq i \leq n$ and $\mathcal{R}(l)(u_1, \ldots, u_n) = 0$ otherwise.

(c) $\mathcal{R}(l_1 + l_2) = \mathcal{R}(l_1) + \mathcal{R}(l_2)$.

(d) $\mathcal{R}(zl) = z\mathcal{R}(l)$, $z \in \mathbf{Z}^+$.

□

Let $\mathbf{A}$, $\mathbf{B}$ be finite sets of formulae and p_1,p_2 formulae of $\mathbf{L}$. Furthermore, let l_1,l_2 denote bounded symbolic sums of $\mathbf{L}$, $\mathcal{R}$ a $\mathbf{L}$-structure and p_3 a formula of $\mathbf{L}_{\mathcal{R}}$. Then the formula p_3 is *valid in* $\mathcal{R}$, $\mathcal{R} \models p_3$, iff the value of each closed instance of p_3 with respect $\mathcal{R}$ is $\top$, and the formula p_1 is *valid*, $\models p_1$, iff p_1 is valid in every $\mathbf{L}$-structure. The relational structure $\mathcal{R}$ *is a model of* $\mathbf{A}$, $\mathcal{R} \models \mathbf{A}$, iff every formula of $\mathbf{A}$ is valid in $\mathcal{R}$. The formula p_1 is *a logical consequence of* $\mathbf{A}$, $\mathbf{A} \models p_1$, iff p_1 is valid in every model of $\mathbf{A}$. The sets $\mathbf{A}$ and $\mathbf{B}$ are *logically equivalent* iff for all relational structures $\mathcal{R}$: $\mathcal{R} \models \mathbf{A}$ iff $\mathcal{R} \models \mathbf{B}$. The symbolic sums l_1 and l_2 are *equivalent*, $l_1 = l_2$, iff for all finite structures $\mathcal{R}$ and for all substitutions α: $\mathcal{R}(l_1 : \alpha) = \mathcal{R}(l_2 : \alpha)$. Finally, the formula p_2 *is implied by* p_1, $p_1 \Rightarrow p_2$, iff $(p_1 \rightarrow p_2)$ is valid.

As briefly mentioned above, the aim of PrT-nets is to aid the inspection of dynamic relational structures. *A dynamic relational structure* is a relational structure in which the extensions of specific relations may vary from time to time. Let Π_{dy} be the set of *dynamic predicates of* $\mathbf{L}_{\mathcal{R}}$, i.e., the set of predicates which are bound to dynamic relations of $\mathcal{R}$ and let Π_{st} denote the set of *static predicates of* $\mathbf{L}_{\mathcal{R}}$ respectively. Additionally, let $\Pi = \Pi_{\mathrm{st}} \cup \Pi_{\mathrm{dy}}$ and $\Pi_{\mathrm{st}} \cap \Pi_{\mathrm{dy}} = \emptyset$ hold. Let $\mathbf{L}_{\mathrm{st}} \subseteq \mathbf{L}$ denote a language including all expressions of $\mathbf{L}$ except those which contain one or more dynamic predicates of $\mathbf{L}$. Furthermore, let $\mathbf{F}_{st}$ and $\mathbf{S}$ denote the formulae and the symbolic sums of $\mathbf{L}_{\mathrm{st}}$.

Definition 3.18 *A PrT-net* is a triple $\mathcal{N} = (N, A, M_0)$ where

1. $N = (S, T; F)$ is *a directed pre-net* where S and T are finite sets such that $S \cap T = \emptyset$ and $F \subseteq (S \times T) \cup (T \times S)$.

2. $A = (A_N, A_S, A_T, A_F)$ is *the annotation of* N where

   (a) $A_N = \mathcal{R}$ is a relational structure for $\mathbf{L}_{\mathrm{st}}$ called the *support of* $\mathcal{N}$,

   (b) $A_S : S \rightarrow \Pi_{\mathrm{dy}}$ is a bijection,

   (c) $A_T : T \rightarrow \mathbf{F}_{st}$ relates an open formula called *the selector of* t to each transition $t \in T$,

   (d) $A_F : F \rightarrow \mathbf{S}$ relates a symbolic sum to each arc $(x, y) \in F$ such that if the place s of the pair (x, y) is linked with a m-ary predicate of $\Pi_{\mathrm{dy}}^{(\mathrm{m})}$ then $A_F(s) \in \mathbf{S}^{(\mathrm{m})}$.

3. $M_0 : S \to \mathbf{S}$ relates a symbolic sum l called the *initial marking of s* to each place s such that every term of l is a constant, $l \in \mathbf{S}^{(\mathrm{m})}$, where m is the arity of the predicate $A_S(s)$ and $\mathbf{0} \leq \mathcal{R}(M_0(s)) \leq \mathbf{1}$.

The smallest set containing all PrT-nets is denoted by $\mathcal{B}(\mathbf{L})$. □

Let t be a transition of a PrT-net $\mathcal{N}$. The *degree of the transition* t is the union of degrees of expressions $A_F(a)$, where each arc a is adjacent to t. It should be noted that the degree of the annotation of t, i.e., the degree of the expression $A_T(t)$ is not necessarily a subset of the degree of t. A free variable of $A_T(t)$ is *a dangling variable* if it does not belong to the degree of t. The dangling variables can be bound with existential quantifiers without changing the semantics of the annotation.

Definition 3.19 Let $\mathcal{N}$ be a PrT-net, t its transition, p the formula $A_T(t)$ and $\alpha : \mathrm{V} \to U_{\mathcal{R}}$ a substitution which replaces all variables in the degree of t by constants. Let $X_i = \{l \mid l = A_T(s,t), (s,t) \in F\}$ and $X_o = \{l \mid l = A_T(t,s), (t,s) \in F\}$. The substitution α is *feasible for t* iff

1. There exists a substitution α_1 of dangling variables of p by constants such that $\mathcal{R}((p : \alpha) : \alpha_1) = \top$,
2. For each $l \in X_i \cup X_o : \mathbf{0} \leq \mathcal{R}(l : \alpha) \leq \mathbf{1}$,
3. For each $(l_1, l_2) \in X_i \times X_o : \mathcal{R}(l_1 : \alpha) \cdot \mathcal{R}(l_2 : \alpha) = \mathbf{0}$.

□

The first claim of definition 3.19 assures that the selector of the transition is satisfied. The intuition behind the second requisite is that the substitution relates a set instead of a multi set to each arc. Finally, the third requirement prevents the same set to be linked to both an incoming and outgoing arc of the transition.

Definition 3.20 Let $\mathcal{N}$ be a PrT-net and $A_N = \mathcal{R}$ its support. Let M_1 and M_2 be markings of $\mathcal{N}$, t a transition and α a substitution for t which replaces the variables in the degree of t by constants. The relation *α-occurrence of t at M_1 leading to M_2* is said to hold between M_1 and M_2, denoted by $M_1[t : \alpha\rangle_{\mathcal{R}} M_2$, iff the following conditions are fulfilled

1. α is a feasible substitution for t,
2. For all $s \in S, (s,t) \in F$: $\mathcal{R}(A_F(s,t) : \alpha) \leq \mathcal{R}(M_1(s))$,
3. For all $s \in S, (t,s) \in F$: $\mathcal{R}(A_F(t,s) : \alpha) \cdot \mathcal{R}(M_1(s)) = 0$,

4. For all $s \in S$: $\mathcal{R}(M_2(s)) = (\mathcal{R}(M_1(s)) - \mathcal{R}(A_F(s,t):\alpha) + \mathcal{R}(A_F(t,s):\alpha))$.

A transition t *is enabled at* M iff there exists a substitution α and marking M_1 such that $M[t:\alpha\rangle_{\mathcal{R}} M_1$. □

The transition rule of definition 3.20 makes it sensible to talk about reachable markings of a PrT-net. Let $\mathcal{N}$ be a PrT-net and M_0 its initial marking. *The set of reachable markings of* $\mathcal{N}$, denoted by $[M_1\rangle_{\mathcal{R}}$, is the smallest set conforming with the following requirements: $M_0 \in [M_0\rangle_{\mathcal{R}}$. If $M_0 \in [M_0\rangle_{\mathcal{R}}$ and there exists a marking M_2, a transition t and a substitution α such that $M_1[t:\alpha\rangle_{\mathcal{R}} M_2$, then $M_2 \in [M_0\rangle_{\mathcal{R}}$. A transition t is *dead* iff for all $M \in [M_0\rangle_{\mathcal{R}}$, t is not enabled at M.

A reachability graph is a way to describe the dynamics of a PrT-net. *The reachability graph of a PrT-net* is an edge labeled directed multigraph[2] whose vertices are members of the set of reachable markings and the edges are labelled with transition instances. Because the set of the reachable markings is not necessarily finite, the approach where the set of graph vertices is equal with the set of reachable markings leads to the definition which allows infinite reachability graphs in some cases. This is very undesirable situation from practical point of view.

A finite characterization for an infinite set of reachable markings can be constructed by utilizing the Karp-Miller trees and coverability graphs [31]. Although the usage and limitations of covering graphs are well documented in the context of low level nets [30, 28] and there exist some preliminary results on the covering graphs of certain high level nets (so called HL-nets) [16], there does not seem to exist a detailed study on the covering graphs of PrT-nets. Moreover, the relation between IFS-nets and PrT-nets can be formally established without the notion of infinite reachability graphs. Therefore the following definition of the reachability graph of PrT-nets is used for the rest of this work.

Definition 3.21 Let $\mathcal{N}$ be a PrT-net. If the set of reachable markings $[M_0\rangle_{\mathcal{R}}$ is finite then the *the reachability graph for* $\mathcal{N}$ is a directed labelled multigraph $\mathcal{G} = (V, E)$ such that

1. $V = [M_0\rangle_{\mathcal{R}}$.
2. If $M_1, M_2 \in V$ and there exists a transition t and a substitution α such that $M_1[t:\alpha\rangle_{\mathcal{R}} M_2$ then

[2] Recall that a directed multigraph is a directed graph which may have more than one edge between a pair of vertices. The edges between a pair of vertices are supposed to be indexed so that they can be distinguished. More information about the definition and applications of multigraphs can be found, e.g., from [11].

f	a	b	c	d
a	a	b	c	d
b	a	c	c	d
c	a	b	d	d
d	d	d	d	d

Table 1: Operator **f** of the example PrT-net

(a) $(M_1, M_2) \in E$ and

(b) the edge (M_1, M_2) is labelled with $t : \alpha$.

If the set of reachable markings of a PrT-net is not finite then the reachability graph is not defined. A node of the reachability graph is a *terminal node* iff there does not exist an edge which leaves the node. □

Some authors prefer the term *deadlock* to terminal node promoted here. This term leads to the interpretation where a marking without follower markings is always considered to describe an undesirable state of the system. There exist, however, many examples where a marking without follower markings cannot be interpreted as an evidence of error. Therefore a deadlock is defined to be an undesirable terminal node in this work.

The following example clarifies the previously given definitions and introduces the graphical notation to be used when PrT-nets are drawn.

Example 3.22 Consider a language **L** which is defined over the set of operators $\Omega = \{f^{(2)}\} \cup \Omega^{(0)}$ and the set of predicates $\Pi = \Pi_{\mathrm{st}} \cup \Pi_{\mathrm{dy}} = \emptyset \cup \{P_1{}^{(1)}, P_2{}^{(2)}\}$. Moreover, let's assume that set V contains distinguished variables $x, y, z \in \mathrm{V}$. Let $\mathcal{N} = (N, A, M_0)$ be a PrT-net where the related pre-net is $N = (S, T; F)$, $S = \{s_1, s_2\}$, $T = \{t_1, t_2, t_3, t_4\}$, $F = \{(s_1, t_1), (s_1, t_2), (t_1, s_2), (t_2, s_2), (s_2, t_3), (s_2, t_4), (t_3, s_1)\}$. The annotation of the net is $A = (A_N, A_S, A_T, A_F)$ where $A_N = \mathcal{R} = (\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}; \mathbf{f}; \emptyset)$ is a **L**-structure for $\mathbf{L}_{\mathrm{st}}$ such that $a, b, c, d \in \Omega^{(0)}$ correspond to the elements **a**, **b**, **c** and **d** of the **L**-structure at hand and the operator symbol $f^{(2)}$ corresponds to the function **f** which is defined in table 1. Furthermore, A_S links a dynamic predicate $P_1{}^{(1)}$ to place s_1 and a predicate $P_2{}^{(2)}$ to place s_2, A_T links the formula $x = a \vee x = b$ to transition t_1, formula $x = a \vee x = c$ with t_2, formula $\top$ with transition t_3 and formula $x = d$ to transition t_4, and A_F relates a symbolic sum to each arc as follows: $\langle x \rangle$ to (s_1, t_1), $\langle x \rangle$ to (s_1, t_2), $\langle x, b \rangle$ to (t_1, s_2), $\langle x, c \rangle$ to (s_1, t_1), $\langle y, z \rangle$ to (s_2, t_3), $\langle y, z \rangle$ to (s_2, t_4) and $\langle \mathbf{f}(y, z) \rangle$ to (t_3, s_1). Finally, function M_0 associates markings to the places as follows: $\langle a \rangle$ with s_1 and $\mathbf{0}$ with s_2. The graphical presentation of the net is given in figure 2 and the reachability graph of the net is given in figure 3. The selector formula of the transition

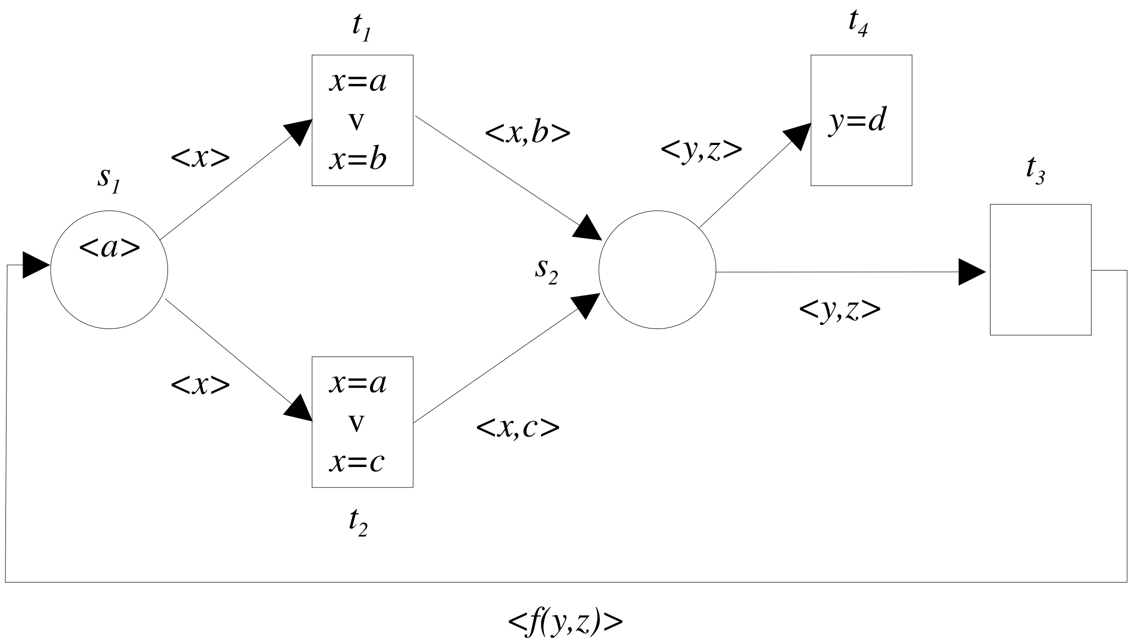

Figure 2: A sample PrT-net

t_3 and the initial marking of the place s_2 have been intentionally left out to shorten the notation. The set of reachable markings contains eight markings. Marking M_7 is a terminal node because there does not exist an arc which leaves M_7. Transition t_4 is dead because it is not enabled at any marking reachable from M_0. □

3.2 The Weak Interpretation of PrT-nets

The second requirement of definition 3.19 and the third item of definition 3.20 and the imply that a marking or a symbolic sum annotating an arc of a PrT-net may contain only one copy of any tuple of terms. From practical point of view this is not a desirable property. It is not, however, necessary to revise *the strict interpretation of PrT-nets* introduced in the previous definitions if the arcs of the net are to be annotated with multiple copies of symbolic sums. A sketch of a simulation of *the weak interpretation of PrT-nets* based on previously given definitions is described in the following paragraphs. For a detailed treatment of the issue, see [7].

Consider a marking M of a PrT-net where $M(s) = \langle a \rangle$ for place s. Any other token (of arity one) but $\langle a \rangle$ may well be added to place s. If the arity of the dynamic predicate annotating s is increased by one, a marking, say, $M(s) = \langle a, 0\rangle + \langle a, 1\rangle$ can be legally assigned to s. Now, tokens $\langle a, 0\rangle$ and $\langle a, 1\rangle$ may be read to represent different copies of original token $\langle a \rangle$. Thus, by selecting a finite set of tags, e.g., a subset of $\mathbf{N}$ the problem of multiple

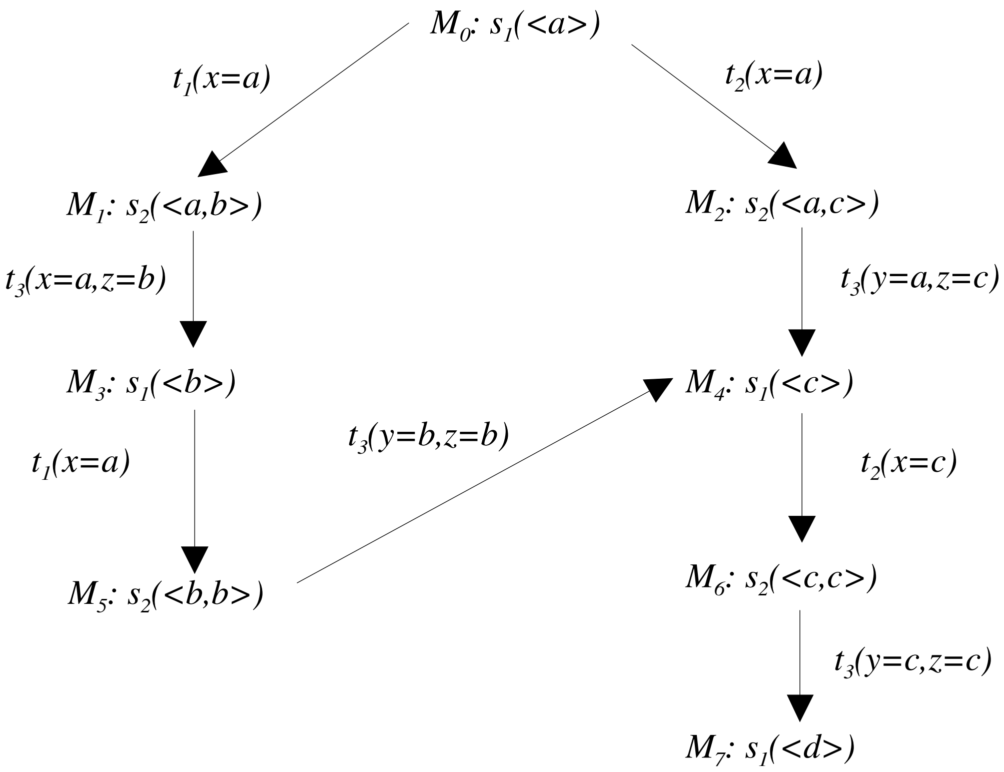

Figure 3: The reachability graph of the sample PrT-net

copies of tuples in places can be solved.

