Department of Computing and Software

Faculty of Engineering — McMaster University

Automated Verification of Information Flow in Agent-Based Systems

by

Khair Eddin Sabri, Ridha Khedri and Jason Jaskolka

C.A.S. Report Series **CAS-09-01-RK**

Department of Computing and Software **January 2009**

Information Technology Building
McMaster University
1280 Main Street West Hamilton, Ontario, Canada L8S 4K1

Contents

Automated Verification of Information Flow in Agent-Based Systems

Khair Eddin Sabri* and Ridha Khedri† and Jason Jaskolka
{sabrike, khedri, jaskolj}@mcmaster.ca

Technical Report CAS-09-01-RK
Department of Computing and Software
McMaster University

January 16, 2009

Abstract

Analyzing information flow is beneficial on ensuring the satisfiability of security policies during the exchange of information between the agents of a system. In the literature, models such as Bell-LaPadula model and the Chinese Wall model are proposed to capture and govern the exchange of information among agents. Also, we find several verification techniques for analyzing information flow within programs or multi-agent systems. However, these models and techniques assume the atomicity of the exchanged information; that is the information cannot be decomposed or combined with other pieces of information. Also, their policies prohibits any transfer of information from a high level agent to a low level agent. In this report, we propose a technique that relaxes these assumptions. Indeed, the proposed technique allows classifying information into frames and articulating finer granularity policies that involve information, its elements, or its frames. It allows as well information manipulation through several operations such as focusing and combining information. Relaxing the atomicity of information assumption permits an analysis that takes into account the ability of an agent to link elements of information in order to evolve its knowledge.

The technique uses global calculus to specify the communication between agents, information algebra to represent agent knowledge, and an amended version of Hoare logic to verify the satisfiability of policies.

Keywords: Global calculus, Information Algebra, Agent Knowledge, Information Flow, Hoare Logic

*This research was supported by the University of Jordan.

†This research was supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

1 Introduction and Motivation

Security is an important aspect that ought to be considered during the software development life cycle. An early detection of a system vulnerability would reduce the cost of addressing it. Information security has three major facets. One facet is confidentiality which is the concealment of information. The second facet is integrity which refers to the trustworthiness of information. The last facet is availability that refers to the ability to use information.

The confidentiality of information ensures that only those with sufficient privileges may access a pre-specified set of information while unauthorized agents should be denied its access. Access control mechanisms are used to protect information from being read or modified by unauthorized agents. Cryptography provides confidentiality in open environments. Once the information is released, it can be transmitted by mistake or malice to unauthorized users. Analysis techniques and prevention and detection mechanisms are necessary to track and control information flows within a system to prevent information from leaking to unauthorized agents.

Models are proposed to anticipate the authorized paths that information should follow and articulate rules for its circulation. For instance, Bell-LaPadula model [2] has its origin in the military and it is widely used in many organizations. The model gives security labels to objects (e.g., information). Each object is considered as one component that cannot be decomposed and is assigned one security level. The security levels form a lattice such that the highest element of the lattice is the most sensitive one. The model also gives security levels to subjects (e.g., agents). Bell-LaPadula model describes a set of rules that proscribe any flow of information from a high level to a lower one. The rules prohibit low-level subject to read high-level objects and high-level subjects to write into low-level objects. We have two remarks on the Bell-LaPadula model. First, the model prohibits any transfer of information from a high level agent to a low level agent. Second, Bell-LaPadula model does not take into consideration policies on composite objects. For example, an agent can access separately a patient's name and lists of drugs administered by a hospital but should not be able to link a patient to a drug.

The Chinese Wall [3] is another model where access to information is not constrained to its security level. Instead, datasets are grouped into conflict of interest classes and an agent can have an access to an information of one of these datasets. For example, assume that there are two banking systems A and B and an oil company C. One policy can group A and B into a conflict of interest class so that an agent can have an access either to A or to B. At the same time, an agent can have an access to C. However, one may want to state a policy that an agent should not be able to associate information together from the bank system and the oil company. For example, the investment of an employee in a company. Similar polices cannot be articulated within the Chinese Wall model.

The manual verification of information flow is extremely demanding in time and resources especially in complex systems. Therefore, formal methods that can constitute the background for sound automation of the analysis of information flow policies becomes necessary. For example, Security Process Algebra (SPA) [6] is a CCS-like process algebra

where actions are partitioned into two security level (*high* and *low*). Using the notion of bisimulation, SPA is used to verify that no information flow is possible from high-level user to low-level user. SPA can specify concurrent system and detect direct and indirect information transmission but it deals only with two security levels and message passing is not specified. Varadharajan [14] proposes an extended Petri-net formalism to model information flow security requirements such as security classes of the output cannot be lower than the security class of any of the received messages. Alghathbar et al. [1] use a logical-based system called FlowUML to validate information flow policies of UML sequence diagrams. Their aim is to detect security flaws at an early stage of the software development life cycle. Typing systems [9, 10, 15] have been widely used for analyzing information flow within the code of programs but not at an earlier stage. Analyzing composite-information flow is not addressed in the techniques presented above.

In this paper, we propose a technique for verifying information flow in agent-based systems. The technique allows classifying information into frames (i.e., classes) and articulating finer granularity policies that involve information, its elements, or its frames. Also, the technique allows analyzing policies governing the flow of composite-information. A composite-information is an information formed from pieces of information of different attributes. For example, a student's name can be seen as an atomic information as it contains only one of the student attributes. Knowing such information may not violate a security policy. However, an unauthorized knowledge of a composite information that consists of a student's name and her grades could cause a security breach in the registrar system. Analyzing the ability of an agent to link together pieces of information of different attributes has several security applications. For example, a security policy may state that an agent should not be able to link a patient name to a disease, a credit card number to a card holder name, or an employee to a salary. Also, the technique removes the restriction that pieces of information should have the same classification in the agents' knowledge.

In Section 2, we introduce the required background. In Section 3, we introduce an example to illustrate the proposed technique. In Section 4, we present the proposed technique. In Section 5, we present related work with a discussion. Finally, we conclude in Section 6.

2 Background

The proposed technique is for analyzing composite-information flow policies in a distributed system where agents are communicating by sending messages. To specify the information flow in a distributed system, we need to specify the knowledge of each of its agents and their communications. We use information algebra [11] for knowledge modeling and global calculus [4] for capturing the communication among agents. The verification is based on an amended version of Hoare Logic [8]. In the following subsections, we introduce these formal settings.

Information algebra [11] is a mathematical structure, introduced by Kholas and Stark, to represent pieces of information and their domains. Having an information algebra to

represent agent knowledge enables classifying information into as many frames as needed. The frames represents the nature of information (e.g., key, secret). Information algebra contains operators to extract and combine information. These operators are used in specifying extracting and inserting information into an agent knowledge. Information algebra also contains a relation to compare the informativeness of pieces of information. This relation is used in verifying the existence of a piece of information in the knowledge of an agent. The developed mathematical structure is expressive as it allows combining information regardless of their frames, extracting a part of information, or associating information with a frame. Also, the structure allows verifying composite information flow policies.

Global calculus [4] is introduced by Carbone et al. to specify the communication between agents. Global calculus originates from the Choreography Description Language, a web service description language developed by W3C WS-CDL working group [7]. It gives global description of the communication between agents and how messages are transmitted between them. An advantage of the calculus is the use of session channel to bind the series of communications between two agents that belong to the same protocol run. Session channels can be associated with types which offers a high-level of abstraction of complex communication behaviours. Furthermore, the calculus is expressive and can be used to represent initiating a session, in-session communication, branching, conditional, and recursion.

The verification of information flow is based on Hoare logic. We rephrase the weakest precondition axioms such that they can be applied to distributed systems specified using global calculus and information algebra. To automate the verification, we develop a prototype tool. The tool finds the weakest precondition from a system specification. Then, it makes use of theorem prover PVS to check whether weakest precondition holds. For now, we are proving formula in PVS through the interactive mode. However, we are in progress of developing strategies to maximize the automation of proving formula.

2.1 Information Algebra

In [11], Kholas and Stark explore connections between different representations of information. They introduce a mathematical structure called *information algebra*. This mathematical structure involves a set of information Φ and a lattice D. In the rest of this report, we denote elements of Φ by φ, ψ and χ. Each piece of information is associated with a *frame* (also called domain in [11]), and the lattice D is the set of all frames. Each frame x contains a *unit element* e_x which represents the empty information. Information can be combined or restricted to a specific frame. Combining two pieces of information φ and ψ is represented by $\varphi\psi$. Information φ and ψ can be associated with different frames, and $\varphi\psi$ is associated with a more precise frame than φ and ψ. Kholas and Stark [11] assume that the order of combining information does not matter and, therefore, the combining operator is both commutative and associative. Restricting an information φ to a frame x is denoted by $\varphi^{\downarrow x}$ which represents only the part of φ associated with x.

In the following definition and beyond, let $(D, \curlyvee, \curlywedge)$ be a lattice and x and y be elements of D called frames (also called domain in [11]). Let $\preccurlyeq$ be a binary relation between frames

such that $x \curlyvee y = y \Leftrightarrow x \preccurlyeq y$. Let Φ be a set of information and φ, ψ, χ be elements of Φ. We denote the frame of information $\varphi \in \Phi$ by $d(\varphi)$. Let e_x be the empty information over the frame $x \in D$, the operation $\downarrow$ be a partial mapping $\Phi \times D \rightarrow \Phi$ to restrict an information to a specific domain, and $\cdot$ be a binary operator to combine pieces of information. For simplicity, to denote $\varphi \cdot \psi$, we write $\varphi\psi$.

Definition 2.1 (Information Algebra [11])**.** *An information algebra is a system* (Φ, D) *that satisfies the following axioms:*

1. $(\varphi\psi)\chi = \varphi(\psi\chi)$
2. $\varphi\psi = \psi\varphi$
3. $d(\varphi\psi) = d(\varphi) \curlyvee d(\psi)$
4. $x \preccurlyeq y \Rightarrow (e_y)^{\downarrow x} = e_x$
5. $d(\varphi) = x \Rightarrow \varphi e_x = \varphi$
6. $\forall(x \mid x \in D : d(e_x) = x)$
7. $x \preccurlyeq d(\varphi) \Rightarrow d(\varphi^{\downarrow x}) = x$
8. $x \preccurlyeq y \preccurlyeq d(\varphi) \Rightarrow (\varphi^{\downarrow y})^{\downarrow x} = \varphi^{\downarrow x}$
9. $d(\varphi) = x \ \wedge \ d(\psi) = y \Rightarrow (\varphi\psi)^{\downarrow x} = \varphi(\psi^{\downarrow x \wedge y})$
10. $x \preccurlyeq d(\varphi) \Rightarrow \varphi\varphi^{\downarrow x} = \varphi$ □

The first two axioms indicate that the set of pieces of information together with the combining operator forms a semi-group. Axiom 3 states that the frame of two pieces of information combined is the join of their frames. Axioms (4-6) give properties of the empty information e_x. Axioms (7-8) give the properties of focusing an information to a specific frame. Axioms (9-10) give properties that involve combining and focusing of information.

As we will illustrate in Section 4, having an information algebra to represent agent knowledge allows verifying policies that involve a flow of composite information as well as expressing policies similar to that of Bell-LaPadula and Chinese Wall models as shown in Section 5.

2.2 Global Calculus

In [4], Carbone et al. introduce a typed calculus to specify communication-centred systems called global calculus. We use global calculus to specify the communication between agents because it gives the view of messages moving between agents. This is the global view of the system to be analyzed. In contrast, π, end-point, CCS, and CSP calculi give the view of the local behaviour of each agent. Also, global calculus is expressive as it can be used to represent initiating a session, in-session communication, branching, conditional, and recursion. Below, we give the global calculus syntax, taken from [4].

$$\text{(EMPTY)}\ \frac{-}{\{P\}\ \mathsf{skip}\ \{P\}} \qquad \text{(ASSIGN)}\ \frac{-}{\{P[x/E]\}\ x{:=}E\ \{P\}} \qquad \text{(COMP)}\ \frac{\{P\}\ S\ \{Q\},\ \{Q\}\ T\ \{R\}}{\{P\}\ S;T\ \{R\}}$$

$$\text{(COND)}\ \frac{\{B\wedge P\}\ S\ \{Q\},\{\neg B\wedge P\}\ T\ \{Q\}}{\{P\}\ \mathsf{if}\ B\ \mathsf{then}\ S\ \mathsf{else}\ T\ \{Q\}} \qquad \text{(LOOP)}\ \frac{\{B\wedge P\}\ S\ \{P\}}{\{P\}\ \mathsf{while}\ B\ \mathsf{do}\ S\ \{\neg B\wedge P\}}$$

Figure 1: Hoare logic inference rules

```
I::= A → B:ch(vs̃).I        (initiation of a session)
  |  A → B:s(op,e,y).I     (communication over a session channel s)
  |  x@A=e.I               (assigning the value of e to x at A)
  |  if e@A then I1 else I2(condition)
  |  (vs)I                 (new)
  |  X                     (recvar)
  |  I1|I2                 (parallel)
  |  I1+I2                 (choice)
  |  μX.I                  (recursion)
  |  0                     (inaction)
```

The terms I_1 and I_2 are called interactions, ch is a service channel, s is a session channel, $\tilde{s}$ is a vector of session channels, A and B are agents, x and y are local variables in each agent, and X is a term variable.

We use global calculus to specify the communication between agents and link it to information algebra. The linkage requires modifying some terms of global calculus that we discus in Section 4.

2.3 Hoare Logic

Hoare logic is used to reason about program correctness with respect to program specifications rather than in terms of how the program could be executed. Verification is based on the Hoare triple $\{P\}\ S\ \{Q\}$. In the triple, S is a program statement or a sequence of statements. The predicate P is the precondition that characterises the initial states for which the program is being defined. The predicate Q is the postcondition that specifies the final states after the execution of the program. In order to verify $\{P\}\ S\ \{Q\}$, one needs to prove that $P \Rightarrow wp(S, Q)$ where $wp(S, Q)$ is the weakest precondition for the program S and the postcondition Q. The weakest precondition can be found by using the inference rules of Hoare logic given in Figure 1.

To use Hoare logic in verifying information flow, we rephrase the weakest precondition inference rules such that they can be applied to distributed systems specified using global calculus and information algebra.

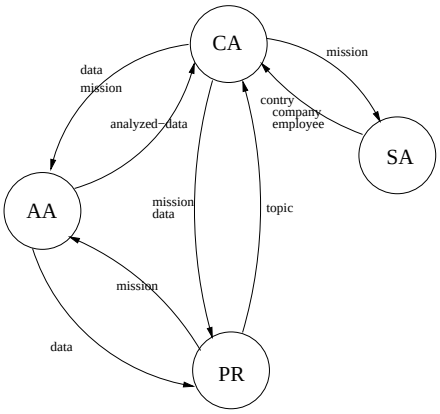

Figure 2: The communication between CA, SA, AA, and PR

3 Illustrative Example

We give an example to illustrate the proposed technique in analyzing composite-information flow in a distributed system. We will use this example as a running example throughout this paper. Our illustrative system consists of four agents: Coordinator Agent (CA), Operations Officer (OO), Analyst Agent (AA), and Public Relation Agent (PR). These agents are communicating be sending messages that contain information. A message also specifies the frames to be associated with the transmitted information in the receiver knowledge. The communication between agents is shown in Figure 2.

The CA sends a *mission* to the OO who collects data regarding this mission and sends it to the CA. The *data* communicated to the coordinator can be classified as either *country*, *company*, or *employee*. Then, the CA communicates with the AA to analyze the data received from the OO. The CA sends out to AA the pieces of information classified as *country* and *company* as well as the *ID* of the officier that sent the information. The AA classifies the received information as *data*. The CA also sends the *mission* to AA. Then, AA runs the required analysis, sends the *analyzed data* back to the CA. The PR communicates with the CA to get information regarding a specific *topic*. The CA sends the *mission* related to the topic to PR. Also, CA sends her pieces of information classified as *country*, which are associated with that topic. These pieces of information are classified as *data* in PR. The PR can seek details from the AA. In this case, the AA sends her *data* to PR which are classified as *data* as well at PR.

Each communication pattern between two agents can be seen as a protocol by itself. Therefore, for this example, we have four protocols I_1, I_2, I_3, and I_4. For instance, the protocol I_1 represents the communication pattern between the CA and the OO. The protocol I_2 gives the communication pattern between CA and AA. The protocol I_3 gives the communication pattern between PR and CA and the fourth protocol I_4 gives the communication pattern between PR and AA. In this example, the second protocol should follow the first one and can run in parallel with the third and fourth protocols.

We use the proposed technique to specify the communication between agents and their knowledge. Also, we use it to specify and verify polices on the exchange of information among these agents.

Note that, in the communication, an agent can send an empty information (nothing). Also, for information classification, one can use labels other than what we use in this example (*data*, *mission* etc.). We use these labels to specify policy as shown in Section 4.3.