The tuples annotating the arcs of the net are also expanded with extra position. These positions reserve places for the tags. Every position is occupied by a distinct variable which does not occur in the annotation of the adjacent arc. The effect of this arrangement is that the tags do not have any kind of effect to the occurrences of the transition. In order to shorten the notation, let the *don't care symbol* stand for any variable reserving place for a tag. As a result of appending tags arc annotations like $\langle x\rangle + \langle x\rangle = 2\langle x\rangle$ can be translated to $\langle x, \sim\rangle + \langle x, \sim\rangle = 2\langle x, \sim\rangle$. Any substitution which assigns different tags to two different variables represented by the two don't care symbols is now feasible. For instance, a substitution which turns the arc annotation $\langle x, \sim\rangle + \langle x, \sim\rangle$ to $\langle x, 1\rangle + \langle x, 2\rangle$ is now perfectly feasible.

Weakly interpreted PrT-nets will be frequently seen in the subsequent sections. These nets should always be considered as shortening of strictly interpreted nets with tags and extra positions in tuples as sketched above.

4 The Mapping of IFS-nets to PrT-nets

The aim of this chapter is to show how the security properties of an IFS-net can be inspected by creating a special PrT-net and by studying the properties of the reachability graph of the constructed net. The cruxes of this approach are the mapping which defines how an IFS-net induces a PrT-net and the theorems which bind the security properties of the original IFS-net to the properties of the reachability graph of the PrT-net.

Consider an IFS-net IN. A PrT-net $\mathcal{N}_{IN}$ corresponds to IN if $\mathcal{N}_{IN}$ is an image of IN under the mapping to be described in subsection 4.1. The Petri net structure of IN and the pre-net of $\mathcal{N}_{IN}$ are, if viewed as bipartite graphs, isomorphic. Every token of $\mathcal{N}_{IN}$ has the arity of four; every tuple of every arc annotation and every place must therefore also have the arity of four. In addition to a data item and a security class each token carries information about the security of the place where the token resides and the security of the transition firing which produced the token. The first position of every token is reserved for a term whose value in respect of every possible Petri net structure is a member of the set *Data*. The second position carries a term which represents the security class associated with the token. The third position of each token reserves space for a special symbol which is needed when the security of a place containing the token is considered. The last position of each token contains a symbol which tells whether the token was produced by a secure firing of a transition. The initial marking and the annotations of the transitions and arcs are defined in a way which leads to the one-to-one correspondence between the sets of reachable markings of IN and $\mathcal{N}_{IN}$.

The subsequent presentation is divided into four subsections. Subsection 4.1 defines the mapping of IFS-nets to PrT-nets, i.e, how the PrT-net corresponding to a given IFS-net should be constructed. Subsection 4.2 contains results about the reachability relations of an IFS-net and the corresponding PrT-net. Subsection 4.3 is dedicated to theorems which show how the various security properties of an IFS-net can be detected from the reachability graph of the corresponding PrT-net. Finally, subsection 4.4 discusses the direct construction of PrT-nets.

4.1 The Definition of the Mapping

The support of a PrT-net is a relational structure containing all those entities which are relevant to the models of specific application domain. The relational structure corresponding to an IFS-net contains all those elements which are needed when a PrT-net corresponding to the IFS-net is constructed.

Definition 4.23 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS net. *The relational structure corresponding to IN* is a structure $\mathcal{R}_{IN} = (D; F_1, \ldots, F_k; R_1, \ldots, R_n)$ such that $D = \mathit{Tokens} \cup \mathit{Data} \cup \mathit{SC} \cup \mathit{OP} \cup \mathit{CLSFN} \cup \mathit{TF} \cup T \cup \{\mathbf{s}, \mathbf{i}\}$, $\bigcup_{i=1}^{k} F_i = \mathit{transop} \cup \mathit{transcls} \cup \mathit{psec} \cup \mathit{psecs} \cup \mathit{tsecs} \cup \mathit{tk}_0 \cup \mathit{tokdata}_0 \cup \mathit{toksec}_0$ where

1. $\mathit{psecs} : P \times SC \rightarrow \{\mathbf{s}, \mathbf{i}\}$, $\forall p \in P\, \forall c \in SC : \mathit{psecs}(p, c) = \mathbf{s}$ if $c \in \mathit{psec}(p)$ and $\mathbf{i}$ otherwise,

2. $\mathit{tsecs} : \bigcup_{i=1}^{|P|} SC^i \times SC \rightarrow \{\mathbf{s}, \mathbf{i}\}, \forall c_1 \in \cup_{i=1}^{|P|} SC^i\, \forall c_2 \in SC : \mathit{tsecs}(c_1, c_2) = \mathbf{s}$ if $\mathit{lub}(c_1) \leq c_2$ and $\mathbf{i}$ otherwise,

3. the extensions of the functions tk_0, $\mathit{tokdata}_0$ and toksec_0 are equal to the extensions of the functions tk, $\mathit{tokdata}$ and toksec respectively at the marking $\bar{M}_0$,

4. $\{\leq\} \in \bigcup_{j=1}^{n} R_j$ where $\leq$ is the partial order of the set of security classes.

□

The set *Tokens* is included in the domain of the relational structure because it would otherwise be impossible to define the initial making of the constructed PrT-net. The functions tk_0, $\mathit{tokdata}_0$ and toksec_0 are only needed when the initial marking of the constructed PrT-net is defined. The transitions of the IFS-net are included in the set D because some functions of $\mathcal{R}_{IN}$ require them as arguments. The functions *psecs* and *tsecs* are used in output arc annotations. The value of *psecs* indicates whether the given security class conform to the security rating of a given place. The purpose of the function *tsecs* is to check the observance of the predefined security policy. It indicates whether it is correct to assign the security class given as the second argument to the result of an atomic computation when the security classes given as the first argument are linked to the inputs of the computation. The set of relations of $\mathcal{R}_{IN}$ is also supposed to contain all the (static) relations needed to describe the particular system under the inspection.

The concept of the corresponding PrT-net, which is formally described in the following definition, defines indirectly the mapping of IFS-nets to PrT-nets. A PrT-net corresponding to a given IFS-net is an image of the IFS-net under the mapping.

Definition 4.24 Let $IN = \langle IC, \bar{M}_0 \rangle$ be an IFS-net. *A PrT-net corresponding IN* is a triple $\mathcal{N} = (N_{\mathcal{N}}, A_{\mathcal{N}}; M_0)$ where

1. $N_{\mathcal{N}} = (S_{\mathcal{N}}, T_{\mathcal{N}}; F_{\mathcal{N}})$ is a directed pre-net such that

   (a) There exists a bijective mapping $V_P : P_{IC} \rightarrow S_{\mathcal{N}}$.

(b) There exists a bijective mapping $V_T : T_{IC} \to T_{\mathcal{N}}$.

(c) For all $t \in T_{IC}$, $p \in P_{IC}$:

i. $p \in I(t)$ iff $(V_P(p), V_T(t)) \in F_{\mathcal{N}}$,

ii. $p \in O(t)$ iff $(V_T(t), V_P(p)) \in F_{\mathcal{N}}$.

2. $A = (A_N, A_S, A_T, A_F)$ is the annotation of N such that

(a) $A_N = \mathcal{R}_{IN}$ is the support of $\mathcal{N}$ where $\mathcal{R}_{IN}$ is the relational structure corresponding to IN.

(b) $A_S : S_{\mathcal{N}} \to \Pi_{\text{dy}}$ is a bijection where $\Pi_{\text{dy}} \subseteq \Pi^{(4)}$.

(c) $A_T : T_{\mathcal{N}} \to \mathbf{F}_{st}$ relates to each transition the selector $\top$.

(d) $A_F : F_{\mathcal{N}} \to \mathbf{S}$ relates a quadruple of $\mathbf{S}$ to each arc as follows.

i. For all $(s,t) \in F_{\mathcal{N}}, s \in S_{\mathcal{N}}, t \in T_{\mathcal{N}}$: $A_F(s,t) = \langle x, y, z, w \rangle$ where x, y, z and w are variables of $\mathbf{L}$.

ii. For all $(t,s) \in F_{\mathcal{N}}, s \in S_{\mathcal{N}}, t \in T_{\mathcal{N}}$: $A_F(t,s) = \langle d, c, u, v \rangle$ where

A. $d = transop({V_T}^{-1}(t))(x_1, \ldots, x_k)$,

B. $c = transcls({V_T}^{-1}(t))(y_1, \ldots, y_k)$,

C. $u = psecs({V_P}^{-1}(s), transcls({V_T}^{-1}(t))(y_1, \ldots, y_k))$,

D. $v = tsecs(\langle y_1, \ldots, y_k \rangle, transcls({V_T}^{-1}(t))(y_1, \ldots, y_k))$,

k is the number of arcs entering to ${V_T}^{-1}(t) \in T_{IC}$ and $\langle x_i, y_i, z_i, w_i \rangle$ is the annotation of the ith incoming arc of ${V_T}^{-1}(t) \in T_{IC}$.

3. $\bar{M}_0 : S_{\mathcal{N}} \to \mathbf{S}$ relates to each place s a symbolic sum of constants such that

$$\sum_{tok \in tk_0(p)} \langle tokdata_0(tok), toksec_0(tok), psecs(p,u), \mathbf{s} \rangle$$

where $p = {V_P}^{-1}(s)$.

□

Recall that no syntactic difference is made between the related elements which occur in a PrT-net and in its support. The difference do exist because, e.g., the entity *transcls* in definition 4.23 refers to the function of $\mathcal{R}_{IN}$ while the entity *transcls* in definition 4.24 refer to the term of the language $\mathbf{L}_{\mathcal{R}_{IN}}$.

Example 4.25 In this example the PrT-net $\mathcal{N}$ corresponding to the IFS-net of example 2.14 is presented. Let $N_{\mathcal{N}} = (S_{\mathcal{N}}; T_{\mathcal{N}}, F_{\mathcal{N}})$ be a pre-net of $\mathcal{N}$ such that $S_{\mathcal{N}} = \{s_1, s_2, s_3\}$, $T_{\mathcal{N}} = \{t'_1, t'_2, t'_3\}$ and $F_{\mathcal{N}} = \{(s_1, t'_1), (s_1, t'_2), (t'_1, s_2), (t'_2, s_2), (s_2, t'_3), (t'_3, s_3)\}$. Let bijections V_P and V_T be fixed in the following way: $V_P = \{(p_1, s_1), (p_2, s_2), (p_3, s_3)\}$, $V_T = \{(t_1, t'_1), (t_2, t'_2), (t_3, t'_3)\}$. The pre-net clearly fulfills the requirements of item 1 in definition 4.24.

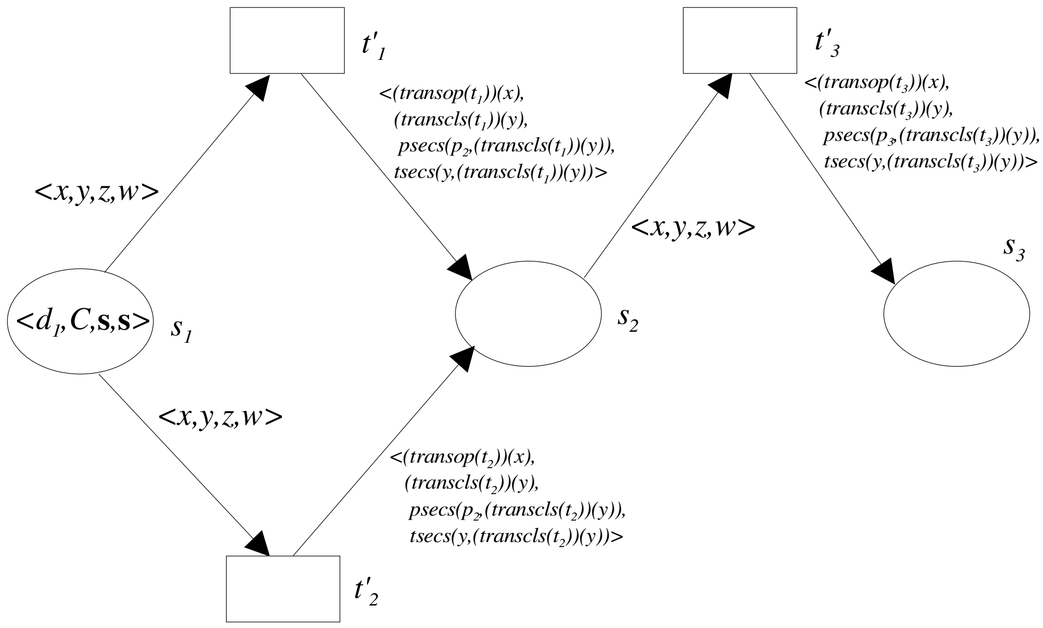

Figure 4: The PrT-net of example 4.25

Let $A_N = \mathcal{R}_{IN}$ be the support of $\mathcal{N}$ such that the domain of $\mathcal{R}_{IN}$ is the set $D = \{d_1, \ldots, d_4\} \cup \{U, C, S\} \cup \{tok_1, tok_2\} \cup \{fd_1, fd_2, fd_3\} \cup \{fs_1, fs_2, fs_3\} \cup \{tf_{t_1}, tf_{t_2}, tf_{t_3}\} \cup \{t_1, t_2, t_3\} \cup \{\mathbf{s}, \mathbf{i}\}$, the set of operations is $\bigcup_{i=1}^{10} F_{\mathrm{i}} = \mathit{transop} \cup \mathit{transcls} \cup \mathit{psec} \cup \mathit{psecs} \cup \mathit{tsecs} \cup \mathit{tk}_0 \cup \mathit{tokdata}_0 \cup \mathit{toksec}_0$ where the meaning of the functions tk_0, $\mathit{tokdata}_0$ and toksec_0 is given by definition 4.23, all other functions are defined as in example 2.14, and the set of relations of $\mathcal{R}_{IN}$ contains only the relation $\leq$. Let the function A_S link the places to predicates as follows: s_1 to P_1, s_2 to P_2 and s_3 to P_3, where $P_1, P_2, P_3 \in \Pi^{(4)}$. The function A_T associates selector $\top$ to transitions t_1', t_2' and t_3'. Let A_F relate quadruples of variables to the input arcs of the transitions as follows: $\langle x, y, z, w\rangle$ to (s_1, t_1'), (s_1, t_2') and (s_2, t_3'). Furthermore, the output arcs of transitions are annotated as follows:

$$
\begin{array}{lll}
(t_1', s_2) & \text{with} & \langle (\mathit{transop}(t_1))(x), (\mathit{transcls}(t_1))(y), \mathit{psecs}(s_2, (\mathit{transcls}(t_1))(y)), \\
 & & \mathit{tsecs}(y, (\mathit{transcls}(t_1))(y))\rangle, \\
(t_2', s_2) & \text{with} & \langle (\mathit{transop}(t_2))(x), (\mathit{transcls}(t_2))(y), \mathit{psecs}(s_2, (\mathit{transcls}(t_2))(y)), \\
 & & \mathit{tsecs}(y, (\mathit{transcls}(t_2))(y))\rangle, \\
(t_3', s_3) & \text{with} & \langle (\mathit{transop}(t_3))(x), (\mathit{transcls}(t_3))(y), \mathit{psecs}(s_3, (\mathit{transcls}(t_3))(y)), \\
 & & \mathit{tsecs}(y, (\mathit{transcls}(t_3))(y))\rangle.
\end{array}
$$

A careful study of definition 4.24 reveals that the given annotation satisfies the requirements of items 2a-2d.

Finally, let the function M_0 associate the following symbolic sums with the places: $\mathbf{0}$ with s_2 and s_3, and $\langle d_1, C, \mathbf{s}, \mathbf{s}\rangle$ with s_1. It should not be a surprise that M_0 conforms to item 3 of definition 4.24 when net IN is considered. The graphical representation of the constructed PrT-net is given in figure 4. □

4.2 The Correspondence of the Reachability Relations

As pointed out in subsection 2.1 the notion of security of IFS-nets is essentially defined in terms of reachable markings and reachability relations. In order to utilize the concept of corresponding PrT-net in the verification of security properties, the connection of the reachability sets of an IFS-net and the corresponding PrT-net as well as the connection of the reachability relations of these nets must be well understood. In order to prove results on this issue, we must first define when a marking of an IFS-net and a marking of the corresponding PrT-net represent the same state of the modelled system.

Definition 4.26 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net and $\bar{M}$ a marking of IN. *The marking of $\mathcal{N}_{IN}$ corresponding to $\bar{M}$* is a marking M such that for all $s \in S$: $\langle d, c, u, v\rangle \in s$ iff $tok \in {V_P}^{-1}(s)$ and $d = tokdata(tok)$ at $\bar{M}$ and $c = toksec(tok)$ at $\bar{M}$ and $u = psecs({V_P}^{-1}(s), toksec(tok))$ and $v \in \{\mathbf{s}, \mathbf{i}\}$. □

For an arbitrary marking of an IFS-net there exist more than one corresponding PrT-net markings. This is due to the fact that for every token tok occuring in the IFS-net there exists two possible tokens which may correspond tok in the PrT-net, namely $\langle d, c, u, \mathbf{s}\rangle$ and $\langle d, c, u, \mathbf{i}\rangle$, where $d \in Data$, $c \in SC$ and $u \in \{\mathbf{s}, \mathbf{i}\}$. Later in this subsection it will be shown (Theorem 4.30) that if IN is an IFS-net, $\bar{M} \in RS_{IN}$ and $\mathcal{N}_{IN}$ is the PrT-net corresponding to IN then there exists an unique marking $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ which corresponds to $\bar{M}$.

Another important observation is that the correspondence of markings implies the correspondence of tokens. Let IN be an IFS-net and $\mathcal{N}_{IN}$ the corresponding PrT-net, $\bar{M}$ and M corresponding markings of IN and $\mathcal{N}_{IN}$ respectively, and let p be a place of IN. It follows directly from definition 4.26 that p contains a token tok at $\bar{M}$ if and only if the place $V_P(s)$ contains a token $\langle d, c, u, v\rangle$ at M where d is the data item and c is the security class associated with the token tok at marking $\bar{M}$.

The following theorem tells how an enabled transition of an IFS-net is reflected in the corresponding PrT-net.

Theorem 4.27 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net, t a transition of IN, $\cup_{i=1}^{n} p_i = I(t)$ the set of input places of t, and $\bar{M}$, M markings of IN and $\mathcal{N}_{IN}$ such that M corresponds to $\bar{M}$. The transition t is enabled at $\bar{M}$ and iff $V_T(t)$ is enabled at M.

Proof: ($\Rightarrow$) By contradiction. Let's suppose that transition t is enabled at $\bar{M}$. Furthermore, let's suppose that $V_T(t)$ is not enabled at M. By the definition of the enabled transition of IFS-nets there exists at least one token in every input place of the transition t at $\bar{M}$. This implies that every input place of the transition $V_T(t)$ also contains at least one token because $\mathcal{N}_{IN}$ corresponds IN and M corresponds $\bar{M}$. Let's suppose that $tok_1 \in tk(p_1), \ldots, tok_n \in tk(p_n)$, $\cup_{i=1}^{n} p_i = I(t)$ are tokens at $\bar{M}$ and $\langle d_1, c_1, u_1, v_1 \rangle \in V_P(p_1), \ldots, \langle d_n, c_n, u_n, v_n \rangle \in V_P(p_n)$ are the corresponding tokens of $\mathcal{N}_{IN}$ at M. Consider now a substitution α which bounds the variables of the degree of $V_T(t)$ in the following way: If $\langle x_i, y_i, z_i, w_i \rangle$ is a tuple annotating an arc form an input place $V_P(p_i)$ to transition $V_T(t)$ and $\langle d_i, c_i, u_i, v_i \rangle$ is the token in $V_P(p_i)$ then α substitutes d_i for x_i, c_i for y_i, u_i for z_i and v_i for w_i, where $\cup_{i=1}^{n} p_i = I(t)$. The inspection of definitions 3.19 and 3.20 reveals that the transition $V_T(t)$ is not enabled only if the α-instance of the selector of $V_T(t)$ is false. However, item 2c of definition 4.24 implies that the α-instance of the selector is certainly true $\Rightarrow$ Contradiction $\Rightarrow$ $V_T(t)$ is enabled at M.