4 The Proposed Technique

First, we tackle agent knowledge representation which is based on information algebra. Then, we use global calculus to represent the communication between agents and link it to their knowledge. Finally, we amend the set of inference rules of Hoare logic by adding new rules and rephrasing the known ones.

The technique assumes that each agent has its own knowledge and information classifications. For example, a piece of information classified as *country* in an agent's knowledge may be classified as *data* in another agent's knowledge. In this representation, we remove the restriction of having the same classification in all the knowledges of the agents of a system. We represent the communication between agents as message passing. Each transmitted message contains: (1) an information that can be composite, and (2) the frames to be assigned to the transmitted information at the receiver's knowledge. In some applications as in cryptographic protocols, the message would contain a condition that the receiver should satisfy to extract the information such as possessing the cipher and the key in its knowledge.

4.1 Knowledge Representation

In [13], we develop a mathematical structure to specify agent knowledge and prove that it is an information algebra. The explicit knowledge of an agent is represented by two elements Φ and D. The set Φ consists of pieces of information (we use the words information and piece of information interchangeably) available to the considered agent. There is no restriction on the representation of these pieces of information. They can be represented as formulae as in artificial intelligence literature, functions, etc. In this paper, we represented pieces of information as functions. While D is a lattice of frames such that each piece of information is associated with a frame.

Definition 4.1 (Agent Information Frame)**.** *Let* $\{\mathbb{A}_i \mid i \in I\}$ *be a family of sets indexed by the set of indices* I *and* $\mathcal{P}(\mathbb{A}_i)$ *be the powerset of* $\mathbb{A}_i$*. An information frame* D_I *is defined as:* $D_I \triangleq \prod_{i \in I} \mathcal{P}(\mathbb{A}_i)$ *Which can be equivalently written as a set of functions as*
$D_I \triangleq \{f : I \to \bigcup_{i \in I} \mathcal{P}(\mathbb{A}_i) \mid \forall(i \mid i \in I : f(i) \in \mathcal{P}(\mathbb{A}_i))\}$ □

Let $J \subseteq I$ and $\mathcal{I}_J \subseteq I \times I$ such that $\mathcal{I}_J = \{(x, x) \mid x \in J\}$ (i.e., $\mathcal{I}_J$ is the identity on J). Given the frame D_I, we can define D_J as $\{g \mid \exists(f \mid f \in D_I : g = \mathcal{I}_J ; f)\}$ where ; denotes relational composition. In [12], we prove that $(\{D_J\}_{J \in I}, \curlyvee, \curlywedge)$ is a lattice where

$\curlyvee$ is the join of two frames while $\curlywedge$ is their meet. For simplicity, we use D to denote the lattice $(\{D_J\}_{J\subseteq I}, \curlyvee, \curlywedge)$. On the lattice D and for D_J and D_K frames in D, it is known [5, page 39] that we have $D_J \preccurlyeq D_K \Leftrightarrow (D_J \curlyvee D_K = D_K) \Leftrightarrow (D_J \curlywedge D_K = D_J)$. Figure 3 shows a lattice constructed from the set of indices $I = \{company, country\}$.

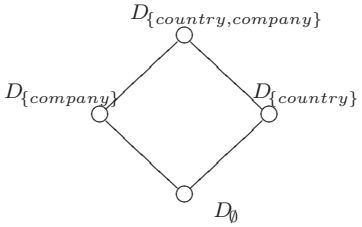

Figure 3: A lattice constructed from $I = \{company, country\}$

It should be noted that the lattice that we have is different from that of Bell-LaPadula model. Our aim from this lattice representation is to represent frames of atomic information as in $D_{\{country\}}$ and $D_{\{company\}}$ and to represent frames of composite information as in $D_{\{country,\ company\}}$. The frame $D_{\{country,\ company\}}$ contains composite information that represents a company associated with a country. We compare in Section 5 the proposed technique with Bell-LaPadula model.

For the given example, the set of relevant frames of CA is indexed by $I_C = \{$*officerID, mission, country, company, employee, analyzed-data, topic*$\}$, the set of frames of OO is indexed by $I_O = \{$*mission, data*$\}$, the set of frames of AA is indexed by $I_A = \{$*mission, data, analyzed-data*$\}$, and the set of frames of PR is indexed by $I_P = \{topic, mission,\ data\}$.

An information φ is a function which can be written as a set of 2-tuples (i, A) where i is an index and A is a set. The initial knowledge of the coordinator (Φ^C, D^C) can contain one piece of a composite information φ such that $\Phi^C = \{\varphi\}$ where $\varphi = \{($*officierID,* $\{$*JohnDo*$\})$, $($*mission,* $\{$*Cobra*$\})$, $($*topic,* $\{$*Economy*$\})\}$. We denote the domain of an information by using the labelling operator d. The domain of φ is $d(\varphi) = D_{\{officierID,\ mission,\ topic\}}$. The initial set of pieces of information of OO can contain three pieces of information $\Phi^O = \Big\{$ $\{($*mission,* $\{$*Cobra*$\})$, $($*data,* $\{$*France*$\})\}$, $\{($*mission,* $\{$*Cobra*$\})$, $($*data,* $\{$*AirFrance*$\})\}$, $\{($*mission,* $\{$*Cobra*$\})$, $($*data,* $\{$*Manager*$\})\}$ $\Big\}$. The initial set of pieces of information of PR can contain one piece of information $\Phi^P = \Big\{\{($*topic,* $\{$*Economy*$\})\}\Big\}$. Finally, the initial set of pieces of information of AA is the empty set.

The set of information of each agent can be represented in a tabular format. For example Table 1 gives Φ^{CA}.

Table 1: The set Φ^C in a tabular format

	Topic	*country*	*company*	*employee*	*mission*	*analyzed-data*	*officerID*
φ_1	Economy					Cobra	JohnDo

Each frame D_J contains a special element called the *empty information* e_{D_J} and is defined as $\{(i, \emptyset) \mid i \in J\}$. Whenever, it is clear from the context, we write e_J instead of e_{D_J}. An information φ is called *atomic* if $\varphi = e_\emptyset$ or $d(\varphi) = D_{\{j\}}$ for $j \in I$. A piece of information can be seen as a row in a table where the table header represents the indices of the frame of an information. An empty information can be seen as a table with only header and $e_\emptyset$ can be seen as empty page that dose not contain even the header. An atomic information can be seen as a one cell of the table or as an empty page.

For the following definitions, let $d(\varphi) = D_J$ and $d(\psi) = D_K$. We define a binary operator $\cdot$ (however, we write $\varphi\psi$ to denote $\varphi \cdot \psi$) to combine information. We use this operator to represent composite information that is composed of pieces of information.
$\varphi\psi \triangleq \{(i, A) \mid i \in J \cap K \ \wedge \ A = \varphi(i) \cup \psi(i)\}$
$\cup \ \{(i, A) \mid i \in J - K \ \wedge \ A = \varphi(i)\} \ \cup \ \{(i, A) \mid i = K - J \ \wedge \ A = \psi(i)\}$.

We also define a binary operator $\downarrow\colon \Phi \times D \to \Phi$ to extract a part of an information that belongs to a specific frame as $\varphi^{\downarrow D_J} \triangleq \mathcal{I}_J{:}\varphi$ where D_J is a frame and φ is an information such that $D_J \in D$ and $\varphi \in \Phi$. The $\downarrow$ operator can be used to extract a specific kind of information. For example, let $\varphi = \{(\mathit{OfficierID}, \{\mathit{John\ Do}\}), (\mathit{mission}, \{\mathit{Cobra}\}), (\mathit{topic}, \{\mathit{Economy}\})\}$, then $\varphi^{\downarrow D_{\{mission\}}} = \{(\mathit{mission}, \{\mathit{Cobra}\})\}$.

We define a partial order relation $\leq$ on information as $\varphi \leq \psi \Leftrightarrow J \subseteq K \ \wedge \ \forall(i \mid i \in J \ : \ \varphi(i) \subseteq \psi(i))$ and we say that ψ is *more informative* than φ. This relation indicates whether an information is a part of another one. We use this relation to verify the existence of an information in the knowledge of an agent; an information can be in the knowledge of an agent as a part of a composite information. The special element $e_\emptyset$ of $D_\emptyset$ is the least informative information i.e., $\forall(\varphi \mid \varphi \in \Phi \ : \ e_\emptyset \leq \varphi)$.

In addition to the information algebra operators, we define an operator to remove a piece of information from another one as
$\varphi{-}\psi \triangleq \{(i, A) \mid i \in J{\cap}K \ \wedge \ A = \varphi(i){-}\psi(i)\} \ \cup \ \{(i, A) \mid i \in J{-}K \ \wedge \ A = \varphi(i)\}$. We prove the following proposition by using the definition of the operators and the properties of sets. We give the proof in Appendix B.

Proposition 4.1. *Let φ and ψ be two pieces of information, and let e_J be the empty information on D_J*

1. $d(\varphi - \psi) = d(\varphi)$
2. $\varphi - e_J = \varphi$
3. $e_J - \varphi = e_J$
4. $\varphi \leq (\psi - \chi) \Rightarrow \varphi \leq \psi$
5. $\varphi \leq \psi \Rightarrow \varphi - \psi = e_{d(\varphi)}$
6. $(\varphi\psi - \psi)^{\downarrow d(\varphi)} \leq \varphi$
7. $\varphi \leq \psi \Rightarrow (\chi - \varphi)\psi = \chi\psi$ □

The proposition gives some properties of the remove operator such as Proposition 4.1(1) indicates that removing pieces from an information does not change the frame of that information. Proposition 4.1(2, 3) states that removing an empty piece from an information does not affect that information, and removing pieces of information from the empty information does not change the empty information. Also, the proposition relates the more

informative relation with the remove operator as shown in Proposition 4.1(4,5). Proposition 4.1(6,7) relates the remove operator with the combine operator.

As we assume the frame of pieces of information can be changed during their transmission from one agent to another, we define a frame substitution function that substitute a part of a frame of an information with another. We define *frame substitution* as $fs(\varphi, D_J, D_K) \triangleq \varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J])$ where the sets J and K are singleton subsets of the set of indices I and $d(\varphi) = D_L$.
For example, let $\varphi = \{(country,\{France\}),(topic,\{economy\})\}$. Then, $fs(\varphi, D_{\{country\}}, D_{\{data\}})$ $= \{(data, \{France\}), (topic, \{economy\})\}$ and
$fs(\varphi, D_{\{country\}}, D_{\{topic\}}) = \{(topic, \{France,\ economy\})\}$.

Proposition 4.2. *Let J and K be singleton subsets of the set of indices I*

1. $D_J \preccurlyeq d(\varphi) \quad \vee \quad \varphi = fs(\varphi, D_J, D_K)$
2. $D_K \preccurlyeq d(\varphi)) \quad \vee \quad \varphi = fs(fs(\varphi, D_J, D_K), D_K, D_J)$ □

We give the proof in Appendix C. We find the other cases of Proposition 4.2(1) do not not hold. For example, A counterexample of disproving $\neg(D_J \preccurlyeq d(\varphi)) \ \vee \ D_K \preccurlyeq d(\varphi) \ \vee \ \varphi = fs(\varphi, D_J, D_K)$ is to have $\varphi = \{(country,\{France\}),(topic,\{economy\})\}$ and $fs(\varphi, D_{\{country\}}, D_{\{data\}}) = \{(data, \{France\}), (topic, \{economy\})\}$. Also, we find that $\neg(D_J \preccurlyeq d(\varphi)) \quad \vee \quad \varphi = fs(\varphi, D_J, D_K)$ is not satisfied. For example, $fs(\varphi, D_{\{country\}}, D_{\{topic\}}) = \{(topic, \{France,\ economy\})\} = \{(country, \{France,\ economy\})\}$.

Similarly, we find the other cases of Proposition 4.2(2) do not hold as well. For example $fs(fs(\varphi, D_{\{country\}}, D_{\{topic\}}), D_{\{topic\}}, D_{\{country\}}) = \{(country, \{France,\ economy\})\}$ and $fs(fs(\varphi, D_{\{data\}}, D_{\{topic\}}), D_{\{topic\}}, D_{\{data\}}) = \{(country, \{France\}), (data, \{economy\})\}$.

The *knowledge* of each agent is modeled as an information algebra $\mathcal{N} \triangleq (\Phi, D)$. Based on the operators of information algebra, we introduce several functions that we use later for specifying communication between agents.

- $isInKnowledge(\mathcal{N}, x, \varphi) \triangleq \exists(\psi \mid \psi \in \Phi : x \in D \ \wedge \ \varphi \leq \psi \ \wedge \ x \preccurlyeq d(\psi))$. This function verifies the existence of an information in the knowledge $\mathcal{N}$ that is associated with the frame x and is more informative than φ.
- $extract(\mathcal{N},\ x,\ \varphi) \triangleq \{\psi^{\downarrow x} \mid x \in D \ \wedge \ \psi \in \Phi \ \wedge \ \varphi \leq \psi \ \wedge \ x \preccurlyeq d(\psi)\}$. This function extracts the pieces of information from the knowledge $\mathcal{N}$ that contains φ and restrict them to the frame x.

As Φ in $\mathcal{N}$ is a set, the operatorts on sets can be applied on Φ. For protocol specification, we use the insert and update functions. The insert function $insert(\mathcal{N}, \varphi)$ inserts the information φ into Φ. While the update function $update(\mathcal{N},\ \psi,\ \varphi)$ updates the information ψ with φ in Φ. Formally, $update(\mathcal{N},\ \psi,\ \varphi) \triangleq (\{(\chi - \psi) \cdot \varphi \mid \chi \in \Phi \ \wedge \ \psi \leq \chi\} \ \cup \ \{\chi \mid \chi \in \Phi \ \wedge \ \neg(\psi \leq \chi)\},\ D)$. In the insert and update functions, there is always a condition that $d(\varphi) \in D$. We define the function $choose(\Phi)$ to select a piece of information randomly from Φ. If Φ is empty, it returns the empty information $e_\emptyset$. We prove the following propositions which help in verifying policies.

Proposition 4.3. *Let φ and ψ be pieces of information and let $\mathcal{N}$ be a knowledge.*

1. $\varphi \leq \psi \Rightarrow update(update(\mathcal{N},\ \varphi,\ \psi),\ \varphi,\ \chi) = update(\mathcal{N},\ \varphi,\ \psi \cdot \chi)$
2. $isInKnowledge(\mathcal{N},\ d(\varphi),\ \varphi) \quad \vee \quad update(\mathcal{N},\ \varphi,\ \psi) = \mathcal{N}$

Proof. The detailed proof can be found in Appendix D.

1. The proof applies the definitions of *update*. Also, it uses the set union axiom, the distributivity axiom, the trading rule for $\exists$, the nesting axiom, Proposition 4.1(4), the substitution axiom, and properities of propositional logic.
2. We use the definitions of update and *isInKnowledge* functions. Also, the proof uses Proposition A.2(1) and properities of set theory.

□

4.2 Specification of Subsystem Communication

To link global calculus with agent knowledge representation in order to use it to specify distributed systems, we explicitly articulate communication, assignment, and conditional terms that involve the knowledge of agents in the context of information algebra.

The *Communication Term* in global calculus is $A \rightarrow B : s\langle op, e, y\rangle$. It is used to express the communication between agents A and B using the channel s, where A is the sender, B is the receiver, op is an operator name used in the communication, and e is an expression evaluated at A that its value is stored in the variable y at B. The operator op does not have a semantics. It is mainly used to have a structured communication between agents and for type checking.

To specify the exchange of information among agents of a system, we represent the expression e as an information φ. We also give a semantics to the operator op which contains a condition c on the receiver knowledge to extract information from the transmitted message and indicates the frame of the information at the receiver knowledge (i.e., frame substitution for the transmitted information). We represent the operator as $op(c, D_J/D_K)$. Therefore, we represent the communication step as $A \rightarrow B : s\langle op(c, D_J/D_K), \varphi, y\rangle$. This step indicates that agent A sends an information φ to agent B. If the condition is satisfied in the knowledge of B, then B applies frame substitution to φ and stores the result in its local variable y. Otherwise, B stores $e_\emptyset$ in y. We can generalize the operator to apply more than one frame substitution at one step.

The Assignment Term: In global calculus, the term $x@A := e$ is used to specify assigning the value of the expression e to the variable x located at A. Since we represent the knowledge of agents as an information algebra, the assignment term in our specification becomes $insert(\mathcal{N}^A, \varphi)$ and $update(\mathcal{N}^A,\ \varphi,\ \psi)$ depending on the context. The knowledge $\mathcal{N}^A \triangleq (\Phi^A, D^A)$ represents the knowledge of agent A.