($\Leftarrow$) Let's assume that $V_T(t)$ is enabled at M. Every input place of $V_T(t)$ must then contain at least one token. By the correspondence of $\bar{M}$ and M every input place of t then also contains at least one token at $\bar{M}$. According to formula (5) t is thereby enabled at $\bar{M}$. □

Lemma 4.28 and its generalization theorem 4.29 express how the reachability relation of an IFS-net and the reachability relation of the corresponding PrT-net are connected.

Lemma 4.28 Let IN be an IFS-net and $\mathcal{N}_{IN}$ the corresponding PrT-net. Let t be a transition of IN, $\bar{M}_1$ $\bar{M}_2$ markings of IN and M_1, M_2 markings of $\mathcal{N}_{IN}$ such that M_1 corresponds $\bar{M}_1$ and M_2 corresponds $\bar{M}_2$. Then

$$\bar{M}_1 \xrightarrow{t} \bar{M}_2 \text{ iff } M_1[V_T(t)\rangle_{\mathcal{R}_{IN}} M_2. \tag{19}$$

Proof: ($\Rightarrow$) Let IN, $\mathcal{N}_{IN}$, t, $\bar{M}_1$ and $\bar{M}_2$ be as above and let $\bar{M}_1 \xrightarrow{t} \bar{M}_2$ hold. Let M_1 be a marking which corresponds to $\bar{M}_1$. Because t is enabled at $\bar{M}_1$, $V_T(t)$ is enabled at M_1 by lemma 4.27 and thereby $V_T(t)$ may fire at M_1. Let M_2 be the marking which is achieved when $V_T(t)$ fires at M_1. To complete the proof of the claim, it must be shown that M_2 corresponds to $\bar{M}_2$.

Let's suppose that $tok_1 \in tk(p_1), \ldots, tok_n \in tk(p_n)$, $\cup_{i=1}^{n} p_i = I(t)$ are the tokens which are removed from the input places when t fires at $\bar{M}_1$ and

$$\langle d_1, c_1, u_1, v_1 \rangle \in V_P(p_1), \ldots, \langle d_n, c_n, u_n, v_n \rangle \in V_P(p_n) \tag{20}$$

are the corresponding tokens of $\mathcal{N}_{IN}$ at M_1. Furthermore, let's assume that tok is emitted when t fires at $\bar{M}_1$ and respectively $\langle d, c, u, v\rangle$ is emitted when $V_T(t)$ fires at M_1. Let $\bar{M}_3$ and M_3 be markings which are constructed by removing the previously mentioned corresponding tokens from the input places of t and $V_T(t)$. Clearly markings $\bar{M}_3$ and M_3 must be corresponding markings because $\bar{M}_1$ and M_1 are corresponding markings and only corresponding tokens are removed in the construction of $\bar{M}_3$ and M_3. The inspection of item 2(d)ii in definition 4.24 shows that the tokens tok and $\langle d, c, u, v\rangle$ are corresponding tokens in each place p and $V_P(p)$, $p \in O(t)$. But because $\bar{M}_2$ and M_2 can be constructed from $\bar{M}_3$ and M_3 by inserting tok and $\langle d, c, u, v\rangle$ to the output places of t and $V_T(t)$ respectively, $\bar{M}_2$ and M_2 must also be corresponding markings.

($\Leftarrow$) By contradiction. Let IN, $\mathcal{N}_{IN}$, t, M_1, M_2 be as above and assume that $M_1[V_T(t)\rangle_{\mathcal{R}_{IN}} M_2$ also holds. Let $\bar{M}_1$ be a marking of IN such that M_1 corresponds $\bar{M}_1$. Because $V_T(t)$ is enabled at M_1, t is enabled at $\bar{M}_1$ by lemma 4.27. Let $\bar{M}_2$ be the marking which is reached when t fires at $\bar{M}_1$. Let's assume that M_2 does not correspond $\bar{M}_2$. When $V_T(t)$ and t fire at M_1 and $\bar{M}_1$ respectively they remove corresponding tokens from their input places. Thus, the only reason for M_2 and $\bar{M}_2$ not being corresponding markings can be the that a place $p \in O(t)$ receive a token tok when t fires at $\bar{M}_1$, the place $V_P(p)$ receives a token $\langle d, c, u, v\rangle$ when $V_T(t)$ fires at M_1, and $\langle d, c, u, v\rangle$ does not correspond to tok. But according to item 2(d)ii in definition 4.24 $V_T(t)$ and tok correspond each other in the output places of $V_T(t)$ and t respectively $\Rightarrow$ Contradiction $\Rightarrow$ M_2 must correspond to $\bar{M}_2$. □

It should be noted that the markings $\bar{M}_1$, $\bar{M}_2$, M_1 and M_2 in lemma 4.28 need not be the members of the sets RS_{IN} and $[M_0\rangle_{\mathcal{R}_{IN}}$. Thus, the lemma holds for any collection of corresponding markings. This remark is also true for the markings occuring in the next theorem.

Theorem 4.29 Let IN be a IFS-net and $\mathcal{N}_{IN}$ the corresponding PrT-net. Let $t_1, \ldots, t_{n-1}$ be transitions of IN, $\bar{M}_1, \ldots, \bar{M}_\mathrm{n}$ markings of IN and $M_1, \ldots,$ M_n markings of $\mathcal{N}_{IN}$ such that M_i corresponds $\bar{M}_\mathrm{i}$ for $1 \leq i \leq n$, $n \geq 2$. Then

$$\bar{M}_1 \xrightarrow{t_1} \bar{M}_2 \ldots \bar{M}_{\text{n-1}} \xrightarrow{t_{n-1}} \bar{M}_\mathrm{n} \text{ iff } M_1[V_T(t_1)\rangle_{\mathcal{R}_{IN}} M_2 \ldots M_{\text{n-1}}[V_T(t_{n-1})\rangle_{\mathcal{R}_{IN}} M_\mathrm{n}. \quad (21)$$

Proof: By induction on n. Let $n = 2$. Then the task is to show that

$$\bar{M}_1 \xrightarrow{t} \bar{M}_2 \text{ iff } M_1[V_T(t)\rangle_{\mathcal{R}_{IN}} M_2. \quad (22)$$

According to theorem 4.28 this condition indeed holds.

Let's assume that the theorem holds for all n, $2 \leq n \leq m$. Moreover, let t_m be a transition and $\bar{M}_{\text{m+1}}$, $M_{\text{m+1}}$ corresponding markings of IN and $\mathcal{N}_{IN}$

respectively. By theorem 4.28

$$\bar{M}_{\mathrm{m}} \xrightarrow{t_m} \bar{M}_{\mathrm{m+1}} \text{ iff } M_{\mathrm{m}}[v_T(t_m)\rangle_{\mathcal{R}_{IN}} M_{\mathrm{m+1}}. \tag{23}$$

But then must also the claim of the theorem hold for $m+1$. □

With theorem 4.29 it is now easy to show that there exists an one-to-one correspondence between the set of reachable markings of an IFS-net and the set of reachable markings of the corresponding PrT-net.

Theorem 4.30 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net and $\bar{M}$, M corresponding markings of IN and $\mathcal{N}_{IN}$ respectively. Then

$$\bar{M} \in RS_{IN} \text{ iff } M \in [M_0\rangle_{\mathcal{R}_{IN}}. \tag{24}$$

Proof: Let IN, $\mathcal{N}_{IN}$, $\bar{M}$ and M be as above. By item 3 of definition 4.24 and by definition 4.26, the initial markings $\bar{M}_0$ and M_0 correspond each other. By definition 2.4 $\bar{M} \in RS_{IN}$ iff there exists a sequence of transitions $t_1, \ldots, t_n$ and markings $\bar{M}_1, \ldots, \bar{M}_{\text{n-1}}$, $n \geq 1$ such that

$$\bar{M}_0 \xrightarrow{t_1} \bar{M}_1, \ldots, \bar{M}_{\text{n-1}} \xrightarrow{t_n} \bar{M} \tag{25}$$

According to theorem 4.29 formula (25) holds iff there exists markings $M_1, \ldots,$ $M_{\text{n-1}}$ of $\mathcal{N}_{IN}$ such that

$$M_0[v_T(t_1)\rangle_{\mathcal{R}_{IN}} M_1, \ldots, M_{\text{n-1}}[v_T(t_n)\rangle_{\mathcal{R}_{IN}} M. \tag{26}$$

But formula (26) in turn implies that $M \in [M_0\rangle_{\mathcal{R}_{IN}}$. □

Example 4.31 The reachability graph of the net described in example 4.25 is given in figure 5. □

4.3 The Expression of Security Properties

The notions of a secure marking, a secure transition and a secure information flow are the explicit concepts which are exploited when the security of IFS-nets is discussed. The subsequent results show how these concepts are reflected in the PrT-net which corresponds to a given IFS-net. Lemma 4.32 and theorem 4.33 lay the basis for the recognition of secure markings.

Lemma 4.32 Let IN be an IFS-net, p a place and $\bar{M}$ a marking of IN. Let $\mathcal{N}_{IN}$ be the PrT-net corresponding to IN and M the marking corresponding $\bar{M}$. $\bar{M}(p)$ is secure iff every token of place $V_P(p)$ has a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$.

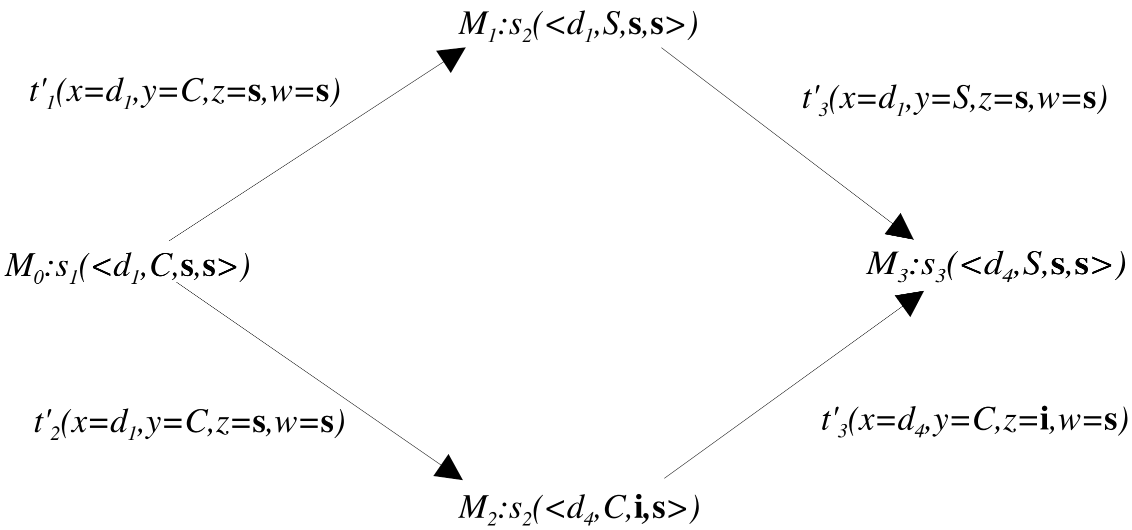

Figure 5: The reachability graph of the PrT-net in figure 4

Proof: ($\Rightarrow$) Let's suppose that $\bar{M}(p)$ is secure. Then by definition 2.5 $\forall tok \in tk(p) : toksec(tok) \in psec(p)$. But now by definition 4.26 and by the definition of the function *psecs* every token of $V_P(p)$ must have a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$.

($\Leftarrow$) By contradiction. Let's suppose that every token of $V_P(p)$ has a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$. Moreover, let's suppose that $\bar{M}(p)$ is not secure. Then according to definition 2.5 the following must hold: $\exists tok \in tk(p) : toksec(tok) \notin psec(p)$. The definition of *psecs* implies that there must exist at least one token of the form $\langle d, c, \mathbf{i}, v\rangle$ $d \in Data$, $c \in SC$, $v \in \{\mathbf{s}, \mathbf{i}\}$ which resides in $\bar{M}(p)$. $\Rightarrow$ Contradiction $\Rightarrow$ $\bar{M}(p)$ is secure. □

Theorem 4.33 Let IN be an IFS-net and $\bar{M}$ a marking of IN. Let $\mathcal{N}_{IN}$ be the PrT-net corresponding IN and M the marking corresponding $\bar{M}$. $\bar{M}$ is secure iff for every place s of M, every token of s has a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$.

Proof: Let's suppose that IN, $\bar{M}$, $\mathcal{N}_{IN}$ and M are defined as above. According to definition 2.5 $\bar{M}$ is secure iff $\forall p \in P : \bar{M}(p)$ is secure. But by lemma 4.32 and by the definition of V_P this holds iff for every place s of M, every token of s has a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$. □

According to theorem 4.33 the security of a marking of an IFS-net can be decided by inspecting all the tokens occuring in the corresponding marking of the corresponding PrT-net. If each token has the symbol **s** in the third position then the marking of the IFS-net is secure. If, however, there exists at least one token which has some other symbol in the third position, then the marking of the IFS-net is not secure.

Definition 2.7 shows that necessary prerequisites for a transition to be secure are facts that the transition is securely enabled at every reachable marking and the firing of the transition is secure. Let's first consider what the former concept means from the PrT-net point of view.

If a transition of an IFS-net is securely enabled at a marking then the markings of the input places of the transition are secure and the security class of the token to be emitted in the firing of the transition obeys the security policy associated with the transition.

Theorem 4.34 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net, t a transition of IN and $\bar{M}$, M markings of IN and $\mathcal{N}_{IN}$ such that M corresponds to $\bar{M}$. Transition t securely enabled at $\bar{M}$ iff $V_T(t)$ is enabled at M and every token of every input place has a form $\langle d_1, c_1, \mathbf{s}, v_1\rangle$ and the token to be emitted by $V_T(t)$ has a form $\langle d_2, c_2, u_2, \mathbf{s}\rangle$, where $d_1, d_2 \in Data$, $c_1, c_2 \in SC$ and $v_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$.

Proof: ($\Rightarrow$) Let's suppose that transition t is securely enabled at $\bar{M}$. By the definition of securely enabled transition t is also enabled at $\bar{M}$ and $\forall p \in I(t)\ \forall tok \in tk(p) : securereceive(tok, p, t) = TRUE$. According to lemma 4.27 $V_T(t)$ is enabled at M. By the definition of *securereceive* predicate $\forall p \in I(t)\ \forall tok \in tk(p) : toksec(tok)$. This leads to the conclusion that every input place of the transition $V_T(t)$ also contains at least one token because $\mathcal{N}$ corresponds IN and M corresponds $\bar{M}$. Furthermore, according to the definition of the function *psecs* and the fact that $\forall p \in I(t)\ \forall tok \in tk(p) : toksec(tok)$ every token of every input place of $V_T(t)$ must have a form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in Data$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$.

By the definition of securely enabled transition $\forall tok_1 \in tk(p_1), \ldots, \forall tok_n \in tk(p_n) : securetransform(tf_t(tok_1, \ldots, tok_n), t) = TRUE$, $\cup_{i=1}^{n} p_i = I(t)$. By the definition of predicate *securetransform* then

$$\begin{aligned} &\forall tok_1 \in tk(p_1), \ldots, \forall tok_n \in tk(p_n) : \\ &\quad lub(toksec(tok_1), \ldots, toksec(tok_n)) \leq toksec(tok), \end{aligned} \tag{27}$$

where tok is the token emitted by t and $\cup_{i=1}^{n} p_i = I(t)$. Let's suppose that $tok_1 \in tk(p_1), \ldots, tok_n \in tk(p_n)$, $\cup_{i=1}^{n} p_i = I(t)$ are tokens at $\bar{M}$ and

$$\langle d_1, c_1, u_1, v_1\rangle \in V_P(p_1), \ldots, \langle d_n, c_n, u_n, v_n\rangle \in V_P(p_n) \tag{28}$$

are the corresponding tokens of $\mathcal{N}_{IN}$ at M. Furthermore, assume that $\langle d, c, u, v\rangle$ is the token to be emitted when $V_T(t)$ fires. Due to the correspondence of the tokens $lub(toksec(tok_1), \ldots, toksec(tok_n)) = lub(y_1, \ldots, y_n)$. But then item 2(d)ii of definition 4.24 together with the definition of the function *tsecs* implies that $v = \mathbf{s}$ holds for the token to be emitted by $V_T(t)$.

($\Leftarrow$) By contradiction. Let's suppose that the transition $V_T(t)$ is enabled at M, every token of every input place has a form $\langle d_1, c_1, \mathbf{s}, v_1 \rangle$ and the token to be emitted by $V_T(t)$ has a form $\langle d_2, c_2, u_2, \mathbf{s} \rangle$, where $d_1, d_2 \in Data$, $c_1, c_2 \in SC$ and $v_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$. By lemma 4.27 t is enabled at $\bar{M}$. Let's suppose that the transition t of IN is not securely enabled at $\bar{M}$.

The inspection of formula (9) reveals that the reason for t not being securely enabled can only be due to the fact that either the predicate *securereceive* or *securetransform* or both are false. The premise that every token of every input place of $V_T(t)$ has a form $\langle d_1, c_1, \mathbf{s}, v_1 \rangle$, $d_1 \in Data$, $c_1 \in SC$ and $v_1 \in \{\mathbf{s}, \mathbf{i}\}$ together with the correspondence of markings $\bar{M}$ and M imply that $\forall p \in I(t)$ $\forall tok \in tk(p) : toksec(tok) \in psec(p)$. But now by formula (6), the value of the predicate *securereceive* is true at $\bar{M}$.

Due to the form of the token to be emitted by $V_T(t)$ and due to the annotation of output arcs given in item 2(d)ii of definition 4.24,

$$\mathbf{s} = tsecs(\langle y_1, \ldots, y_k \rangle, transcls({V_T}^{-1}(t))(y_1, \ldots, y_k)) \tag{29}$$

must hold. By the definition of function $tsecs$ in definition 4.23 formula

$$lub(y_1, \ldots, y_k) \leq transcls({V_T}^{-1}(t))(y_1, \ldots, y_k) \Leftrightarrow \tag{30}$$

$$lub(toksec(tok_1), \ldots, toksec(tok_k)) \leq toksec(tok) \tag{31}$$

also holds were $tok = transcls({V_T}^{-1}(t))(toksec(tok_1), \ldots, toksec(tok_k))$, $\cup_{i=1}^{n} p_i$ $= I(t)$ and each tok_i corresponds the the token entering to $V_T(t)$ from $V_P(s_i)$ and tok corresponds the token to be emitted if $V_T(t)$ fires. The inspection of formula (8) now indicates that the predicate *securetransform* is true at $\bar{M}$. Thus both *securereceive* and *securetransform* are true for any combination of input tokens of t at $\bar{M}$ $\Rightarrow$ Contradiction $\Rightarrow$ t must be securely enabled at $\bar{M}$.
□

The firing of a transition of an IFS-net is considered to be secure if the security classes of the emitted tokens conform with the security classes of the output places. The following theorem shows how a secure firing of a transition can be detected from the corresponding PrT-net.