The Conditional Term: In global calculus, the term *if* $e@A$ *then* I_1 *else* I_2 is used to specify selecting one of the branches I_1 and I_2 based on the evaluation of the Boolean

expression e at the agent A. Since we represent the knowledge of the agents as an information algebra, the expression e is based on the knowledge of agent A and is represented as $isInKnowledge(\mathcal{N}, x, \varphi)$.

The communication between agents, which is presented in the example, can be specified in global calculus as $(I_1; I_2)|I_3|I_4$. The parallel operator between two terms is equivalent to the all possible interleaving between their steps. Below, we give the specification.

The first protocol represents the communication pattern between the CA and the OO which is specified in global calculus as:

$\mathtt{I}_1 :=$
1 CA $\rightarrow$ OO : ch(vv).
2 CA $\rightarrow$ OO : v($op(\{\mathtt{true}\}, D_{\{mission\}}/D_{\{mission\}})$,
$choose(extract(D_{\{\mathtt{mission}\}}, \{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\}, \mathcal{N}^{\mathtt{OO}}))$
x).
3 OO $\rightarrow$ CA : v($op(\{\mathtt{true}\}, D_{\{country\}}/D_{\{data\}})$,
$choose(extract(D_{\{\mathtt{data}\}}, \{(\mathtt{data}, \{\mathtt{France}\})\}, \mathcal{N}^{\mathtt{CA}}))$
x_1).
4 OO $\rightarrow$ CA : v($op(\{\mathtt{true}\}, D_{\{company\}}/D_{\{data\}})$,
$choose(extract(D_{\{\mathtt{data}\}}, \{(\mathtt{data}, \{\mathtt{AirFrance}\})\}, \mathcal{N}^{\mathtt{CA}}))$
x_2).
5 OO $\rightarrow$ CA : v($op(\{\mathtt{true}\}, D_{\{company\}}/D_{\{data\}})$,
$choose(extract(D_{\{\mathtt{data}\}}, \{(\mathtt{data}, \{\mathtt{Manager}\})\}, \mathcal{N}^{\mathtt{CA}}))$
x_3).
6 update CA $\{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\}$ $\{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\} \cdot x_1 \cdot x_2 \cdot x_3$.
7 0

The second protocol represents the communication pattern between the CA and the AA which is specified in global calculus as:

$\mathtt{I}_2 :=$
1 CA $\rightarrow$ AA : ch(vt).
2 CA $\rightarrow$ AA : t($op(\{\mathtt{true}\}, [D_{\{\mathtt{data}\}}/D_{\{\mathtt{officerID}\}}, D_{\{\mathtt{data}\}}/D_{\{\mathtt{country}\}}, D_{\{\mathtt{data}\}}/D_{\{\mathtt{company}\}}])$,
$choose(extract(D_{\{\mathtt{mission,country,company,officerID}\}}, \{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\}, \mathcal{N}^{\mathtt{CA}}))$
x).
3 insert AA $x \cdot \{(\mathtt{analysed\text{-}data}, \{\mathtt{Performance}\})\}$.
4 AA $\rightarrow$ CA : t($op(\{\mathtt{true}\}, D_{\{\mathtt{analyzed\text{-}data}\}}/D_{\{\mathtt{analyzed\text{-}data}\}})$,
$choose(extract(D_{\{\mathtt{analyzed\text{-}data}\}}, x^{\downarrow D_{\{\mathtt{mission}\}}}, \mathcal{N}^{\mathtt{AA}}))$
z).
5 update CA $\{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\}$ $\{(\mathtt{officerID}, \{\mathtt{JohnDo}\})\} \cdot z$.
6 0.

The third protocol represents the communication pattern between the CA and the PR which is specified in global calculus as:

$I_3 :=$
1 PR $\rightarrow$ CA : ch(vs).
2 PR $\rightarrow$ CA : s($op(\{\mathtt{true}\}, D_{\{topic\}}/D_{\{topic\}})$,
$choose(extract(D_{\{\mathtt{topic}\}}, \{(\mathtt{topic}, \{\mathtt{economy}\})\}, \mathcal{N}^{\mathtt{PR}}))$
y).
3 CA $\rightarrow$ PR : s($op(\{\mathtt{true}\}, D_{\{data\}}/D_{\{country\}})$,
$choose(extract(D_{\{\mathtt{mission,\ country}\}}, y, \mathcal{N}^{\mathtt{CA}}))$
x).
4 update PR $\{(\mathtt{topic}, \{\mathtt{Economy}\})\}$ $\{(\mathtt{topic}, \{\mathtt{Economy}\})\} \cdot x$.
5 0

The first step represents initiating a session between PR and CA. The second step represents sending a message through the channel s. This message contains a topic. The third step specifies sending a composite information that is associated with the frames mission and country. The message indicates through the operator that the frame *country* to be substituted with the frame *data* at PR. The condition for extracting the information is true i.e., no condition. The last step specifies updating the knowledge of PR.

The fourth protocol represents the communication pattern between the PR and the AA which is specified in global calculus as:

$I_4 :=$

1 PR $\rightarrow$ AA : ch(vu).

2 PR $\rightarrow$ AA : u($op(\{\mathsf{true}\}, D_{\{mission\}}/D_{\{mission\}})$,
$choose(extract(D_{\{\mathtt{mission}\}},\ x^{resInf\{\mathtt{mission}\}},\ \mathcal{N}^{\mathtt{PR}}))$
y).

3 AA $\rightarrow$ PR : u($op(\{\mathsf{true}\}, D_{\{data\}}/D_{\{data\}})$,
$choose(extract(D_{\{\mathtt{data}\}},\ y,\ \mathcal{N}^{\mathtt{CA}}))$
z).

4 update PR $\{(\mathtt{Topic}, \{\mathtt{topic1}\})\}\ \{(\mathtt{Topic}, \{\mathtt{topic1}\})\} \cdot \pi_1(y)$.

5 0.

4.3 Analysis

We analyze security policies in multi-agent systems by using an amended version of Hoare logic. In the proposed technique, a *policy* is a predicate on the knowledges of a set of agents. A precondition P in the Hoare triple $\{P\}S\{Q\}$ represents agents initial knowledge, S represents the specification of a communication protocol expressed in global calculus and information algebra, and the postcondition Q represents the negation of a policy on the knowledge of the considered agents. The postcondition is expressed as a predicate articulated using terms of the language of information algebra.

To verify a policy, we first calculate the weakest precondition of the protocol based on the postcondition (negation of a policy) and protocol specification. Then, we evaluate the term $P \Rightarrow wp(S, Q)$. If the evaluation is true, then the policy is not satisfied, otherwise, it is satisfied.

The inference rules of Hoare logic are generally articulated in term of primitives of a programming language. In Figure 4, we rephrase some of them in terms of the language of information algebra and global calculus.

(INACTION) $\dfrac{-}{\{P\}\ 0\ \{P\}}$ (COMM) $\dfrac{\{P\wedge C\}\ \ y@B{:=}f_{S(\varphi,D_J,D_K)}\{Q\}\quad \{\neg P\wedge C\}\ \ y@B{:=}e_{\emptyset}\{Q\}}{\{P\}A{\rightarrow}B{:}s(op(C,D_J/D_K),\varphi,y)\{Q\}}$

(INIT) $\dfrac{-}{\{P\}\ \ A{\rightarrow}B{:}ch(v\tilde{s})\ \ \{P\}}$ (UPDATE) $\dfrac{-}{\{P[\mathcal{N}^A/\mathsf{update}(\mathcal{N}^A,\varphi,\psi)]\}\ \mathsf{update}(\mathcal{N}^A,\varphi,\psi)\ \ \{P\}}$

(INSERT) $\dfrac{-}{\{P[\mathcal{N}^A/\mathsf{insert}(\mathcal{N}^A,\varphi)]\}\ \mathsf{insert}(\mathcal{N}^A,\varphi)\ \ \{P\}}$ (COMP) $\dfrac{\{P\}\ \ S\ \ \{Q\},\ \{Q\}\ \ T\ \ \{R\}}{\{P\}\ \ S;T\ \ \{R\}}$

Figure 4: Weakest precondition inference rules for the verification of information flow

The (INACTION) and (INIT) terms are considered to be equivalent to the skip program as they do not have any affect on the knowledge of the communicating agents. The (INSERT) and (UPDATE) terms are considered to be equivalent to the assignment statement. Therefore, their weakest precondition is substituting the agent knowledge $\mathcal{N}^A$ with $insert(\mathcal{N}^A, \varphi)$ or $update(\mathcal{N}^A,\ \varphi,\ \psi)$. The protocol step composition is the same as program composition. Therefore, the (COMP) inference rule is equivalent to the weakest precondition of program composition. The (COMM) contains a condition to extract the transmitted information and a frame substitution function. If the condition is satisfied, then the frame substitution function is applied to the information and the result is stored in the variable y located at the receiver. Otherwise, the empty information is stored in y.

In the analysis, we are taking the initial knowledge of agents into consideration which plays an important role as shown by the following propositions.

Proposition 4.4. *let $\mathcal{N} = (\{e_\emptyset\}, D)$ be an empty knowledge. We have:*

1. $update(\mathcal{N},\ \varphi,\ e_\emptyset) = \mathcal{N}$
2. $insert(\mathcal{N}, e_\emptyset) = \mathcal{N}$
3. $choose(extract(\mathcal{N},\ x,\ \varphi)) = e_\emptyset$

Proof. The deialed proof can be found in Appendix E.

1. The proof applies the definition of update, Proposition 4.1(3), singleton membership, Proposition 4.1(3), set union axiom and properities of propositional logic.
2. The proofs applies the definition of insert and the idempotency of $\cup$.
3. The extract function would either return an empty set or a set with one element $e_\emptyset$. In both cases, *choose* will return $e_\emptyset$.

□

As a consequence result of Proposition 4.4, we can prove the following proposition.

Proposition 4.5. *Let $\mathcal{N}^A = (\{e_\emptyset\}, D^A)$ and $\mathcal{N}^B = (\{e_\emptyset\}, D^B)$ be the knowledges of two agents communicating through a protocol S. We have $wp(S, Q) \Leftrightarrow Q$.*

Proof. The postcondition Q is on the knowledge of agents. According to Proposition 4.4, the knowledge of agents with empty knowledge does not change through the insert and update functions. □

Proposition 4.5 states that if the initial knowledge of agents A and B contain only the empty information, then from an information flow prospective, protocol specification is equivalent to skip. On the other side, if the initial knowledge of agents A and B contain all possible information i.e., $\Phi = \{\varphi \mid \varphi \in D_J \text{ for } J \subseteq I\}$, then we can prove the following proposition.

Proposition 4.6. *let $\mathcal{N} = (\Phi, D)$ where $\Phi = \{\varphi \mid \varphi \in D_J \text{ for } J \subseteq I\}$. We have $insert(\mathcal{N}, \varphi) = \mathcal{N}$*

Proof. The proofs uses the definition of insert and the fact that $\varphi \in \Phi$ □

Proposition 4.7. *Let* $\mathcal{N}^A = (\Phi, D^A)$ *and* $\mathcal{N}^B = (\Phi, D^B)$ *where* $\Phi = \{\varphi \mid \varphi \in D_J \text{ for } J \subseteq I\}$ *be the knowledge of two agents communicating through a protocol* S*. Also, assume that* S *does not contain an update. We have* $wp(S, Q) \Leftrightarrow Q$*.*

Proof. The postcondition Q is on the knowledge of agents. According to Proposition 4.6, the knowledge of agents does not change through the insert function. □

We exclude the update function from Proposition 4.7 because it does not preserve the knowledge. The update function removes and combines a piece of information to the information already exsists in the knowledge of an agent.

For the given example, we can verify several policies such as (1) the AA should not know pieces of information classified as employee in the CA knowledge (2) the PR should not know a piece of information classified as company in the CA knowledge (3) the PR should not know the mission of an officer ID.

To analyze a policy, we specify the communication between agents S, the precondition P, and the postcondition Q. Then, we prove that $p \Rightarrow wp(S, Q)$. We specify the communication between agents by using global calculus and information algebra as shown in Section 4.2. The precondition is a predicate on the initial knowledge of agents. We represent the initial knowledge of agents as a conjunction of formulae of the form $isInKnowledge(\mathcal{N},\ x,\ \varphi)$. For example, to indicate that the initial knowledge of the coordinator contain the information $\varphi = \{(OfficierID,\ \{JohnDo\}),\ (mission,\ \{Cobra\}),\ (topic,\ \{Economy\})\}$, we use the predicate $isInKnowledge(\mathcal{N}^C,\ d(\varphi),\ \varphi)$. The postcondition, which is a predicate on the knowledge of agents, is the negation of a policy.The postcondition of the policies that are stated above are:

The first policy is *the Analyst Agent (AA) should not know a pieces of information classified as employee in the Coordinator Agent's (CA) knowledge.* The postcondition is specified as

$$\begin{aligned}&\exists(\varphi, x \mid x \in D^{AA} \wedge \varphi \in D^{CA}_{\{employee\}} : \varphi \neq e_{\{employee\}} \wedge \\ &\quad isInKnowledge(\mathcal{N}^{CA},\ D_{\{employee\}},\ \varphi) \wedge \\ &\quad isInKnowledge(\mathcal{N}^{PR},\ x,\ fs(\varphi, D_{\{employee\}}, x)))\end{aligned}$$

The second policy is that *the Public Relation Agent (PR) should not know a piece of information classified as company in the Coordinator Agent's (CA) knowledge.* The postcondition is specified as

$$\begin{aligned}&\exists(\varphi, x \mid x \in D^{PR} \wedge \varphi \in D^{CA}_{\{company\}} : \varphi \neq e_{\{company\}} \wedge \\ &\quad isInKnowledge(\mathcal{N}^{CA},\ D_{\{company\}},\ \varphi) \wedge \\ &\quad isInKnowledge(\mathcal{N}^{PR},\ x,\ fs(\varphi, D_{\{company\}}, x)))\end{aligned}$$

The third policy is that *the Public Relation Agent (PR) should not know the mission of an officer OO.*

$$\exists(\varphi, x, y \mid x, y \in D^{PR} \wedge \varphi \in D^{CA}_{\{officerID,\ mission\}} : \varphi \neq e_{\{officerID,\ mission\}} \wedge isInKnowledge(\mathcal{N}^{CA},\ D_{\{officerId,\ mission\}},\ \varphi) \wedge isInKnowledge(\mathcal{N}^{PR},\ x \curlyvee y,\ fs(\varphi^{\downarrow D_{\{officerID\}}}, D_{\{officerID\}}, x) \cdot fs(\varphi^{\downarrow D_{\{mission\}}}, D_{\{mission\}}, y)))$$

We developed a prototype tool which finds the weakest precondition from S and Q. Then, the tool writes it using the syntax of PVS language. After that, we use PVS to prove that $P \Rightarrow wp(S, Q)$. Currently, we are proving theorem in PVS through the interactive mode. However, we intend to automate this step. The prove is based on the propositions that we define in this report. We are able to prove that $P \Rightarrow wp(S, Q)$ for the second policy which means that it contains a flaw. Indeed, the policy is not satisfied because PR gets an information from AA that is classified as company in the CA knowledge. More details on the proof of this policy can be found in Appendix F. We are able to prove $P \Rightarrow wp(S, Q)$ for the third policy also. The third policy is not satisfied because PR get a mission from CA and the officier ID of that mission from AA. If we are unable to prove $P \Rightarrow wp(S, Q)$ for a policy, this does not mean that the policy is satisfied. To prove a policy is satisfied we should prove $\neg(P \Rightarrow wp(S, Q))$. We prove that the first policy is not satisfied.

Manual analysis of the system by a skillful analyst might lead to the detection of these flaws. However, assume that the system consists of hundreds of agents instead of four. In this case, the manual analysis of the system would be resource consuming and subject to errors especially that all interleaving steps obtained from the concurrent communication between agents should be taken into consideration in the analysis.

5 Discussion and Related Work

In this section, we compare the proposed framework with Bell-LaPadula and Chinese Wall models. Then we present the aspects that distinguish the proposed framework from the existing techniques.

5.1 Bell-LaPadula model

We compare Bell-LaPadula model [2] with the proposed technique regarding the following aspects (1) *Security Level*: Bell-LaPadula model assign security levels to subjects and objects. The security level forms a lattice which enables to have a relation as: *private* is more sensitive than *public*. In the proposed technique, we use a lattice of frames to classify information. Our lattice enables to perform analysis on a composite information. We can give security level to information or agents through policies (2) *Objects*: In Bell-LaPadula model objects cannot be decomposed or combined. While in the proposed technique they can be combined (i.e., combining information) (3) *Rules*: Bell-LaPadula model defines

rules for reading and writing into objects. These rules can be specified within the proposed technique as the condition in the communication step.

5.2 The Chinese Wall model

The Chinese Wall [3] is a security model which has three levels of significance. The lowest level contains pieces of information. Pieces of information of the same cooperation are grouped into a company dataset. Company datasets whose cooperation are in competition are grouped into a conflict of interest class. An agent should have access only to information of one company dataset of the conflict of interest class.