Theorem 4.35 Let IN be an IFS-net and $\mathcal{N}_{IN}$ the corresponding PrT-net. Let $\bar{M}$ and M be corresponding markings of IN and $\mathcal{N}$ respectively and t a transition of IN such that t is enabled at $\bar{M}$. The firing of t is secure iff the token emitted in the firing of $V_T(t)$ has a form $\langle d, c, \mathbf{s}, v \rangle$, $d \in Data$, $c \in SC$, $v \in \{\mathbf{s}, \mathbf{i}\}$.

Proof: ($\Rightarrow$) Let IN, $\mathcal{N}_{IN}$, $\bar{M}$, M and t be as above and let's suppose that the firing of t is secure. By the lemma 4.27 $V_T(t)$ is enabled at M. Definition 2.6

then implies that for all output places p of t, $securesend(tok, p, t) = TRUE$ holds at the marking $\bar{M}_1$ where tok is the token emitted and $\bar{M}_1$ is the marking immediately reachable from $\bar{M}$ when t fires. According to formula (7) in the definition of the predicate *securesend*, $toksec(tok) \in psec(p)$ for all output places of t at $\bar{M}_1$. Let M_1 be a marking of $\mathcal{N}_{IN}$ which corresponds $\bar{M}_1$. By theorem 4.28 M_1 is reachable from M when $V_T(t)$ fires. Let $\langle d, c, u, v\rangle$, $d \in Data$, $c \in SC$, $u, v \in \{\mathbf{s}, \mathbf{i}\}$ be the token emitted when $V_T(t)$ fires. Because M corresponds to $\bar{M}$ and M_1 corresponds to $\bar{M}_1$ token $\langle d, c, u, v\rangle$ must correspond to tok. But this implies that $c = toksec(tok)$ and furthermore $c \in psec(p)$ for all output places p of t. But this in turn implies that the element $\mathbf{s}$ will be assigned to the third position of the emitted token of $\mathcal{N}_{IN}$, i.e., $u = \mathbf{s}$ and therefore the token transferred to the output places when $V_T(t)$ fires has a form $\langle d, c, \mathbf{s}, v\rangle$, $d \in Data, c \in SC$, $v \in \{\mathbf{s}, \mathbf{i}\}$.

($\Leftarrow$) Suppose that IN, $\mathcal{N}_{IN}$, $\bar{M}$, M and t are defined as above. Moreover, let's assume that $V_T(t)$ is enabled at M. Then t is enabled at $\bar{M}$ by lemma 4.27.

Let M_1 be a marking immediately reachable from M when $V_T(t)$ fires and let $\langle d, c, \mathbf{s}, v\rangle$, $d \in Data$, $c \in SC$, $v \in \{\mathbf{s}, \mathbf{i}\}$ be the token emitted when $V_T(t)$ fires. Let $\bar{M}_1$ be a marking of IN which corresponds M_1. By theorem 4.28 $\bar{M}_1$ is reachable from $\bar{M}$ via the firing of t. Let tok be the token emitted by t. Because M corresponds to $\bar{M}$ and M_1 corresponds to $\bar{M}_1$ token $\langle d, c, \mathbf{s}, v\rangle$ must correspond to tok. But then $toksec(tok) = c$ must hold at $\bar{M}_1$. According to the properties of the function *psecs* given in definition 4.23 and item 2(d)ii of definition 4.24 $\mathbf{s}$ will be assigned to the third position of the emitted token of $\mathcal{N}_{IN}$ only if $c \in psec(p)$ holds for every output place p of t at $\bar{M}_1$. But according to formula (7) this implies that $securesend(tok, p, t)$ is true for every output place p of t at $\bar{M}_1$. According to definition 2.6 this in turn implies that the firing of t at $\bar{M}$ is secure. □

With theorems 4.34 and 4.35 it is now straightforward to prove the following result which binds the concept of secure transition of an IFS-net to the reachable markings of the corresponding PrT-net.

Theorem 4.36 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net and t a transition of IN. Transition t of IN is secure iff $\forall M \in [M_0\rangle_{\mathcal{R}_{IN}}$: If $V_T(t)$ is enabled at M then every token of every input place of $V_T(t)$ has a form $\langle d_1, c_1, \mathbf{s}, v_1\rangle$ and the token emitted in the firing of $V_T(t)$ has a form $\langle d_2, c_2, \mathbf{s}, \mathbf{s}\rangle$, $d_1, d_2 \in Data$, $c_1, c_2 \in SC$, $v_1 \in \{\mathbf{s}, \mathbf{i}\}$.

Proof: ($\Rightarrow$) Let IN, $\mathcal{N}_{IN}$ and t be defined as above. Furthermore, let's assume that t is a secure transition of IN. Then according to definition 2.7 the following conditions hold: for all $\bar{M} \in RS_{IN}$, if t is enabled at $\bar{M}$ then t is securely enabled at $\bar{M}$ and the firing of t is secure.

Let $\bar{M}$ be a marking of the set RS_{IN} such that t is enabled at $\bar{M}$. By theorem 4.30 there exists a marking $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that M corresponds $\bar{M}$ and by lemma 4.27 $V_T(t)$ is enabled at M. Theorem 4.34 implies that every token of every input place of t must have a form $\langle d_1, c_1, \mathbf{s}, v_1\rangle$ and the token to be emitted by $V_T(t)$ has a form $\langle d_2, c_2, u_2, \mathbf{s}\rangle$, $d_1, d_2 \in \mathit{Data}$, $c_1, c_2 \in SC$, $u_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$. On the other hand, the value of u_2 must be $\mathbf{s}$ by theorem 4.35. Thus, the token to be emitted by $V_T(t)$ must have a form $\langle d_2, c_2, \mathbf{s}, \mathbf{s}\rangle$. Because $\bar{M}$ is an arbitrary member of the set RS_{IN} and there exists an one-to-one correspondence between the sets RS_{IN} and $[M_0\rangle_{\mathcal{R}_{IN}}$, the claim holds for every $M \in [M_0\rangle_{\mathcal{R}_{IN}}$.

($\Leftarrow$) Let IN, $\mathcal{N}_{IN}$ and t be as above. Assume that $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ is a marking such that $V_T(t)$ is enabled at M. Then according to the premises every token of every input place of $V_T(t)$ has a form $\langle d_1, c_1, \mathbf{s}, v_1\rangle$ and the token to be emitted when $V_T(t)$ fires has a form $\langle d_2, c_2, \mathbf{s}, \mathbf{s}\rangle$, $d_1, d_2 \in \mathit{Data}$, $c_1, c_2 \in SC$, $u_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$.

Theorem 4.30 implies that there exists a marking $\bar{M} \in RS_{IN}$ such that M corresponds $\bar{M}$ and lemma 4.27 implies that t is enabled at $\bar{M}$. Transition t must be securely enabled at $\bar{M}$ by theorem 4.34 and the firing of t at $\bar{M}$ must be secure by theorem 4.35. But because M is an arbitrary member of $[M_0\rangle_{\mathcal{R}_{IN}}$ and there exists an one-to-one correspondence between the sets $[M_0\rangle_{\mathcal{R}_{IN}}$ and RS_{IN}, the transition t must be securely enabled at every marking of RS_{IN} and the firing of t must be secure at every marking of RS_{IN}. Thus, by definition 2.7 t must be secure. □

As explained in subsection 2.1 information may flow along a path of an IFS-net. Before any results on the flow of information and corresponding PrT-nets can be given, the concept of path of corresponding PrT-net must be defined.

Definition 4.37 Let IN be an IFS-net and $\mathcal{N}_{IN}$ the corresponding PrT-net. Let $c = p_1t_1 \ldots p_kt_kp_{k+1}$, $k \geq 1$ be a path of IN. *Path c' corresponds c* iff $c' = V_P(p_1)V_T(t_1) \ldots V_P(p_k)V_T(t_k)V_P(p_{k+1})$. □

Theorems 4.38 and 4.39 tell how an information flow and a secure information flow occuring in an IFS-net can be perceived from the corresponding PrT-net.

Theorem 4.38 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net, and c a path of IN such that $c = p_1t_1 \ldots p_kt_kp_{k+1}$, $k \geq 1$. There exists an information flow from t_1 to t_k along c iff there exists markings $M_1, \ldots, M_{\mathrm{k}+1} \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that

$$M_1[V_T(t_1)\rangle_{\mathcal{R}_{IN}} M_2 \ldots M_{\mathrm{k}}[V_T(t_k)\rangle_{\mathcal{R}_{IN}} M_{\mathrm{k}+1}. \qquad (32)$$

Proof: Let IN, $\mathcal{N}_{IN}$ and c be as above. Furthermore, let's assume that there exists an information flow from t_1 to t_k along the path c. According to definition 2.10 there exists markings $\bar{M}_1, \ldots, \bar{M}_{\mathrm{k}+1} \in RS_{IN}$ such that

$$\bar{M}_1 \xrightarrow{s} \bar{M}_{\mathrm{k}+1} \Leftrightarrow \bar{M}_1 \xrightarrow{t_1} \bar{M}_2 \xrightarrow{t_2} \ldots \bar{M}_{\mathrm{k}} \xrightarrow{t_k} \bar{M}_{\mathrm{k}+1}, \tag{33}$$

where $s = t_1 t_2 \ldots t_k$, $k \geq 1$. Theorem 4.30 implies that for each $\bar{M}_{\mathrm{i}}$, $1 \leq i \leq k$ there exists $M_{\mathrm{i}} \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that M_{i} corresponds to $\bar{M}_{\mathrm{i}}$. But then according to theorem 4.29 formula (33) holds iff

$$M_1[v_T(t_1)\rangle_{\mathcal{R}_{IN}} M_2 \ldots M_{\mathrm{k}}[v_T(t_k)\rangle_{\mathcal{R}_{IN}} M_{\mathrm{k}+1}. \tag{34}$$

$\square$

Theorem 4.39 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net, and c a path of IN such that $c = p_1 t_1 \ldots p_k t_k p_{k+1}$, $k \geq 1$ and there exists an information flow from t_1 to t_k along c. The information flow from t_i to t_k is secure iff for every marking $\bar{M}_1, \ldots, \bar{M}_{\mathrm{k}+1} \in RS_{IN}$ such that $\bar{M}_1 \xrightarrow{t_1} \bar{M}_2 \xrightarrow{t_2} \ldots \bar{M}_{\mathrm{k}} \xrightarrow{t_k} \bar{M}_{\mathrm{k}+1}$, and for every transition t_i, $1 \leq i \leq k$ the following conditions hold:

1. $M_{\mathrm{i}}[v_T(t_i)\rangle_{\mathcal{R}_{IN}} M_{\mathrm{i}+1}$, where M_{i} corresponds to $\bar{M}_{\mathrm{i}}$ and $M_{\mathrm{i}+1}$ corresponds to $\bar{M}_{\mathrm{i}+1}$,
2. every token of every input place of $V_T(t_i)$ has a form $\langle d_1, c_1, \mathbf{s}, v_1 \rangle$,
3. the token to be emitted by $V_T(t_i)$ has a form $\langle d_2, c_2, \mathbf{s}, \mathbf{s} \rangle$,

where $d_1, d_2 \in \mathit{Data}$, $c_1, c_2 \in SC$ and $v_1 \in \{\mathbf{s}, \mathbf{i}\}$.

Proof: ($\Rightarrow$) Let IN, $\mathcal{N}_{IN}$ and c be defined as above. Assume that the information flow from t_1 to t_k along c is secure. Definition 2.11 then implies that for every pair of markings $\bar{M}_1, \bar{M}_{\mathrm{k}+1} \in RS_{IN}$ such that $\bar{M}_1 \xrightarrow{t_1} \bar{M}_2 \xrightarrow{t_2} \ldots \bar{M}_{\mathrm{k}} \xrightarrow{t_k} \bar{M}_{\mathrm{k}+1}$, $k \geq 1$, $\bar{M}_1, \ldots, \bar{M}_{\mathrm{k}+1} \in RS_{IN}$, and for every transition t_i, $1 \leq i \leq k$ the following conditions hold: transition t_i is securely enabled at $\bar{M}_{\mathrm{i}}$, marking $\bar{M}_{\mathrm{i}+1}$ is the result of the firing of t_i at $\bar{M}_{\mathrm{i}}$ and the firing of t_i is secure. By theorem 4.28 $M_{\mathrm{i}}[v_T(t_i)\rangle_{\mathcal{R}_{IN}} M_{\mathrm{i}+1}$ where $M_{\mathrm{i}}, M_{\mathrm{i}+1} \in [M_0\rangle_{\mathcal{R}_{IN}}$ and M_{i}, $M_{\mathrm{i}+1}$ correspond to $\bar{M}_{\mathrm{i}}$, $\bar{M}_{\mathrm{i}+1}$ respectively, $1 \leq i \leq k$. Thus, condition 1. is satisfied. Because t_i is securely enabled at $\bar{M}_{\mathrm{i}}$ theorem 4.34 implies that every token of every input place of $V_T(t_i)$ has a form $\langle d_i, c_i, \mathbf{s}, v_i \rangle$ and the token emitted when $V_T(t_i)$ fires at M_{i} has a form $\langle d_2, c_2, u_2, \mathbf{s} \rangle$, where $d_1, d_2 \in \mathit{Data}$ $c_1, c_2 \in SC$, $v_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$. According to theorem 4.35 $u_2 = \mathbf{s}$ because the firing of t_i at $\bar{M}_{\mathrm{i}}$ is secure. But then the requirements 2. and 3. are also satisfied.

($\Leftarrow$) Let IN, $\mathcal{N}_{IN}$ and c be defined as above and let $\bar{M}_1, \ldots, \bar{M}_{\mathrm{k}+1} \in RS_{IN}$ be markings such that $\bar{M}_1 \xrightarrow{t_1} \bar{M}_2 \xrightarrow{t_2} \ldots \bar{M}_{\mathrm{k}} \xrightarrow{t_k} \bar{M}_{\mathrm{k}+1}$, $k \geq 1$, and for every transition t_i, $1 \leq i \leq k$, requirements 1.-3. given above hold. Transition $V_T(t_i)$

is enabled by the requirement 1. Requirements 2. and 3. together with theorem 4.34 imply that the transition t_i is securely enabled at $\bar{M}_{\text{i}}$. Furthermore, theorem 4.35 together with the requirement 3. imply that the firing of t_i at $\bar{M}_{\text{i}}$ is secure, $1 \leq i \leq k$. □

The next theorem explains when an IFS-net is information flow secure.

Theorem 4.40 Let IN be an IFS-net, $\mathcal{N}_{IN}$ the corresponding PrT-net. IN is information flow secure iff every token of every marking $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ has a form $\langle d, c, \mathbf{s}, \mathbf{s}\rangle$, $d \in Data$, $c \in SC$.

Proof: ($\Rightarrow$) By contradiction. Let IN and $\mathcal{N}_{IN}$ be defined as above and let IN be information flow secure. Then by definition 2.8 the marking $\bar{M}_0$ is secure and every transition sequence of FS_{IN} is secure. Because the marking $\bar{M}_0$ is secure, definition 2.5 of secure marking together with item 3 of definition 4.24 imply that each token of each place of $\mathcal{N}_{IN}$ at M_0 has a form $\langle d, c, \mathbf{s}, \mathbf{s}\rangle$, $d \in Data$, $c \in SC$.

Let's assume that $M \in [M_0\rangle_{\mathcal{R}_{IN}}$, $M \neq M_0$ and there exists a place s of $\mathcal{N}_{IN}$ such that s contains a token $\langle d, c, u, v\rangle$ at M where $d \in Data$, $c \in SC$ and either $u = \mathbf{i}$ or $v = \mathbf{i}$ or both. Because $M \in [M_0\rangle_{\mathcal{R}_{IN}}$, and IN and $\mathcal{N}_{IN}$ are corresponding nets, there exists a transition t of IN and a marking $M_{\text{a}} \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that $M_{\text{a}}[V_T(t)\rangle_{\mathcal{R}_{IN}} M$ and $\langle d, c, u, v\rangle$ is emitted when $V_T(t)$ fires at M_{a}. If $u = \mathbf{i}$ then according to theorem 4.35 the firing of t can not be secure. If $v = \mathbf{i}$ then theorem 4.34 implies that the transition t is not securely enabled at $\bar{M}_{\text{a}}$, where $\bar{M}_{\text{a}} \in RS_{IN}$ and M_{a} corresponds to $\bar{M}_{\text{a}}$. But then definition 2.7 implies that t can not be a secure transition. Thus, if either $u = \mathbf{i}$ or $v = \mathbf{i}$ then t is not a secure transition of IN.

Because $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ there exist transitions $t'_1, \ldots, t'_k$ of $\mathcal{N}_{IN}$, $k \geq 1$ such that

$$M_0[t'_1\rangle_{\mathcal{R}_{IN}} M_1 \ldots M_{\text{k-1}}[t'_k\rangle_{\mathcal{R}_{IN}} M_{\text{k}}, \tag{35}$$

where $M_1, \ldots, M_{\text{k}} \in [M_0\rangle_{\mathcal{R}_{IN}}$, $t'_k = V_T(t)$, $M_{\text{k-1}} = M_{\text{a}}$ and $M_{\text{k}} = M$. But then according to theorem 4.29 there exist transitions $t_1, \ldots, t_k$ of IN and markings $\bar{M}_1, \ldots, \bar{M}_{\text{k}} \in RS_{IN}$ such that

$$\bar{M}_0 \xrightarrow{t_1} \bar{M}_1 \ldots \bar{M}_{\text{k-1}} \xrightarrow{t_k} \bar{M}_{\text{k}}, \tag{36}$$

where M_{i} corresponds $\bar{M}_{\text{i}}$, $V_T(t_i) = t'_i$, $1 \leq i \leq k$ and $t_k = t$. According to the definition 2.4 transition sequence $s = t_1 t_2 \ldots t_k$ is a member of FS_{IN}. Because $t_k = t$ was shown to be an insecure transition, the sequence s can not be secure $\Rightarrow$ Contradiction $\Rightarrow$ $u \neq \mathbf{i}$ and $v \neq \mathbf{i}$, i.e., $u = \mathbf{s}$ and $v = \mathbf{s}$ $\Rightarrow$ Every token of every marking of $[M_0\rangle_{\mathcal{R}_{IN}}$ must have the form $\langle d, c, \mathbf{s}, \mathbf{s}\rangle$, $d \in Data$, $s \in SC$.

($\Leftarrow$) Let IN and $\mathcal{N}_{IN}$ be defined as above and let every token of every marking have a form $\langle d, c, \mathbf{s}, \mathbf{s}\rangle$, $d \in \mathit{Data}$, $s \in SC$. Theorem 4.33 together with the fact that M_0 corresponds to marking $\bar{M}_0$ imply that $\bar{M}_0$ is a secure marking.