The Chinese Wall can be represented within the proposed technique as follows. The pieces of information of Chinese Wall corresponds to the pieces of information within the proposed technique while the company datasets corresponds to frames. We do not have a direct representation of the conflict of interest class. However, this is a policy which can be stated within the proposed technique in different ways. One way is to state it as the condition in the communication step i.e., an agent can receive an information only if the knowledge of that agent does not contain information belonging to frames which are in conflict with the received information frame.

5.3 Verification techniques

Bell-LaPadula and Chinese Wall are two models that specify policies for a secure information flow between agents. In the literature, there exists several techniques [1, 6, 9, 14] to verify that the information flow between agents satisfy predefined policies. The proposed technique has the following aspects which distinguish our work from others.

Analyzing a composite information flow: An advantage of the proposed technique over the existing ones is analyzing composite information flow. The technique can analyze the ability of an agent to link pieces of information together from different resources. Existing techniques such as SPA [6] verifies only if an agent is able to get an information that it is not supposed to have.

Specifying agent knowledge: Agent knowledge representation adopted in this paper enables specifying the evolution of knowledge through communication. This specification is different from using variables that their new value substitute the old one (i.e., there is no evolution). Also, each agent has its own lattice of frames that classifies the information it holds. We remove the restriction that all pieces of information should have the same classification in agents' knowledge. Removing this restriction provides flexibility in specifying heterogeneous subsystems.

Analyzing distributed systems: The technique can specify distributed systems. By using global calculus, we are able to compose communication steps either concurrently or sequentially.

Analyzing system at design level: The technique enables the analysis of systems at design level. Analyzing systems at an early stage of the software development life cycle and an early detection of a system security vulnerability would reduce the cost of addressing

it. This stage of analysis is different from typing systems [9, 10, 15] that have been widely used for analyzing information flow at the implementation level.

Information level analysis: We conduct analysis at the information level which enables to cover additional flaws. For example, having the same information with two different classification in the knowledge of an agent may breach a policy.

6 Conclusion

In this paper, we propose a technique for the analysis of information flow between agents in multi-agent systems. We develop an algebraic structure to specify the knowledge of agents based on information algebra. We also link global calculus with information algebra and use them to specify communication protocols. Furthermore, we use an amended version of Hoare logic to analyze the information flow between agents. We report on a prototype tool to derive the weakest precondition of a communication protocol. The tool verifies if the initial knowledge of agents implies the weakest precondition by using the PVS theorem prover.

The proposed technique provides a comprehensive language to specify agents knowledge and their communication. For example, information algebra allows reasoning on a composite information as in the third policy of the given example. The global calculus allows specifying a composition of protocols using the sequential and parallel operators. Also, the analysis takes the initial knowledge of agents into consideration which affects the satisfability of a policy. Finally, performing the analysis before implementing the system allows detecting flows at an early stage. The implementation can be generated automatically from the specification which can be a future work. However, the trade off for having a detailed analysis is increasing its complexity.

A Related Propositions

We find the following axioms and propositions in [12] which we use in this report.

Axioms A.1.

1. $\varphi \in D_J \Rightarrow d(\varphi) = D_J$
2. $e_J \triangleq \{(i, \emptyset) \mid i \in J\}$

□

Proposition A.1. *For $J, K \subseteq I$ and $\varphi \in \Phi$*

1. $\varphi \in D_K \Rightarrow \mathcal{I}_J ; \varphi = \{(i, A) \mid i \in (J \cap K) \wedge A \subseteq \mathbb{A}_i\}$
2. $\mathcal{I}_J ; \mathcal{I}_K = \mathcal{I}_{J \cap K}$
3. $\varphi \in D_K \Rightarrow d(\mathcal{I}_J ; \varphi) = D_{J \cap K}$

Proposition A.2.

1. $\forall (J, K \mid J, K \subseteq I : J = K \Rightarrow D_J = D_K)$
2. $\forall (J, K \mid J, K \subseteq I : D_J \preccurlyeq D_K \Rightarrow J \subseteq K)$

Proposition A.3.

1. $d(\varphi) = D_J \Rightarrow \varphi^{\downarrow D_J} = \varphi$
2. $\varphi\varphi = \varphi$
3. $D_J \preccurlyeq D_K \Rightarrow e_J e_K = e_K$
4. $d(\varphi) = D_J \Rightarrow (\varphi e_K)^{\downarrow D_J} = \varphi$
5. $D_J \preccurlyeq d(\varphi) \Rightarrow (\varphi e_K)^{\downarrow D_J} = \varphi^{\downarrow D_J}$
6. $e_{J \curlyvee K} = e_J e_K$

Also, we prove the following propositions.

Proposition A.4. *For $\varphi \in D_J$, we have $\varphi^{\downarrow D_\emptyset} = e_\emptyset$*

Proof.

$$
\begin{array}{ll}
 & \varphi^{\downarrow D_\emptyset} \\
= & \qquad \langle\ \varphi \in D_J\ \rangle \\
 & \{(i,A)\ \mid\ i \in J\ \wedge\ A = \varphi(i)\}^{\downarrow D_\emptyset} \\
= & \qquad \langle\ \text{Definition of } \downarrow_{D_\emptyset}\ \rangle \\
 & \mathcal{I}_\emptyset;\{(i,A)\ \mid\ i \in J\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Proposition A.1(1)}\ \rangle \\
 & \{(i,A)\ \mid\ i \in J \cap \emptyset\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ J \cap \emptyset = \emptyset\ \rangle \\
 & \{(i,A)\ \mid\ i \in \emptyset\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ i \in \emptyset \Leftrightarrow \mathsf{false}\ \rangle \\
 & \{(i,A)\ \mid\ \mathsf{false}\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Zero of}\ \wedge\ \rangle \\
 & \{(i,A)\ \mid\ \mathsf{false}\} \\
= & \qquad \langle\ \text{Zero of}\ \wedge\ \rangle \\
 & \{(i,A)\ \mid\ \mathsf{false}\ \wedge\ A = \emptyset\} \\
= & \qquad \langle\ i \in \emptyset \Leftrightarrow \mathsf{false}\ \rangle \\
 & \{(i,A)\ \mid\ i \in J\ \wedge\ A = \emptyset\} \\
= & \qquad \langle\ \text{Definition of } e_\emptyset\ \rangle \\
 & e_\emptyset
\end{array}
$$

□

Proposition A.5. *For $J \subseteq I$, we have $e_\emptyset^{\downarrow D_J} = e_\emptyset$*

Proof.

$$
\begin{array}{ll}
 & e_\emptyset^{\downarrow D_J} \\
= & \qquad \langle\ \text{Definition of } e_\emptyset\ \rangle \\
 & \{(i,\emptyset)\ \mid\ i \in \emptyset\}^{\downarrow D_J} \\
= & \qquad \langle\ \text{Definition of } \downarrow_{D_J}\ \rangle \\
 & \mathcal{I}_J;\{(i,\emptyset)\ \mid\ i \in \emptyset\} \\
= & \qquad \langle\ \text{Proposition A.1(1)}\ \rangle \\
 & \{(i,\emptyset)\ \mid\ i \in \emptyset \cap J\} \\
= & \qquad \langle\ J \cap \emptyset = \emptyset\ \rangle \\
 & \{(i,\emptyset)\ \mid\ i \in \emptyset\} \\
= & \qquad \langle\ \text{Definition of } e_\emptyset\ \rangle
\end{array}
$$

$e_\emptyset$

□

Proposition A.6. *For $\varphi \in D_J$, we have $\varphi e_\emptyset = \varphi$*

Proof.

$$
\begin{array}{ll}
& \varphi e_\emptyset \\
= & \qquad \langle \text{ Definition of } \cdot \ \rangle \\
& \quad \{(i,A) \mid i \in J \cap \emptyset \ \wedge \ A = \varphi(i) \cup \emptyset\} \ \cup \ \{(i,A) \mid i \in J - \emptyset \ \wedge \ A = \varphi(i)\} \\
& \cup \ \{(i,A) \mid i \in \emptyset - J \ \wedge \ A = \emptyset\} \\
= & \qquad \langle \text{ Set theory } \rangle \\
& \quad \{(i,A) \mid i \in \emptyset \ \wedge \ A = \varphi(i) \cup \emptyset\} \ \cup \ \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \\
& \cup \ \{(i,A) \mid i \in \emptyset \ \wedge \ A = \emptyset\} \\
= & \qquad \langle \ i \in \emptyset \Leftrightarrow \mathsf{false} \ \rangle \\
& \quad \{(i,A) \mid \mathsf{false} \ \wedge \ A = \varphi(i) \cup \emptyset\} \ \cup \ \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \\
& \cup \ \{(i,A) \mid \mathsf{false} \ \wedge \ A = \emptyset\} \\
= & \qquad \langle \text{ Zero of } \wedge \ \rangle \\
& \{(i,A) \mid \mathsf{false}\} \ \cup \ \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \ \cup \ \{(i,A) \mid \mathsf{false}\} \\
= & \qquad \langle \text{ Empty range axiom } \rangle \\
& \emptyset \ \cup \ \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \ \cup \ \emptyset \\
= & \qquad \langle \ S \cup \emptyset = S \ \rangle \\
& \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \\
= & \qquad \langle \ \varphi \in D_J \ \rangle \\
& \varphi
\end{array}
$$