Let t' be a transition of $\mathcal{N}_{IN}$ such that t' is enabled at least at one marking of $[M_0\rangle_{\mathcal{R}_{IN}}$. Theorem 4.36 together with the premises then imply that the transition $t, V_T(t) = t'$, is secure. But because there exist one-to-one correspondence between the transitions of IN and $\mathcal{N}$, every transition of IN must be secure. Then according to definition 2.7 and definition 2.4 every transition sequence of FS_{IN} must be secure. Because $\bar{M}_0$ is secure and every transition sequence of FS_{IN} is secure, IN is information flow secure by definition 2.8. $\square$

Example 4.41 The reachability graph of example 4.31 reveals the fact that the original IFS-net is not information flow secure. To see how this conclusion was drawn, let's consider the reachable markings of the PrT-net. There clearly exist markings where tokens do not have the form $\langle d, c, \mathbf{s}, \mathbf{s}\rangle$, $d \in \mathit{Data}$, $c \in SC$. Theorem 4.40 then says that the original IFS-net can not be information flow secure.

According to definition 2.8 the reason for the insecurity of the IFS-net can only be the insecurity of the initial marking or the existence of an insecure transition sequence in the set FS_{IN}. Because the initial marking M_0 of $\mathcal{N}_{IN}$ corresponds to the initial marking $\bar{M}_0$ of IN and the only token present at M_0 has the form $\langle d_1, C, \mathbf{s}, \mathbf{s}\rangle$, theorem 4.33 implies that the marking $\bar{M}_0$ of IN is secure. The reason for IN for not being information flow secure must then be the existence of an insecure transition sequence in FS_{IN}.

It can be seen from the reachability graph in figure 5 that exactly two sequences of transition instances start from the marking M_0 namely

$$t'_1(x = d_1, y = C, z = \mathbf{s}, w = \mathbf{s})t'_3(x = d_1, y = S, z = \mathbf{s}, w = \mathbf{s}), \tag{37}$$
$$t'_2(x = d_1, y = C, z = \mathbf{s}, w = \mathbf{s})t'_3(x = d_4, y = S, z = \mathbf{i}, w = \mathbf{s}). \tag{38}$$

Recall that transitions t'_1, t'_2 and t'_3 of $\mathcal{N}_{IN}$ correspond to transitions t_1, t_2 and t_3 of IN. By regarding M_2, which is an intermediate marking of sequence (38), to theorem 4.36 it can be seen that neither t_2 nor t_3 is a secure transition of IN. This implies that the firing sequence t_2t_3 can not be secure. On the other hand, the transition instances of the sequence presented by formula (37) and its intermediate markings fulfill the requirements of theorem 4.36. This in turn implies that the transition sequence t_1t_3 is secure.

In the IFS-net of figure 2.14 an information flow may only occur from t_1 to t_3 along the path $c_1 = p_1t_1p_2t_3p_3$ and from t_2 to t_3 along the $c_2 = p_1t_2p_2t_3p_3$. Information flows along these paths indeed occur because, according to the reasoning presented above, $t_1t_3 \in RS_{IN}$ and $t_2t_3 \in RS_{IN}$. Because the firing sequence t_1t_3 is secure, the requirements of definition 2.11 are fulfilled and thus the information flow from t_1t_3 is secure. A similar argument shows that the information flow from t_2 to t_3 is not secure. □

4.4 A Strategy for the Direct Construction of PrT-nets

Whatever technical system is being built, there exist paradigms which help one to structure the required steps of engineering. The mapping between IFS-nets and PrT-nets suggests one such methodology for the Petri net based analysis of security. In this straightforward approach one constructs an IFS-net model from the given system specification, constructs the PrT-net corresponding to the IFS-net, and finally verifies the security properties of the system with the aid of the reachability graph of the PrT-net.

After applying the straightforward approach to a few examples one is tempted to try to construct a PrT-net directly from the specifications of a system in the spirit of the IFS-net to PrT-net mapping. The correct construction of the PrT-net requires the clear understanding of the underlying IFS-net modelling methodology and the identification of certain key aspects from the system specification. An essential part of the IFS-net modelling methodology is the set of those abstract concepts which are used to describe the intuitive concept of a system. Let's first review the system model behind IFS-nets.

Recall that in the context of IFS-nets a system is supposed to consist of atomic processes which may send messages to each other via unidirectional message channels. The security policy of the system is described with a partially ordered set of security classes. A subset of security classes is linked to each message channel. These sets describe the sensitivities of the channels. Each message carries a data item and a security class describing the security rating of the data. When an atomic process has at least one message in each input channel it removes the messages from the input channels and performs a computation based on the data items and the security classes of the incoming messages. After the computation is finished the atomic process sends a message to all output channels. The data item of the outgoing message as well as the associated security class are the results of the computation.

The successful construction of a PrT-net directly from the specifications of a system requires that the following aspects are unambiguously described.

- The static structure of the system.
- The set of possible data items.

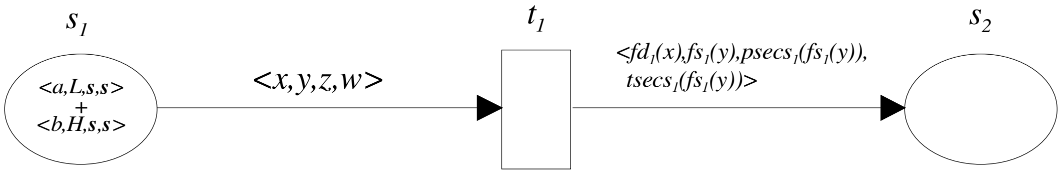

Figure 6: A directly constructed PrT-net

- The security policy.
- The sensitivities of message channels.
- The operations which manipulate data in transitions.
- The operations which manipulate security classes in transitions.
- The initial state of the system.

The static structure of the system, i.e., the atomic processes, the message channels and their relation, induces the pre-net of the PrT-net. The partial order relation of the security policy is needed when function *tsecs* is defined. As pointed out above, the sensitivities of the message channels are expressed in terms of subsets of security classes. These sets are needed in the definition of functions *psec* and *psecs*. The operations which manipulate data and security classes in transitions are needed in the definition of functions *transop* and *transcls*. Finally, the initial state of the system determines the initial distribution of tokens in the constructed PrT-net. After the definition of these functions the actual construction of the PrT-net can be fairly easily done along the lines of definition 4.24.

Example 4.42 Consider a system which consists of two message channels and one atomic process. The task of the atomic process is to read a message from the first channel, do internal computation, and send a message to the second channel.

There occurs only two kinds of data items in messages, namely a and b. Either a security class Low or $High$ is associated with each message of the system. The security policy of the system is described with a lattice in which $High$ is the maximal element. The input channel of the process may contain messages with both Low and $High$ rated messages but the output channel may only contain messages of $High$ sensitivity.

Whenever the process receives a message containing data item a it sends a message containing data item b to output channel. On the other hand, if the process receives a message containing data item b it sends the same data item to the output channel. The atomic process adjusts the security ratings of messages as follows. If the security rating of the data of an incoming message is *Low* then the security rating of the outgoing message is *High* and whenever the incoming security rating is *High* then also the outgoing security rating is *High*. At the initial state the input channel contains two messages: one message carries data item a with security rating *Low* and another message carries data item b with security rating *High*.

The PrT-net of the system is depicted in figure 6. The atomic process of the example system is modelled with transition t_1 and the input and output channels of the process are modelled by places s_1 and s_2 respectively. The arity of both places as well as the tuples annotating the arcs of the net is four as required by the definition 4.24 of corresponding PrT-net. The manipulation of data in atomic process is formally described with the function $fd_1 : \{a, b\} \to \{a, b\}$, $fd_1(a) = b$ and $fd_1(b) = b$. The adjusting of security classes in the atomic process is described by the function $fs_1 : \{L, H\} \to \{L, H\}$, $fs_1(L) = H$ and $fs_1(H) = H$ where L stands for *Low* and H stands for *High*.

The tuples of the arcs of the net describe the structure of messages. The first field of the tuple is reserved for the data item of the message and the the second field is reserved for the security class. The third field is reserved for the symbol which describes whether the token conform with the sensitivity of the place in which the token occurs. The fourth field of the tuple is reserved for the symbol which tells whether the token was produced by a transition firing which obeys the security policy of the system. Symbol **s** in the third field of a token indicates that the token conforms with the sensitivity of the place and **s** in the fourth field of a token indicates that the token is a result of a secure firing. Symbol **i**, however, indicates the violation of these security properties.

The contents of the third field of a token emitted by t_1 is determined by function $psecs_1 : \{L, H\} \to \{\mathbf{s}, \mathbf{i}\}$, $psecs_1(H) = \mathbf{s}$, $psecs_1(L) = \mathbf{i}$. Respectively, the contents of the fourth field of an emitted token is determined by the function $tsecs_1 : \{L, H\} \to \{\mathbf{s}, \mathbf{i}\}$, $tsecs_1(y) = \mathbf{s}$, if $y \leq H$ and $tsecs_1(y) = \mathbf{i}$ otherwise. The semantics of the functions $psecs_1$ and $tsecs_1$ correspond to semantics of the functions $tsecs$ and $psecs$ in definition 4.24. The place parameter has been omitted form $psecs_1$ due to simplicity of the example. For the same reason $tsecs_1$ does not have another security class parameter. □

The construction of PrT-nets directly from specifications leads to an idea which may also be useful in practical engineering. One of the main strengths of PrT-nets is that a transition being enabled depends not only on the existence of tokens in the input places of the transition but also on the selector linked to the transition. Because the firing of a transition may then depend on the data carried by the input tokens, it is far more easier to define complex preconditions for the occurrence of atomic events in the context of PrT-nets than in the context of low-level Petri nets. The mapping of IFS-nets to PrT-nets, however, does not exploit selectors in any way. The mapping links a selector which always allows the firing to each transition of the PrT-net.

The use of more complex selectors does not seem to cause any problems in the analysis of the security properties. This is due to the fact that selectors do not change the contents of tokens emitted by transitions. For example, if selector $x = a$ is associated with the transition of the PrT-net in figure 6, the security analysis is not disturbed in any way. It should therefore be possible to use complex selectors to construct more compact PrT-net models of real systems and still preserve the the connection to the IFS-net based concept of security.

5 The Upgrading and Downgrading of Data

The information flow requirements of practical systems often extend or conflict with the information flows defined by the associated security models. In particular, the need for *the upgrading and downgrading of data* occur frequently in practical systems [4, p.321]. The upgrading of data refers to the situation where an aggregate of data requires a higher classification than the straightforward interpretation of the security classes of individual data items might suggest. The downgrading of data, on the other hand, is needed in situations where the security rating of the object needs to be lowered.

Consider a payroll system where a database contains a record for every employee of an organization. Assume that in addition to the name, position and salary each record also contains more general information about the individual, e.g., the phone number, location of the office, home address etc. Suppose that some curious person knows that his superior has the second highest wage in the organization and he has an access to the database in a way which permits the system to return exact salaries but not any information which could reveal the identity of employees associated with particular salary. Intuitively, the sensitivity of a single record should be higher than the pure number describing the salary. Because the curious employee has a perfectly legal access to every record of the database, he or she can deduce the salary of the superior by querying all the wages of the organization and selecting the second largest result from the set of salaries. But this implies that the sensitivity of the aggregate of all salary figures should be higher than the sensitivity of any single figure alone, i.e., the aggregate of all salary figures should be upgraded.

To illustrate the notion of downgrading in the context of the payroll database suppose that the database is used not only for the payroll computation purposes but also for other, less sensitive activities. Suppose that a telephone operator uses the database when the incoming calls are to be connected to the appropriate employee. The query issued by the operator may certainly not return the salary of the individual but only the phone number and perhaps the location of his or her office. If we assume that the security level of the operators information system software is lower than the software responsible for the payroll computations then the result of the query must be *sanitized*, i.e., too sensitive information must be removed. The security level of the sanitized data must also be adjusted to lower level, that is, the security level of the result must be downgraded.

The upgrading and downgrading of data cause problems in practical systems because not all security policies allow the adjustment of security ratings in the intended way. For example, in the context of Dennings's information flow model, reviewed in subsubsection 1.3, an operation does not assign higher security class to the result of the operation than the least upper bound of the security classes of the input items. Similarly, an operation may not convey information from an object with a particular security rating to another object with a lower security rating.

The introduction of system components which may violate the predefined security policy is naturally a very potential source of security flaws. The upgrading of data is usually considered to be less dangerous operation than the downgrading. The upgrading may cause the data to be given higher security rating than actually needed and therefore cause unnecessary overhead for the manipulation of data, but it does not make it easier to abuse the data. If the sensitive parts of data are not properly removed, the downgrading of data offers an opportunity to those parties which are not originally supposed to exploit data.

A standard approach in the design and implementation of real-world systems is to allow only a small, restricted part of the system, such as *a trusted subsystem*, *a security kernel* or *a trusted process*, to violate the ordinary information flow policy. The aim of this kind structuring is to ease the verification of the correctness of the subsystem responsible for security related functions.

While the upgrading and downgrading of data are important operations in database applications [5], they are also relevant operations in such electronic mail and messaging systems which transfer sensitive information. A classical description of this issue in the context of KSOS operating system can be found from [42, 27].

The purpose of the following subsubsections is to illustrate how the upgrading and downgrading of data are expressed and how the security related reasoning is conducted when a system is modelled with an IFS-net. While many practical issues are abstracted away from the example which will be presented, the problems related to the security resemble the problems found from realistic systems.

5.1 The Description of the System

Consider a system where two active agents, *a reader* and *a writer* communicate via *a database* and *a message channel*. The reader is only supposed to read the database and correspondingly the writer only disposes items to the database. The task of the reader is to query the database, process the results and send messages based on the query results to the writer. The messages between the two parties are conveyed by the message channel. The writer receives messages from the channel and stores them to an internal storage. The writer has an ability to analyze and draw conclusions from certain combinations

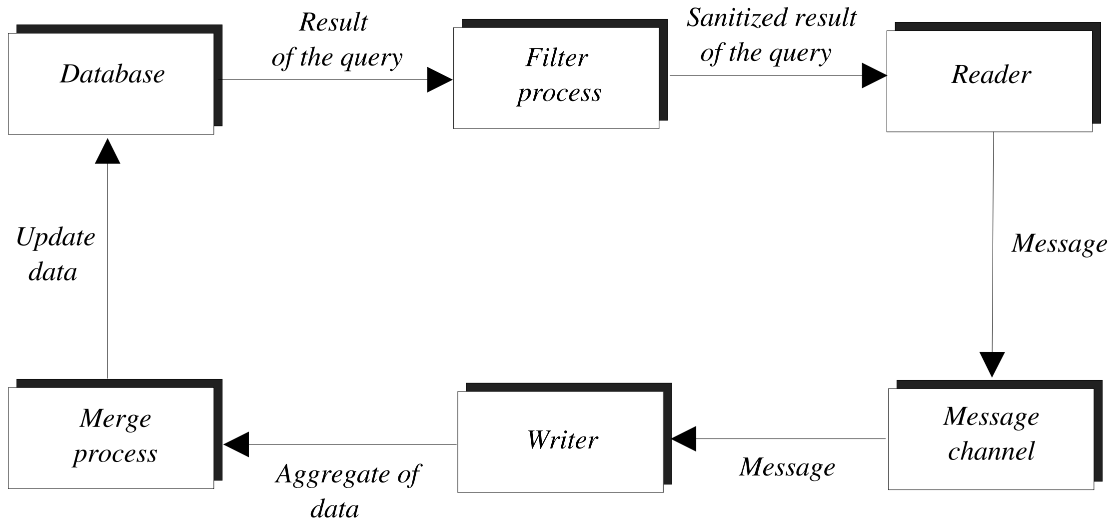

Figure 7: The static structure of the sample system

of incoming messages. The overall structure of the system is presented in figure 7.

The security of the system is based on the control of information flows. The information flow policy is described as a lattice on the set security classes. The members of the domain of the lattice and the related partial order will resemble the intuitive security levels and information flows given in figure 8.

A security class is associated with each item of the database and with each message which is transmitted between the reader and writer. The records of the database are supposed to contain items which the reader are not supposed to see. Therefore the the results of the queries are sanitized by *a filter process* which removes the sensitive material and downgrades the result of the query. Because the information which the writer is able to extract from the groups of incoming messages is considered to be more sensitive than the given information flow policy would imply, *a merge process* is needed to upgrade the aggregates of data prior to the actual database write operation.

A record of the database contains three fields. The data items to be stored to these fields are represented by the members of set $\{a, b, c, d, z\}$. Furthermore, the contents of a database record is described with a string from set $\{a, b, c, d, z\}^3$. For example string abz corresponds to a record in which the first, second and third field contain the data items a, b and z respectively. A security class is associated with every possible data item as follows: a is considered to be *Top secret* information, b, c, d and z are considered to be *Secret*, *Reviewed*, *Classified* and *Unclassified* information respectively. The security ratings of the records of the database are computed from the ratings of the individual items according to the rules of the security policy.

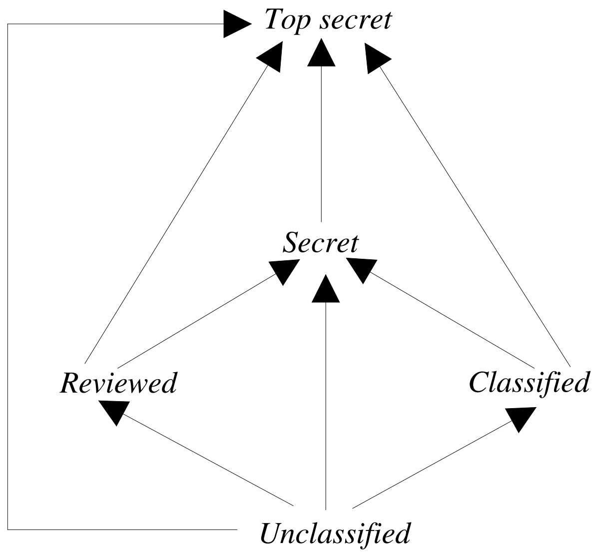

Figure 8: An informal representation of the allowed information flows

The database and the filter process of the system may contain data of any security level. All other parts of the system may only process $Unclassified$, $Reviewed$, $Classified$ or $Secret$ data.

5.2 The IFS-net Model of the System

The aim of this subsection is to present the IFS-net model of the system sketched in the previous subsection. Let $IN = \langle IC, \bar{M}_0 \rangle$ denote the IFS-net to be constructed. The Petri net structure $C = \langle P, T, I, O \rangle$ of IN can be deduced from the figure 9. As given in the figure, $P = \{p_1, \ldots, p_6\}$ and $T = \{t_1, \ldots, t_6\}$. Furthermore, the input and output places of the transitions are defined as follows: $I = \{(t_1, p_1), (t_2, p_2), (t_3, p_3), (t_4, p_3), (t_5, p_3), (t_6, p_4), (t_6, p_5), (t_6, p_6)\}$ and $O = \{(t_1, p_2), (t_2, p_3), (t_3, p_3), (t_4, p_5), (t_5, p_6), (t_6, p_1)\}$. The dashed line boxes around places and transitions specify which places and transitions constitute the various functional parts of the system.

As briefly mentioned above, the contents of the database is modelled with a collection of strings over an alphabet. More specifically, let $DA = \{a, b, c, d, z\}$ be an alphabet. Then $Data = DA^3$, i.e., the set of possible data items is the set of all tree letter strings over the alphabet DA.

The set *Tokens* contains only one token namely tok_1. Recall that different pieces of data may be linked to different instances of the same token; therefore the different instances of tokens may represent different messages.

The set SC is defined as $\{U, R, C, S, T\}$. Classes U, R, C, S and T are the formal counterparts of *Unclassified*, *Reviewed*, *Classified*, *Secret* and *Top secret*

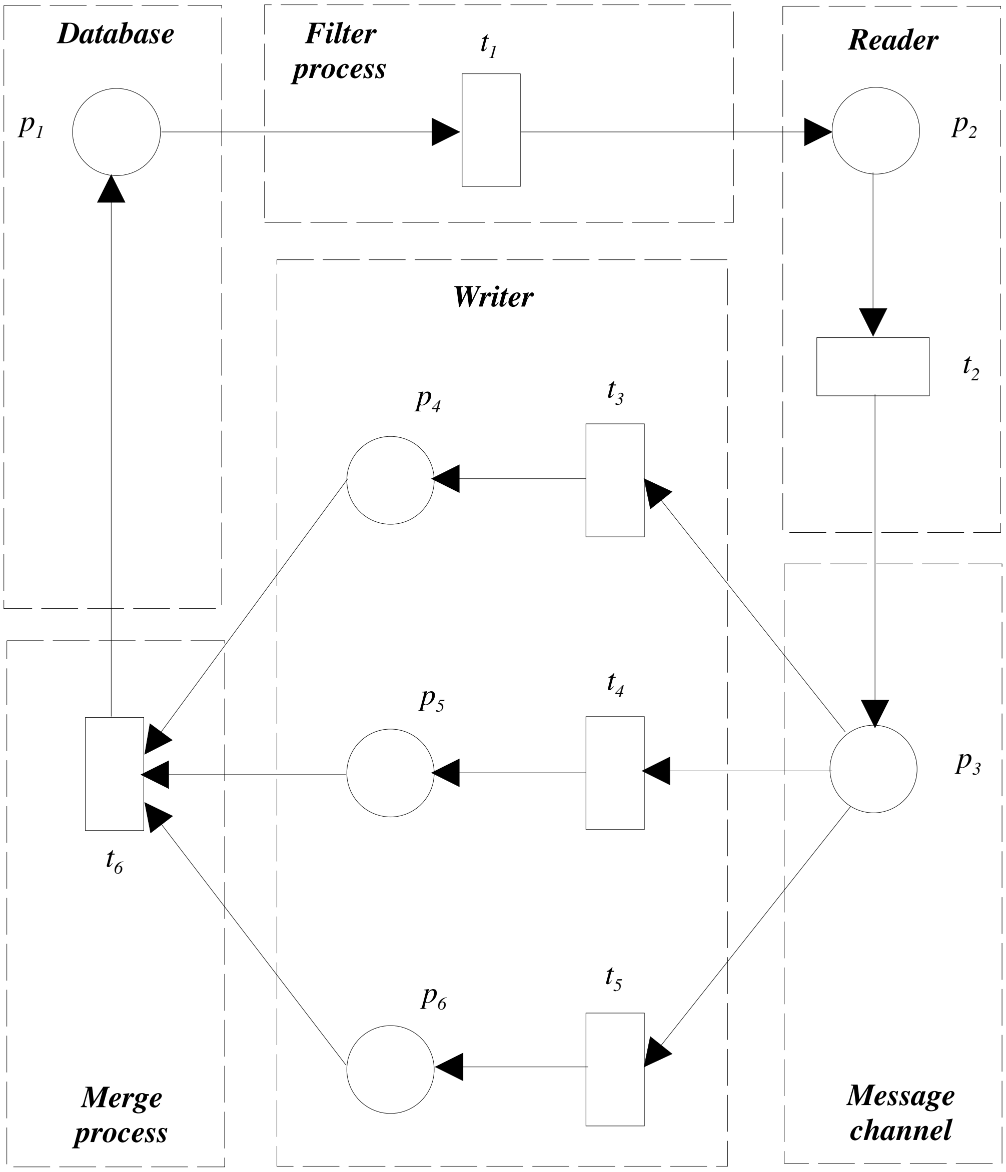

Figure 9: The IFS-net model of the example system

lub	*U*	*R*	*S*	*C*	*T*
U	*U*	*R*	*S*	*C*	*T*
R	*R*	*R*	*S*	*S*	*T*
S	*S*	*S*	*S*	*S*	*T*
C	*C*	*S*	*S*	*C*	*T*
T	*T*	*T*	*T*	*T*	*T*

Table 2: The *lub* operator of the example system

respectively. Because the lattice governing the information flow policy differs a bit from the lattices presented in the standard literature, it is advisable to check that that the partial order and the associated operators indeed form a lattice. Consider a partial order $\langle SC, \leq \rangle$ where the relation $\leq$ has been defined as follows $\{(U,U), (R,R), (S,S), (C,C), (T,T), (U,R), (U,S), (U,C), (U,T), (R,S), (R,T), (S,T), (C,S), (C,T)\}$. As pointed out in subsection 1.3, the partial order is a lattice if it has a finite domain, the domain has a lower bound and there exists a least upper bound operator for the partial order. The set SC is certainly finite and the security class U seems to be the required lower bound of the partial order. Assume that the least upper bound operator lub on SC[3] has been defined as in table 2. The comparison of the definition of partial order and the operator lub reveals that lub is really the least upper bound operator of the partial order at hand. Thus, $\langle SC, \leq \rangle$ is a lattice.

The set OP intuitively contains functions which will be linked to the transitions to describe the relation between the data carried by the input and output tokens. Every transition of the inspected net has either one or three input places and therefore every transition always receive either one or three input tokens. For this reason we only need to consider those functions of OP which have either one or three arguments. Let $OP = \{fd_1, \ldots, fd_6\}$ where $fd_1 : Data \rightarrow Data$ such that $fd_1(abb) = zbb$, $fd_1(dbd) = dzd$, $fd_1(ccc) = ccc$ and $fd_1(w) = zzz$ for each other $w \in Data$, $fd_2 : Data \rightarrow Data$ such that for each $w \in Data$ $fd_2(w) = w$, $fd_3 : Data \rightarrow Data$ such that $fd_3(zbb) = zbb$ and $fd_3(w) = zzz$ for each other $w \in Data$, $fd_4 : Data \rightarrow Data$ such that $fd_4(dzd) = dzd$ and $fd_4(w) = zzz$ for each other $w \in Data$, $fd_5 : Data \rightarrow Data$ such that $fd_5(ccc) = ccc$ and $fd_5(w) = zzz$ for each other $w \in Data$, and finally $fd_6 : Data^3 \rightarrow Data$ such that $fd_6(zbb, dzd, ccc) = abb$ and $fd_6(w) = zzz$ for each other $w \in Data^3$.

As stated in the definition of IFS-nets, the set $CLSFN$ contains those functions which control the modification of security classes of tokens when a transition fires. As with set OP we only need to define those members of

[3] The operator lub is used to denote the least upper bound operator instead of $\oplus$ in the context of the IFS-net model.

CLSFN which have either one or three arguments. More specifically, let $CLSFN = \{fs_1, \ldots, fs_6\}$ where the member functions are defined as follows: $fs_1 : SC \rightarrow SC$ such that $fs_1(T) = S$ and $fs_1(c) = c$ for each other $c \in SC$, $fs_i : SC \rightarrow SC$ such that $fs_i(c) = c$ for every $c \in SC$ for each $i \in \{2, \ldots, 5\}$, and finally $fs_6 : SC^3 \rightarrow SC$ such that $fs_6(c, d, S) = T$, and otherwise $fs_6(c, d, e) = lub(c, d, e)$, where $c, d, e \in SC$.

The set *TF* consists of functions which describe the input-output behaviour of the transitions in respect of tokens. In this example $TF = \{tf_{t_1}, tf_{t_2}, tf_{t_3}, tf_{t_4}, tf_{t_5}, tf_{t_6}\}$ where $tf_{t_i} : Tokens \rightarrow Tokens$ for each $i \in \{1, \ldots, 5\}$, $tf_{t_6} : Tokens^3 \rightarrow Tokens$, and each function of *TF* is a constant function which return tok_1 for each set of input arguments.

The function *transop* associates a function from the set *OP* with each transition in the following way: fd_i with t_i for each $i \in \{1, \ldots, 6\}$. The function *transcls* in turn links a function from the set *CLSFN* to each transition as follows: fs_i to t_i for each $i \in \{1, \ldots, 6\}$. The function *psec* associates a set of security classes with the places of the net: $\{U, R, C, S, T\}$ with p_1, $\{U, R, C, S\}$ with p_2, p_4, p_5, p_6, and $\{U, R, C\}$ with p_3.

Finally, the initial marking of the net is the following: $\bar{M}(p_1) = \{\langle tok_1, T, abb\rangle, \langle tok_1, S, dbd\rangle, \langle tok_1, R, ccc\rangle\}$ and $\bar{M}(p_2) = \bar{M}(p_3) = \bar{M}(p_4) = \bar{M}(p_5) = \bar{M}(p_6) = \emptyset$.

5.3 The PrT-net Model of the System

The construction of the PrT-net corresponding to the IFS-net described in the previous subsection can be straightforwardly done along the lines of chapter 4. Recall that in order to define the corresponding PrT-net for an IFS-net, the mappings of places and transitions must be specified; all the other parts of the PrT-net then follow from definitions 4.23 and 4.24.

Let $IN = \langle IC, \bar{M}_0\rangle$ be the IFS-net defined in the previous subsection and $\mathcal{R}_{IN}$ the relational structure corresponding to IN. Let $\mathcal{N}_{IN} = (N_{\mathcal{N}}, A_{\mathcal{N}}; M_0)$ be a PrT-net such that $\mathcal{R}_{IN}$ is the support of $\mathcal{N}_{IN}$, the pre-net of $\mathcal{N}_{IN}$ is $N_{\mathcal{N}} = (S_{\mathcal{N}}, T_{\mathcal{N}}; F_{\mathcal{N}})$ where $S_{\mathcal{N}} = \{s_1, \ldots, s_6\}$, $T_{\mathcal{N}} = \{t'_1, \ldots, t'_6\}$ and $F_{\mathcal{N}} = \{(s_1, t'_1), (t'_1, s_2), (s_2, t'_2), (t'_2, s_3), (s_3, t'_3), (s_3, t'_4), (s_3, t'_5), (t'_3, s_4), (t'_4, s_5), (t'_5, s_6), (s_4, t'_6), (s_5, t'_6), (s_6, t'_6), (t'_6, s_1)\}$, and the initial marking of the net is $\bar{M}_0(s_1) = \langle aab, T, \mathbf{s}, \mathbf{s}\rangle + \langle dbd, S, \mathbf{s}, \mathbf{s}\rangle + \langle ccc, R, \mathbf{s}, \mathbf{s}\rangle$ and $\bar{M}_0(s_i) = \mathbf{0}$ for $i \in \{2, \ldots 6\}$. It is claimed that $\mathcal{N}_{IN}$ corresponds to IN in the sense of definition 4.24.

To see that $\mathcal{N}_{IN}$ indeed corresponds to IN, consider the mappings V_P and V_T where $V_P : P_{IC} \rightarrow S_{\mathcal{N}}$, $V_P = \{(p_i, s_i) \mid i \in \{1, \ldots, 6\}\}$ and $V_T : P_{IC} \rightarrow S_{\mathcal{N}}$,

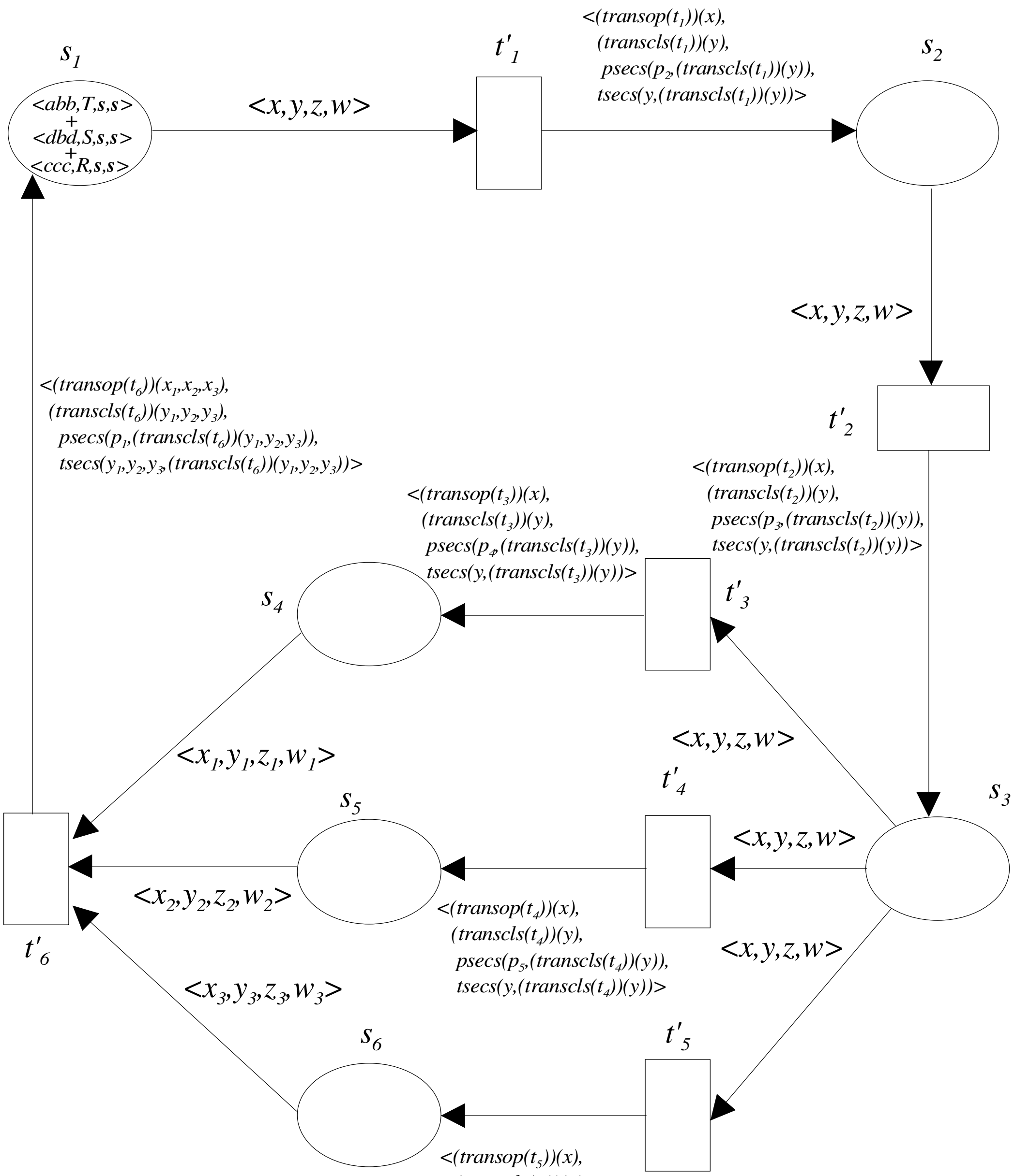

Figure 10: The PrT-net model of the example system

$V_T = \{(t_i, t'_i) \mid i \in \{1, \ldots, 6\}\}$. A closer study reveals that the functions V_P and V_T are bijections which fulfill the requirements of item 1c in definition 4.24. Furthermore, the initial marking assigns symbolic sums to the places of the net exactly as item 3 of definition 4.24 requires. The graphical representation of the constructed net is given in figure 10.

5.4 The Analysis of the Security Properties

Before any statements on the security of the modelled system can be given, the security requisites of the system must be specified. The description of the sample system induces the following requirements.

Requirement 5.43 The example system given in figure 7 fulfills the security requirements iff

1. Insecure information flows occur only in the filter process of the system.
2. The illegal information flows in the filter process are caused by the illegal transformations of security classes, i.e., transformations which do not obey the security policy associated with the example system.

Item 1 of requirement 5.43 stems from the nature of the filter process. Recall that the task of the filter process is to sanitize the incoming messages and "lower" the security rating of the information carried by these messages. The security policy in the IFS-net formalism does not allow the security rating of an information to be transformed into a lower level and therefore the filter process will violate the security policy. The aim of item 2 is to guarantee that even though a restricted and well specified part of the system may violate the security policy, this violation does not cause the system to move to an insecure state.

The security specification of requirement 5.43 given above can be restated in terms of the constructed IFS-net IN as follows.

Requirement 5.44 The example system given in figure 7 fulfills the security requirements iff the IFS-net IN has the following properties:

1. The insecure information flows occur along the path $c = p_1 t_1 p_2$. Information flows along other paths are secure.
2. The security classes of tokens occurring in places p_1 and p_2 must conform with the security ratings of these places. The security classes of tokens occurring in the place p_2 may be results of illegal transformations of security classes in t_1.

Because the filter process of the example system has been modelled with only one transition, the intuitive concept of information flow between transitions reduces to the concept of information flow between transitions t_1 and t_1 in item 1 of requirement 5.44. While this interpretation may look strange at first sight, it is perfectly consistent with the formal definition of information flow given in definition 2.10. The illegal transformation of security classes in t_1 refers to those firings of t_1 where the predicate *securetransform* is false.

Before the security analysis based on the PrT-net corresponding to the IFS-net of the system can be done, the given security specification must be further refined.

Requirement 5.45 The example system given in figure 7 fulfills the security requirements iff the PrT-net $\mathcal{N}_{IN}$ has the following properties:

1. There exists a marking $M_1 \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that token $\langle d_1, c_1, \mathbf{i}, v_1\rangle$ or $\langle d_2, c_2, u_2, \mathbf{i}\rangle$ occurs in s_2 at M_1 and for all $M_2 \in [M_0\rangle_{\mathcal{R}_{IN}}$ every token occurring in s_1, s_3, s_4, s_5 and s_6 at M_2 must have the form $\langle d_3, c_3, \mathbf{s}, \mathbf{s}\rangle$, $d_1, d_2, d_3 \in \mathit{Data}$, $c_1, c_2, c_3 \in SC$ and $v_1, u_2 \in \{\mathbf{s}, \mathbf{i}\}$.

2. For each marking $M \in [M_0\rangle_{\mathcal{R}_{IN}}$ every token occurring in places s_1 and s_2 at M must have the form $\langle d, c, \mathbf{s}, v\rangle$, where $d \in \mathit{Data}$, $c \in SC$ and $v \in \{\mathbf{s}, \mathbf{i}\}$.

The first part of the conjunction given in item 1 of requirement 5.45 is due to theorem 4.39. The second part of the conjunction in turn follows from theorem 4.40. Because places s_1, s_3, s_4, s_5 and s_6 correspond to such places of IN which may not be parts of paths conveying insecure information flow, the tokens occurring in these places must have the required form. Because place s_1 is an image of p_1 under the function V_P and because p_1 is a part of a path which is allowed to convey insecure information, the requirement imposed on the tokens occurring in s_2 may seem odd at first sight. Place p_2, however, is also an end of a path which may not convey insecure information and therefore every token occurring in s_2 must also have the form $\langle d_3, c_3, \mathbf{s}, \mathbf{s}\rangle$, $d_3 \in \mathit{Data}$, $c_3 \in SC$. Correspondingly, item 2(d)iiC of definition 4.24 and item 1 of definition 4.23 help one to translate item 2 of requirement 5.44 into item 2. Requirement 5.45 can be expressed in more compact way as a conjunction of items 1 and 2 if the redundancy related to the requisites of place p_1 is removed.

Requirement 5.46 The example system given in figure 7 fulfills the security requirements iff the PrT-net $\mathcal{N}_{IN}$ has the following properties: for all $M_1 \in [M_0\rangle_{\mathcal{R}_{IN}}$ every token occurring in s_1, s_3, s_4, s_5 and s_6 at M_1 must have the form $\langle d_1, c_1, \mathbf{s}, \mathbf{s}\rangle$ and for all $M_2 \in [M_0\rangle_{\mathcal{R}_{IN}}$ every token occurring in place s_2 at M_2

must have the form $\langle d_2, c_2, \mathbf{s}, v_2\rangle$ and there exists marking $M_3 \in [M_0\rangle_{\mathcal{R}_{IN}}$ such that there occurs a token $\langle d_3, c_3, \mathbf{s}, \mathbf{i}\rangle$ in place s_2 at M_3, where $d_1, d_2, d_3 \in \mathit{Data}$ and $c_1, c_2, c_3 \in \mathit{SC}$ and $v_2 \in \mathit{SC}$.

The next step of the verification is to check that all markings reachable from the initial markings satisfy the requisites imposed by requirement 5.46. A brief study of $\mathcal{N}_{IN}$ reveals that it is infeasible to enumerate and investigate the reachable markings of the net manually. The results to be presented in the subsequent paragraphs were obtained with `PROD` [12]. `PROD` is a PrT-net analysis tool developed in the Digital Systems Laboratory of Helsinki University of Technology. It offers facilities to describe a PrT-net with the net description language and generate as well as inspect the reachability graph of the coded PrT-net. The inspection of the reachability graph is done with the query language which exploits the ideas presented in the context of computational tree logic (CTL) [2]. One of the key features of the query language of `PROD` is the possibility to make queries about the paths of the reachability graph[4]. A good example of the applicability of `PROD` can be found from [29], where the modelling and analysis of a high speed data bus is described.

The analysis of $\mathcal{N}_{IN}$, whose net description language presentation is given in appendix A, reveals the following security related facts:

- Every token occurring in places s_1, s_4, s_5, s_6 at any marking of $[M_0\rangle_{\mathcal{R}_{IN}}$ has the form $\langle d, s, \mathbf{s}, \mathbf{s}\rangle$, $d \in \mathit{Data}$, $s \in \mathit{SC}$.
- Every token occurring in place s_2 at any marking of $[M_0\rangle_{\mathcal{R}_{IN}}$ has the form $\langle d, s, \mathbf{s}, v\rangle$, $d \in \mathit{Data}$, $s \in \mathit{SC}$, $v \in \{\mathbf{s}, \mathbf{i}\}$.
- There exist markings of $[M_0\rangle_{\mathcal{R}_{IN}}$ such that at least one token of place s_2 has the form $\langle d, s, \mathbf{s}, \mathbf{i}\rangle$, $d \in \mathit{Data}$, $s \in \mathit{SC}$ at these markings.
- Every token occurring in place s_3 at any marking of $[M_0\rangle_{\mathcal{R}_{IN}}$ has the form $\langle d, s, u, \mathbf{s}\rangle$, $d \in \mathit{Data}$, $s \in \mathit{SC}$, $u \in \{\mathbf{s}, \mathbf{i}\}$.
- There exist markings of $[M_0\rangle_{\mathcal{R}_{IN}}$ such that at least one token of place s_3 has the form $\langle d, s, \mathbf{i}, v\rangle$, $d \in \mathit{Data}$, $s \in \mathit{SC}$, $v \in \{\mathbf{s}, \mathbf{i}\}$ at these markings.

An annotated `PROD`-session which shows how these facts were found, is given in appendix B.

The facts related to places s_1, s_2, s_4, s_5, s_6 are clearly consistent with the requisites imposed by requirement 5.46. Place s_3, however, does not meet the security specifications associated with $\mathcal{N}_{IN}$. Because $V_P(p_3) = s_3$, there exist

[4] It should be noted that this concept of a path is different from the concept of paths in IFS-nets.

markings of IN such that the marking of p_3 does not conform with the given security specifications. From the given facts one can deduce that the tokens occurring in p_3 are always results of secure firings, but the markings of p_3 are not always secure.

After the tracing the firing sequences which lead to the illegal markings of s_3 and after the checking of the IFS-net definition it becomes evident that the reason for the problem is the definition of the function *psec*. Recall that the task of the function *psec* is to associate a set of security classes with every place of an IFS-net. The intuitive purpose of each set of security classes is to describe the sensitivity of a place, i.e, to define those security classes which can be legally carried by the token occurring in the place. The informal description of the example system in subsection 5.1 says that the reader should be able to process messages with all other security ratings but *top secret*. The function *psec* relates to p_3 the set $\{U, R, C\}$. This means that both S (=*secret*) and T (=*top secret*) rated tokens can not legally occur in place p_3. Thus, there is a modelling error in the constructed IFS-net.

If the erroneously modelled detail of IN is corrected, i.e., *psecs* is modified to assign the set $\{U, R, C, S\}$ to p_3 and the verification is done again, the revised version of the example IFS-net is noticed to fulfill requirement 5.46.

6 Conclusions

The purpose of security models is to lay a basis for reasoning about the security properties of information processing systems. The model of computation and the notion of security are the two central concepts of security models. The model of computation describes abstractly the computational abilities of a security model. The notion of security offers primitives which are used to define when a system is secure.

IFS-nets provide a Petri net based framework for the definition of information flow security models. An inconvenient property of IFS-nets is that the underlying Petri net class is extended in a way which prevents the analysis of the nets without specialized tools. The aim of this work is to show how the security properties of an IFS-net can be inspected indirectly by constructing a PrT-net which corresponds to the IFS-net and by examining the reachability graph of the constructed PrT-net.

To reach the goal of the work, the following steps were taken. Firstly, the correspondence of IFS-nets and PrT-nets was investigated. As a result of this work the mapping of IFS-nets to PrT-nets was presented. Secondly, several results which relate the security properties of IFS-nets to the properties of the reachability graphs of the constructed PrT-nets were proved. Finally, the usage of the method was illustrated in an example where the upgrading and downgrading of data play a major role.

Intuitively speaking, a PrT-net corresponds to a given IFS-net if the Petri net structure of the IFS-net and the pre-net of the PrT-net are isomorphic, the arc annotations obey a given set of constraints and the initial marking of the PrT-net is a predefined function of the initial marking of the IFS-net. The correspondence of the IFS-net and the PrT-net guarantees that there exists an one-to-one correspondence between the set of reachable markings of the IFS-net and the set of reachable markings of the PrT-net. Furthermore the correspondence also guarantees that there exists an one-to-one correspondence between the set of firing sequences of the IFS-net and the set of firing sequences of the PrT-net.

The upgrading and downgrading of data are operations which adjust the security level of data, occur frequently in practical systems, and are difficult to describe with classical security models. The upgrading refers to the operation which assigns higher security rating to the output of computation than the security ratings of the inputs would imply. The downgrading, on the other hand, is an operation which assigns lower security rating to the result of the computation than the security policy would imply. Although these operations are powerful design primitives, they also introduce a major source of security problems. While the erroneous use of the upgrading may cause unnecessary resource consumption, the wrong usage of the downgrading leads to the occurrence of insecure information flows.

The presented mapping between IFS-nets and PrT-nets induces an interesting subject for further research. Because the concept of selector does not exist in IFS-nets, the mapping associates a selector which is always true with the transitions of corresponding PrT-nets. This approach clearly does not exploit the full modelling power of PrT-nets. The preliminary observations suggest that the use of more complex selectors in transitions might lead to more compact PrT-net models and still allow the security analysis to be based on the concepts emerging from IFS-nets.

Acknowledgements

I am very grateful to professor Leo Ojala for pointing out the subject of this work and for encouraging me during my studies. I also wish to thank Lic. Tech. Kimmo Varpaaniemi from the Digital Systems Laboratory of Helsinki University of Technology for the assistance related to the PrT-net tool `PROD`.

I thank my wife Helena and my daughter Iiris for their support and patience. During the preparation of this work I have too often neglected my duties at home.

The support of VTT, formerly known as Technical Research Centre of Finland, is also acknowledged with gratitude.

References

[1] D. F. C. Brewer and M. J. Nash. The Chinese wall security policy. In *1989 IEEE Computer Society Symposium on Security and Privacy*, pages 206–214, Oakland, USA, May 1989. IEEE.

[2] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages*, 8(2):244–263, April 1986.

[3] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[4] D. E. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, Massachusetts, 1982.

[5] D. E. Denning, S. G. Akl, M. Morgenstern, and P. G. Neumann. Views for multilevel database security. In *1986 IEEE Computer Society Symposium on Security and Privacy*, pages 156–172, Oakland, USA, April 1986. IEEE.

[6] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, July 1977.

[7] H. J. Genrich. Predicate/Transition nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, Berlin, 1987.

[8] H. J. Genrich. Equivalence transformations of PrT-nets. In G. Rozenberg, editor, *Advances in Petri nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 179–208. Springer-Verlag, Berlin, 1990.

[9] J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 IEEE Computer Society Symposium on Security and Privacy*, pages 11–20, Oakland, USA, April 1982. IEEE.

[10] J. A. Goguen and J. Meseguer. Unwinding and inference control. In *1984 IEEE Computer Society Symposium on Security and Privacy*, pages 75–46, Oakland, USA, May 1984. IEEE.

[11] R. P. Grimaldi. *Discrete and Combinatorial Mathematics: An applied Introduction*. Addison-Wesley, Reading, Massachusetts, USA, 1989.

[12] P. Grönberg, M. Tiusanen, and K. Varpaaniemi. PROD–a Pr/T-net reachability analysis tool. Series B: Technical Reports 11, Digital Systems Laboratory, Helsinki University of Technology, Otaniemi, Finland, June 1993.

[13] J. Halme, K. Hiekkanen, T. Pyssysalo, and K. Varpaaniemi. *PROD Reference Manual.* Digital Systems Laboratory, Helsinki University of Technology, Otaniemi, Finland, September 1994.

[14] M. Harris Cheheyl, M. Gasser, G. A. Huff, and J. K. Millen. Verifying security. *ACM Computing Surveys*, 13(3):279–339, September 1981.

[15] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[16] P. Huber, A. M. Jensen, L. O. Jepsen, and K. Jensen. Reachability trees for high-level Petri nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*, pages 319–350. Springer-Verlag, Berlin, 1991.

[17] K. Jensen. Coloured Petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, pages 248–299. Springer-Verlag, Berlin, 1987.

[18] K. Jensen. Coloured Petri nets: A high level language for system design and analysis. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 342–416. Springer-Verlag, Berlin, 1990.

[19] K. Jensen and G. Rozenberg, editors. *High-level Petri Nets.* Springer-Verlag, Berlin, 1991.

[20] P. A. Karger. Implementing commercial data integrity with secure capabilities. In *1988 IEEE Computer Society Symposium on Security and Privacy*, pages 130–139, Oakland, USA, April 1988. IEEE.

[21] R. Kling. Controversies about privacy and open information in CSCW. In *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW 92*, pages 15–16, Toronto, Canada, October 1992. ACM.

[22] C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–278, September 1981.

[23] T. M. P. Lee. Using mandatory integrity to enforce “commercial” security. In *1988 IEEE Computer Society Symposium on Security and Privacy*, pages 140–146, Oakland, USA, April 1988. IEEE.

[24] M. Lindqvist. Parameterized reachability trees for Predicate/Transition nets. *Acta Polytechnica Scandinavica, Mathematics and Computer Science Series*, (54), May 1989.

[25] D. McCullough. Noninterference and the composability of security properties. In *1988 IEEE Computer Society Symposium on Security and Privacy*, pages 177–186, Oakland, USA, April 1988. IEEE.

[26] J. K. Millen. A resource allocation model for denial of service. In *1992 IEEE Computer Society Symposium on Security and Privacy*, pages 137–147, Oakland, USA, May 1992. IEEE.

[27] M. A. Padlipsky, K. J. Biba, and R. B. Neely. KSOS–computer network applications. In *AFIPS 1979 National Computer Conference*, pages 373–381, New York, USA, June 1979. AFIPS.

[28] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, 1981.

[29] T. Pyssysalo. The modelling and analysis of a high speed data bus with Predicate/Transition nets. Series A: Research Reports 29, Digital Systems Laboratory, Helsinki University of Technology, Otaniemi, Finland, March 1994.

[30] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1982.

[31] C. Reutenauer. *The Mathematics of Petri Nets*. Prentice-Hall International Ltd, Herdfordshire, 1990.

[32] R. S. Sandhu. Transformation of access rights. In *1989 IEEE Computer Society Symposium on Security and Privacy*, pages 259–268, Oakland, USA, May 1989. IEEE.

[33] R. S. Sandhu. Non–monotonic transformation of access rights. In *1992 IEEE Computer Society Symposium on Security and Privacy*, pages 148–161, Oakland, USA, May 1992. IEEE.

[34] R. S. Sandhu. The typed access matrix model. In *1992 IEEE Computer Society Symposium on Security and Privacy*, pages 122–136, Oakland, USA, May 1992. IEEE.

[35] R. S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, November 1993.

[36] H. Shen and P. Dewan. Access control for collaborative environments. In *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW 92*, pages 51–58, Toronto, Canada, October 1992. ACM.

[37] L. Snyder. Formal models of capability–based protection systems. *IEEE Transactions on Computers*, 30(3):172–181, March 1981.

[38] D. Sutherland. A model of information. In *Proceedings of the 9th National Computer Security Conference*, pages 175–183, Washington, USA, September 1986. AFIPS.

[39] P. S. Thiagarajan. Some behavioural aspects of net theory. *Theoretical Computer Science*, 71(1):133–153, March 1990.

[40] V. Varadharajan. Petri net based modelling of information flow security requirements. In *The Computer Security Foundations Workshop III*, pages 51–61, Franconia, USA, June 1990. IEEE.

[41] V. Varadharajan. Hook-up property for information flow secure nets. In *The Computer Security Foundations Workshop IV*, pages 154–175, Franconia, USA, June 1991. IEEE.

[42] J. P. Woodward. Applications for multilevel secure operating systems. In *AFIPS 1979 National Computer Conference*, pages 319–328, New York, USA, June 1979. AFIPS.

[43] C. Yu and V. D. Gligor. A formal specification and verification method for the prevention of denial of service. In *1988 IEEE Computer Society Symposium on Security and Privacy*, pages 187–202, Oakland, USA, April 1988. IEEE.

Appendix A
The Net Description of the Example of Section 5

The net description language presentation of the PrT-net from figure 10 is divided into two files. The header file `updownex.h` defines symbols and functions occuring in the original PrT-net and the file `updownex.net` contains the net itself.

Both the symbols and the functions occuring in the net description are straightforwardly derived from the original PrT-net. The only exceptions are the functions `psec` and `psecs` and the symbols `MIN_DOM_VAL` and `MAX_DOM_VAL`. Because the description of the net does not directly refer to the function `psec`, but the only references to this function are within `psecs`, the functionality of `psec` is embedded into the function `psecs`. The existence and usage of the symbols `MIN_DOM_VAL` and `MAX_DOM_VAL` are required by the implementation of `PROD`, not the application domain. The contents of the header file `updownex.h` is the following:

```
/*
   Symbolic names for "updownex.net"
*/

/* Data items of the original IFS-net */
#enum aaa,aab,aac,aad,aaz,aba,abb,abc,abd,abz,aca,acb,\
acc,acd,acz,ada,adb,adc,add,adz,aza,azb,azc,azd,azz,baa,\
bab,bac,bad,baz,bba,bbb,bbc,bbd,bbz,bca,bcb,bcc,bcd,bcz,\
bda,bdb,bdc,bdd,bdz,bza,bzb,bzc,bzd,bzz,caa,cab,cac,cad,\
caz,cba,cbb,cbc,cbd,cbz,cca,ccb,ccc,ccd,ccz,cda,cdb,cdc,\
cdd,cdz,cza,czb,czc,czd,czz,daa,dab,dac,dad,daz,dba,dbb,\
dbc,dbd,dbz,dca,dcb,dcc,dcd,dcz,dda,ddb,ddc,ddd,ddz,dza,\
dzb,dzc,dzd,dzz,zaa,zab,zac,zad,zaz,zba,zbb,zbc,zbd,zbz,\
zca,zcb,zcc,zcd,zcz,zda,zdb,zdc,zdd,zdz,zza,zzb,zzc,zzd,zzz

/* Tokens of the original IFS-net */
#enum tok1

/* Possible security classes */
#enum U,R,S,C,T

/* Places of the original IFS-net*/
#enum p1,p2,p3,p4,p5,p6

/* Transitions of the original IFS-net */
#enum t1,t2,t3,t4,t5,t6
```

```
/* Constants used to represent the security of a place */
#enum s,i

/* Null value for the enumeration */
#enum Null

#define MAX_DOM_VAL (aaa)
#define MIN_DOM_VAL (Null)

/*
  Functions which manipulate the data in transitions
*/
int fd1(d)
    int d;
{
  int retval;
  switch(d){
   case abb: retval = zbb; break;
   case dbd: retval = dzd; break;
   case ccc: retval = ccc; break;
   default:  retval = zzz; break;
  }
  return retval;
}

int fd2(d) int d; {return d;}

int fd3(d)
    int d;
{
  int retval;
  switch(d){
   case zbb: retval = zbb; break;
   default:  retval = zzz; break;
  }
  return retval;
}

int fd4(d)
    int d;
{
  int retval;
  switch(d){
   case dzd: retval = dzd; break;
   default:  retval = zzz; break;
  }
  return retval;
}
```

```
int fd5(d)
    int d;
{
  int retval;
  switch(d){
  case ccc: retval = ccc; break;
  default: retval = ccc; break;
  }
  return retval;
}

int fd6(d1,d2,d3)
    int d1,d2,d3;
{return ((d1==zbb && d2==dzd && d3==ccc) ? abb : zzz);}

/*
  Functions which manipulate sets of security classes in
  transitions. Sets have maximum cardinality of three. A
  parameter is reserved for each possible item of the
  input set.
*/
int fs1(cl)
    int cl;
{
  int retval;
  switch(cl) {
   case T: retval = S; break;
   default: retval = cl;
  }
  return retval;
}

int fs2(cl)
    int cl;
{
  int retval;
  switch(cl) {
   default: retval = cl;
  }
  return retval;
}

int fs3(cl) int cl; {return fs2(cl);}

int fs4(cl) int cl; {return fs2(cl);}

int fs5(cl) int cl; {return fs2(cl);}
```

```
int fs6(cl1,cl2,cl3)
    int cl1,cl2,cl3;
{
  int retval;
  if (cl3==S) {
    retval = T;
  } else {
    retval = lub(lub(cl1,cl2),cl3);
  }
  return retval;
}

/*
  Function transop associates to transitions those
  functions which  modify the incoming data.
*/

int (*transop(tr))()
     int tr;
{
  int (*retval)();
  int printf(const char*, ...);
  switch (tr) {
   case t1: retval = fd1; break;
   case t2: retval = fd2; break;
   case t3: retval = fd3; break;
   case t4: retval = fd4; break;
   case t5: retval = fd5; break;
   case t6: retval = fd6; break;
   default:
     printf("\nError in function transop: ");
     printf("Can not catch the transition !\n");
     exit(1); break;
  }
  return retval;
}



/*
  Function transcls associates to transitions those
  modify the security classes of the incoming
  tokens.
*/
int (*transcls(tr))()
     int tr;
{
  int (*retval)();
  int printf(const char*, ...);
```

```
  switch (tr) {
   case t1: retval = fs1; break;
   case t2: retval = fs2; break;
   case t3: retval = fs3; break;
   case t4: retval = fs4; break;
   case t5: retval = fs5; break;
   case t6: retval = fs6; break;
   default:
     printf("\nError in function transcls: ");
     printf("Can not catch the transition !\n");
     exit(1);break;
  }
  return retval;
}

/*
  Function psec associates to transitions those functions which
  modify the security classes of the incoming tokens.
  Function has been embedded to the function psecs
*/

/*
  Function psecs computes the security of a place.
  The information conveyed by the function psec is directly coded
  into this function for practical reasons. Each branch of the switch
  statement tests whether the security class conveyed by the parameter
  seclass is an acceptable security class for the place pl.
*/
int psecs(pl,secclass)
     int pl; int secclass;
{
  int retval;
  int printf(const char*, ...);

  retval = i;
  switch(pl) {
  case p1:
    if (secclass==U || secclass==R || secclass==C ||
secclass==S || secclass==T) {
      retval = s;}; break;
  case p2:
    if (secclass==U || secclass==R || secclass==C ||
secclass==S) {
      retval = s;}; break;
  case p3:
    if (secclass==U || secclass==R || secclass==C) {
      retval = s;}; break;
  case p4:
    if (secclass==U || secclass==R || secclass==C ||
```

```
secclass==S) {retval = s;}; break;
  case p5:
    if (secclass==U || secclass==R || secclass==C ||
secclass==S) {retval = s;}; break;
  case p6:
    if (secclass==U || secclass==R || secclass==C ||
secclass==S) {retval = s;}; break;
  default:
    printf("\nError in function psecs: Can not catch the place !\n");
    exit(1); break;
  }
  return retval;
}

/*
  Function tsecs1 computes the security of a the firing for the
  transitions with only one input transition
*/
int tsecs1(secclass1,secclass2)
     int secclass1; int secclass2;
{
  int lub(); int leq();
  return (leq(secclass1,secclass2) ? s : i);
  /* lub(secclass1) = seclass */
}

/*
  Function tsecs3 computes the security of a the firing for the
  transitions with three input transitions
*/
int tsecs3(secclass1,secclass2,secclass3,secclass4)
     int secclass1,secclass2,secclass3,secclass4;
{
  int lub();int leq();
  return (leq(lub(lub(secclass1,secclass2),secclass3), secclass4) ? s : i);
}

/*
  Function leq defines the partial order for the set of security
  classes.
*/
int leq(a,b)
     int a,b;
{
  int retval;
  int printf(const char*, ...);
  retval = 0;
  switch(a) {
  case U: if (b==U || b==R || b==S || b==C || b==T) retval = 1; break;
```

```
  case R: if (b==R || b==S || b==T) retval = 1; break;
  case C: if (b==C || b==S || b==T) retval = 1; break;
  case S: if (b==S || b==T) retval = 1; break;
  case T: if (b==T) retval = 1; break;
  default:
     printf("\nleq: Can not catch the security class !\n");
     exit(1); break;
  }
  return retval;
}

/*
  Function lub returns the least upper bound for the set SC
*/
int lub(a,b)
     int a,b;
{
/* The order of the security classes is supposed to be
   U,R,S,C,T */
  static int lub_table[5][5] = {{U,R,S,C,T},{R,R,S,S,T},{S,S,S,S,T},
{C,S,S,C,T},{T,T,T,T,T}};
  int row; int column;
  int printf(const char*, ...);
  row = U-a; column = U -b;
  if (row<0 || row>4) {
    printf("\nlub: %i, row index out of range !\n",row); exit(1);
  }
  if (column<0 || column>4) {
    printf("\nlub: %i, column index out of range !\n",column); exit(1);
  }
  return lub_table[row][column];
}
```

The contents of the file `updownex.net` is the following.

```
/*
    The Net Description of the PrT-net of chapter 5
*/

#include "updownex.h"

#place s1 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)\
          mk(<.abb,T,s,s.>+<.dbd,S,s,s.>+<.ccc,R,s,s.> )

#place s2 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)

#place s3 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)
```

```
#place s4 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)

#place s5 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)

#place s6 lo(<.MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL,MIN_DOM_VAL.>)\
  hi(<.MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL,MAX_DOM_VAL.>)

#trans t1_p
in   { s1: <.x,y,z,w.>; }
out  { s2: <.(*transop(t1))(x),(*transcls(t1))(y),\
     psecs(p2,(*transcls(t1))(y)),\
     tsecs1(y,(*transcls(t1))(y)).>;}
#endtr



#trans t2_p
in   { s2: <.x,y,z,w.>; }
out  { s3: <.(*transop(t2))(x),(*transcls(t2))(y),\
                      psecs(p3,(*transcls(t2))(y)),\
     tsecs1(y,(*transcls(t2))(y)).>;}
#endtr

#trans t3_p
in   { s3: <.x,y,z,w.>; }
out  { s4: <.(*transop(t3))(x),(*transcls(t3))(y),\
                      psecs(p4,(*transcls(t3))(y)),\
     tsecs1(y,(*transcls(t3))(y)).>;}
#endtr

#trans t4_p
in   { s3: <.x,y,z,w.>; }
out  { s5: <.(*transop(t4))(x),(*transcls(t4))(y),\
                      psecs(p5,(*transcls(t4))(y)),\
     tsecs1(y,(*transcls(t4))(y)).>;}
#endtr

#trans t5_p
in   { s3: <.x,y,z,w.>; }
out  { s6: <.(*transop(t5))(x),(*transcls(t5))(y),\
                      psecs(p6,(*transcls(t5))(y)),\
     tsecs1(y,(*transcls(t5))(y)).>;}
#endtr

#trans t6_p
in   { s4: <.x1,y1,z1,w1.>; s5: <.x2,y2,z2,w2.>; s6: <.x3,y3,z3,w3.>;}
```

```
out  { s1: <.(*transop(t6))(x1,x2,x3),(*transcls(t6))(y1,y2,y3),
                     psecs(p1,(*transcls(t6))(y1,y2,y3)),\
                     tsecs3(y1,y2,y3,(*transcls(t6))(y1,y2,y3)).>;}
#endtr
```

Appendix B
A Sample PROD-session

The following annotated `PROD`-session shows how the properties of the PrT-net $\mathcal{N}_{IN}$ presented in subsection 5.4 were discovered. The description of the `PROD` command interpreter and query language have been omitted in the subsequent discussion. An interested reader should consult [13] for the complete representation of these issues.

As pointed out above, the net description of $\mathcal{N}_{IN}$ is divided into files `updownex.h` and `updownex.net`. Furthermore, to ease the writing of the query expressions, some useful macros are defined in file `sec_macros`.

```
lisa 151: ls
sec_macros updownex.h updownex.net
lisa 152: cat sec_macros
define submarking(marking, formula) ((marking):(formula))
define secure_marking_field (field[2])
define secure_firing_field (field[3])
define marking_of_place_is_not_secure(place)\
 ((submarking(place,secure_marking_field==i)!=empty))
define every_token_of_place_is_not_result_of_secure_firing(place)\
 ((submarking(place,secure_firing_field==i)!=empty))
lisa 153:
```

The macro `marking_of_place_is_not_secure(s)` can be used to check whether the marking of a place is secure or not. Notice that this macro returns a non-empty set of paths if the marking of the place given as parameter is not secure. Respectively, macro `every_token_of_place_is_not_result_of_secure_firing` can be used to detect tokens which are results of insecure transition firings. This macro returns a non-empty set of paths if the place contains a token which is the result of an insecure firing of a transition.

The first step of the verification is to start `PROD` and load the macros. Command `defs` can be used to print those standard and user defined macros which has been defined during the current session. Command `statistics` returns general information about the generated reachability graph.

```
lisa 153: prod updownex.net
Compiling...
Generating reachability graph.....................
Computing strongly connected components...
0#load sec_macros
0#defs
#define And(f1,f2) (not not ((f1) and (f2)))
#define EventuallyOnAllBranches(f) (not dspan(not (f), true)
((not step true) and not (f)))
#define EventuallyOnSomeBranch(f) (not not dspan(true, false) (f))
```

```
#define HenceforthOnAllBranches(f) (not dspan(true, false) (not (f)))
#define HenceforthOnSomeBranch(f) (not not dspan(not (f), true)
((not step true) and not not (f)))
#define IfThen(f1,f2) (not ((f1) and not (f2)))
#define MAX_DOM_VAL (aaa)
#define MIN_DOM_VAL (Null)
#define NextOnAllBranches(f) (not step not (f))
#define NextOnSomeBranch(f) (not not step (f))
#define Not(f) (not (f))
#define Or(f1,f2) (not ((not (f1)) and not (f2)))
#define UNIX 1
#define UntilOnAllBranches(f1,f2) (not dspan(not (f2), true)
((not ((step true) and (f1))) and not (f2)))
#define UntilOnSomeBranch(f1,f2) (not not dspan(f1, false) (f2))
#define every_token_of_place_is_not_result_of_secure_firing(place)
((submarking(place,secure_firing_field==i)!=empty))
#define marking_of_place_is_not_secure(place)
((submarking(place,secure_marking_field==i)!=empty))
#define secure_firing_field (field[3])
#define secure_marking_field (field[2])
#define submarking(marking,formula) ((marking):(formula))
0#statistics
Number of nodes: 228
Number of (real) arrows: 558
Number of terminal nodes: 25
Number of fact arrow source nodes: 0
Number of fact arrows: 0
Number of nodes that have been completely processed: 228
Number of strongly connected components: 228
Number of nontrivial terminal strongly connected components: 0
0#
```

The security of markings of different places is checked next with the `query` command. Command

```
query volatile node marking_of_place_is_not_secure(s1)
```

visits every marking of the generated reachability graph and checks how many reachability graph nodes satisfy the given macro instance. Because the number of such reachability graph nodes is zero, the macro instance does not hold in any node of the reachability graph. When all places of the $\mathcal{N}_{IN}$ are inspected it turns out that there exist 68 markings where s_3 contains tokens whose security classes do not conform with the security rating of s_3. The other places never contain such tokens.

```
0#query volatile node marking_of_place_is_not_secure(s1)
0 paths
-----------------------------------------------
0#query volatile node marking_of_place_is_not_secure(s2)
0 paths
-----------------------------------------------
```

```
0#query volatile node marking_of_place_is_not_secure(s3)
    6,        9,       11,       13,       14,       18,       26,
   27,       29,       36,       43,       50,       51,       55,
   63,       65,       67,       68,       69,       70,       71,
   72,       73,       75,       77,       79,       80,       81,
   82,       83,       84,       85,       86,       88,       90,
   92,       94,       96,       98,      126,      127,      128,
  129,      130,      131,      132,      133,      134,      163,
  165,      167,      168,      170,      172,      173,      174,
  175,      176,      178,      180,      181,      183,      184,
  185,      187,      188,      222,      223
68 paths
------------------------------------------------
0#query volatile node marking_of_place_is_not_secure(s4)
0 paths
------------------------------------------------
0#query volatile node marking_of_place_is_not_secure(s5)
0 paths
------------------------------------------------
0#query volatile node marking_of_place_is_not_secure(s6)
0 paths
------------------------------------------------
0#
```

The existence of tokens which are results of insecure firings of transitions is checked next. The command which is used to check this property is analogous to the command used to find the insecure markings of the places. It turns out that there exist 37 markings in which place p_2 contains tokens produced by an insecure firing of a transition.

```
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s1)
0 paths
------------------------------------------------
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s2)
    3,       5,        7,       10,       14,       19,       27,
   28,      40,       41,       42,       47,       48,       49,
   55,      56,       57,       58,       59,       60,       61,
   93,      94,       95,       96,       97,       98,      153,
  154,     155,      156,      157,      158,      159,      160,
  161,     220
37 paths
------------------------------------------------
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s3)
0 paths
------------------------------------------------
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s4)
```

```
0 paths
------------------------------------------------
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s5)
0 paths
------------------------------------------------
0#query volatile node\
        every_token_of_place_is_not_result_of_secure_firing(s6)
0 paths
------------------------------------------------
0#
```

Because the occurrence of insecure markings in place p_3 is against the predefined security requirements, lets investigate more closely those firing sequences which lead to insecure markings. Command

```
query bspan(true) marking_of_place_is_not_secure(s3)
```

returns shortest distinct paths from the initial marking of the net to the markings in which s_3 contains tokens whose security classes do not conform with the security rating of s_3.

```
0#query bspan(true) marking_of_place_is_not_secure(s3)
    0  [0>   1  [2>   6,       0  [2>   3  [2>   9,
    0  [0>   1  [0>  4  [1>  11,       0  [0>   1  [1>
    5  [1>  13,       0  [0>   1  [1>   5  [2>  14,
    0  [1>   2  [1>   7  [1>  18,       0  [0>   1  [0>
    4  [0>  10  [0>  26,       0  [0>   1  [0>   4  [0>
   10  [1>  27,      0  [0>   1  [0>   4  [1>  11  [1>  29,
    0  [0>   1  [1>   5  [1>  13  [1>  36,      0  [1>
    2  [1>   7  [1>  18  [1>  43,       0  [0>   1  [0>
    4  [0>  10  [0>  26  [0>  50,       0  [0>   1  [0>
    4  [0>  10  [0>  26  [1>  51,       0  [0>   1  [0>
    4  [0>  10  [1>  27  [1>  55,       0  [0>   1  [0>
    4  [1>  11  [1>  29  [2>  63,       0  [0>   1  [0>
    4  [1>  11  [1>  29  [4>  65,       0  [0>   1  [0>
    4  [1>  11  [1>  29  [6>  67,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [1>  68,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [2>  69,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [3>  70,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [4>  71,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [5>  72,       0  [0>   1  [1>
    5  [1>  13  [1>  36  [6>  73,       0  [1>   2  [1>
    7  [1>  18  [1>  43  [2>  75,       0  [1>   2  [1>
    7  [1>  18  [1>  43  [4>  77,       0  [1>   2  [1>
    7  [1>  18  [1>  43  [6>  79,       0  [0>   1  [0>
    4  [0>  10  [0>  26  [0>  50  [0>  80,       0  [0>
    1  [0>   4  [0>  10  [0>  26  [0>  50  [1>  81,
    0  [0>   1  [0>   4  [0>  10  [0>  26  [0>  50  [2>  82,
    0  [0>   1  [0>   4  [0>  10  [0>  26  [0>  50  [3>  83,
```

```
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [4>   84,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [5>   85,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [6>   86,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [1>   51  [2>   88,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [1>   51  [4>   90,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [1>   51  [6>   92,
  0  [0>    1  [0>    4  [0>   10  [1>   27  [1>   55  [2>   94,
  0  [0>    1  [0>    4  [0>   10  [1>   27  [1>   55  [4>   96,
  0  [0>    1  [0>    4  [0>   10  [1>   27  [1>   55  [6>   98,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [0>  126,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [1>  127,        0  [0>    1  [0>
  4  [0>   10  [0>   26  [0>   50  [0>   80  [2>  128,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [3>  129,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [4>  130,        0  [0>    1  [0>
  4  [0>   10  [0>   26  [0>   50  [0>   80  [5>  131,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [6>  132,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [7>  133,        0  [0>    1  [0>
  4  [0>   10  [0>   26  [0>   50  [0>   80  [8>  134,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [0>  126  [1>  163,        0  [0>    1  [0>    4  [0>
 10  [0>   26  [0>   50  [0>   80  [0>  126  [3>  165,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [0>  126  [5>  167,        0  [0>    1  [0>    4  [0>
 10  [0>   26  [0>   50  [0>   80  [1>  127  [1>  168,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [1>  127  [3>  170,        0  [0>    1  [0>    4  [0>
 10  [0>   26  [0>   50  [0>   80  [1>  127  [5>  172,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [2>  128  [2>  173,        0  [0>    1  [0>    4  [0>
 10  [0>   26  [0>   50  [0>   80  [2>  128  [3>  174,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [2>  128  [4>  175,        0  [0>    1  [0>    4  [0>
 10  [0>   26  [0>   50  [0>   80  [2>  128  [5>  176,
  0  [0>    1  [0>    4  [0>   10  [0>   26  [0>   50  [0>
 80  [3>  129  [3>  178,        0  [0>
  1  [0>    4  [0>   10  [0>   26  [0>   50  [0>   80  [3>
129  [5>  180,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [4>  130  [3>  181,        0  [0>
  1  [0>    4  [0>   10  [0>   26  [0>   50  [0>   80  [4>
130  [5>  183,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [5>  131  [4>  184,        0  [0>
  1  [0>    4  [0>   10  [0>   26  [0>   50  [0>   80  [5>
131  [5>  185,        0  [0>    1  [0>    4  [0>   10  [0>
 26  [0>   50  [0>   80  [6>  132  [5>  187,        0  [0>
  1  [0>    4  [0>   10  [0>   26  [0>   50  [0>   80  [7>
133  [5>  188,        0  [0>    1  [0>    4  [0>   10  [0>
```

```
   26  [0>   50  [0>   80  [0>  126  [2>  164  [2>  195  [0>
  216  [0>  219  [0>  222,        0  [0>    1  [0>    4  [0>
   10  [0>   26  [0>   50  [0>   80  [0>  126  [3>  165  [2>
  196  [0>  217  [0>  220  [0>  223
68 paths
Built set %2
------------------------------------------------
0#
```

To gain more information about the reasons causing the undesirable marking to occur in s_3, let's trace one of the reachability graph paths discovered above. Consider the path `0 [0> 1 [2> 6`. Every number not enclosed in `[` and `>` is an identifier of a node of the reachability graph. Every number enclosed in `[` and `>` denotes an arc from a node of the reachability graph to another node. For example, the given path starts from the node 0 of the reachability graph and the next node of the path is node 1 which is the end point of the arc 0. Commands `look` and `succ` can be used to print information about the current reachability graph node and the edges which are incident from the current node. The number of the current reachability graph node is displayed in the prompt of the command interpreter, e.g., `#0` means that the the current reachability graph node is 0. Command `next` used to select an edge to be followed from the set of incident edges. This command causes the end node of the selected edge to become the current node. Finally, `PROD` session is terminated with the `quit` command.

```
0#look
Node 0, belongs to strongly connected component %%227
  s1: <.dbd,S,s,s.> + <.ccc,R,s,s.> + <.abb,T,s,s.>
------------------------------------------------
0#succ
Arrow 0: transition t1_p, precedence class 0
  x = dbd
  y = S
  z = s
  w = s
to node 1
------------------------------------------------
Arrow 1: transition t1_p, precedence class 0
  x = ccc
  y = R
  z = s
  w = s
to node 2
------------------------------------------------
Arrow 2: transition t1_p, precedence class 0
  x = abb
  y = T
  z = s
  w = s
to node 3
------------------------------------------------
```

```
Node 0 has 3 successor arrows
------------------------------------------------
0#next 0
1#look
Node 1, belongs to strongly connected component %%191
  s1: <.ccc,R,s,s.> + <.abb,T,s,s.>
  s2: <.dzd,S,s,s.>
------------------------------------------------
1#succ
Arrow 0: transition t1_p, precedence class 0
  x = ccc
  y = R
  z = s
  w = s
to node 4
------------------------------------------------
Arrow 1: transition t1_p, precedence class 0
  x = abb
  y = T
  z = s
  w = s
to node 5
------------------------------------------------
Arrow 2: transition t2_p, precedence class 0
  x = dzd
  y = S
  z = s
  w = s
to node 6
------------------------------------------------
Node 1 has 3 successor arrows
------------------------------------------------
1#next 2
6#look
Node 6, belongs to strongly connected component %%190
  s1: <.ccc,R,s,s.> + <.abb,T,s,s.>
  s3: <.dzd,S,i,s.>
------------------------------------------------
6#quit
lisa 154:
```