□

Proposition A.7. $e_\emptyset = \emptyset$

Proof.

$$
\begin{array}{ll}
& \{(i,\emptyset) \mid i \in \emptyset\} \\
\Leftrightarrow & \qquad \langle \text{ Empty set (i.e., } i \in \emptyset \Leftrightarrow \mathsf{false}) \ \rangle \\
& \{(i,\emptyset) \mid \mathsf{false}\} \\
\Leftrightarrow & \qquad \langle \text{ Empty range } \rangle \\
& \emptyset
\end{array}
$$

□

Proposition A.8. *For $\varphi \in D_L$, we have $\varphi^{\downarrow D_J} \cdot \varphi^{\downarrow D_K} = \varphi^{\downarrow D_{J \cup K}}$*

Proof.

$$
\begin{array}{ll}
& \varphi^{\downarrow D_J} \cdot \varphi^{\downarrow D_K} \\
= & \qquad \langle \text{ Definition of } \downarrow_{D_J} \text{ and } \downarrow_{D_K} \rangle \\
& \mathcal{I}_J{;}\varphi \cdot \mathcal{I}_K{;}\varphi \\
= & \qquad \langle\ \varphi \in D_L\ \rangle \\
& \mathcal{I}_J{;}\{(i,A) \mid i \in L \ \wedge\ A = \varphi(i)\} \cdot \mathcal{I}_K{;}\{(i,A) \mid i \in L \ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle \text{ Proposition A.1(1) } \rangle \\
& \{(i,A) \mid i \in L \cap J \ \wedge\ A = \varphi(i)\} \cdot \{(i,A) \mid i \in L \cap K \ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle \text{ Definition of } \cdot \rangle \\
& \quad \{(i,A) \mid i \in (L \cap J) \cap (L \cap K) \ \wedge\ A = \varphi(i) \cup \varphi(i)\} \\
& \cup\ \{(i,A) \mid i \in (L \cap J) - (L \cap K) \ \wedge\ A = \varphi(i)\} \\
& \cup\ \{(i,A) \mid i \in (L \cap K) - (L \cap J) \ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle \cup \text{ is idempotent } \rangle \\
& \quad \{(i,A) \mid i \in (L \cap J) \cap (L \cap K) \ \wedge\ A = \varphi(i)\} \\
& \cup\ \{(i,A) \mid i \in (L \cap J) - (L \cap K) \ \wedge\ A = \varphi(i)\} \\
& \cup\ \{(i,A) \mid i \in (L \cap K) - (L \cap J) \ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle \text{ Range split (i.e., } \{x \mid r\} \cup \{x \mid p\} = \{x \mid r \vee p\}) \rangle \\
& \{(i,A) \mid i \in (L \cap J) \cap (L \cap K) \ \wedge\ A = \varphi(i) \\
& \ \vee\ i \in (L \cap J) - (L \cap K) \ \wedge\ A = \varphi(i) \\
& \ \vee\ i \in (L \cap K) - (L \cap J) \ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle \text{ Distributivity of } \wedge \text{ over } \vee \rangle \\
& \{(i,A) \mid (i \in (L \cap J) \cap (L \cap K) \ \vee\ i \in (L \cap J) - (L \cap K) \ \vee\ i \in (L \cap K) - (L \cap J)) \\
& \ \wedge\ \ A = \varphi(i)\} \\
= & \qquad \langle \text{ Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \rangle \\
& \{(i,A) \mid i \in ((L \cap J) \cap (L \cap K)) \ \cup\ ((L \cap J) - (L \cap K)) \ \cup\ ((L \cap K) - (L \cap J)) \\
& \ \wedge\ \ A = \varphi(i)\} \\
= & \qquad \langle \text{ Set theory } \rangle \\
& \{(i,A) \mid i \in (L \cap (K \cup J)) \ \ \wedge\ \ A = \varphi(i)\} \\
= & \qquad \langle \text{ Proposition A.1(1) } \rangle \\
& \mathcal{I}_{J \cup K}{;}\{(i,A) \mid i \in L \ \ \wedge\ \ A = \varphi(i)\} \\
= & \qquad \langle\ \varphi \in D_L\ \rangle \\
& \mathcal{I}_{J \cup K}{;}\varphi \\
= & \qquad \langle \text{ Definition of } \downarrow_{D_{J \cup K}} \rangle
\end{array}
$$

$\varphi^{D_{J\cup K}}$

□

Proposition A.9. *For $\varphi \in D_J$ and $\psi \in D_K$, we have $(\varphi\psi)^{\downarrow D_L} = \varphi^{\downarrow D_L}\psi^{\downarrow D_L}$*

Proof.

$$
\begin{array}{ll}
 & (\varphi\psi)^{\downarrow D_L} \\
= & \qquad \langle \text{ Definition of } \cdot \ \rangle \\
 & \quad (\{(i,A) \mid i \in J \cap K \ \wedge \ A = \varphi(i) \cup \psi(i)\} \\
 & \cup \ \{(i,A) \mid i \in J - K \ \wedge \ A = \varphi(i)\} \\
 & \cup \ \{(i,A) \mid i \in K - J \ \wedge \ A = \psi(i)\})^{\downarrow D_L} \\
= & \qquad \langle \text{ Definition of } \downarrow_{D_L} \ \rangle \\
 & \quad \mathcal{I}_L;\{(i,A) \mid i \in J \cap K \ \wedge \ A = \varphi(i) \cup \psi(i)\} \\
 & \cup \ \mathcal{I}_L;\{(i,A) \mid i \in J - K \ \wedge \ A = \varphi(i)\} \\
 & \cup \ \mathcal{I}_L;\{(i,A) \mid i \in K - J \ \wedge \ A = \psi(i)\} \\
= & \qquad \langle \text{ Proposition A.1(1) } \rangle \\
 & \quad \{(i,A) \mid i \in J \cap K \cap L \ \wedge \ A = \varphi(i) \cup \psi(i)\} \\
 & \cup \ \{(i,A) \mid i \in (J - K) \cap L \ \wedge \ A = \varphi(i)\} \\
 & \cup \ \{(i,A) \mid i \in (K - J) \cap L \ \wedge \ A = \psi(i)\} \\
= & \qquad \langle \text{ Set theory } \rangle \\
 & \quad \{(i,A) \mid i \in (J \cap L) \cap (K \cap L) \ \wedge \ A = \varphi(i) \cup \psi(i)\} \\
 & \cup \ \{(i,A) \mid i \in (J \cap L) - (K \cap L) \ \wedge \ A = \varphi(i)\} \\
 & \cup \ \{(i,A) \mid i \in (K \cap L) - (J \cap L) \ \wedge \ A = \psi(i)\} \\
= & \qquad \langle \text{ Definition of } \cdot \text{ and } \downarrow_{D_L} \ \rangle \\
 & \varphi^{\downarrow D_L}\psi^{\downarrow D_L}
\end{array}
$$

□

Proposition A.10. *For $\varphi \in D_J$, we have $\varphi^{\downarrow D_K} = \varphi^{\downarrow D_{J\cap K}}$*

Proof.

$$
\begin{array}{ll}
 & \varphi^{\downarrow D_K} \\
= & \qquad \langle \ \varphi \in D_J \ \rangle \\
 & \{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\}^{\downarrow D_K} \\
= & \qquad \langle \text{ Definition of } \downarrow_{D_K} \ \rangle \\
 & \mathcal{I}_K;\{(i,A) \mid i \in J \ \wedge \ A = \varphi(i)\} \\
= & \qquad \langle \text{ Proposition A.1(1) } \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & \{(i, A) \mid i \in J \cap K \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \text{ Set theory } \rangle \\
 & \{(i, A) \mid i \in J \cap (J \cap K) \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \text{ Definition of } \downarrow_{D_K} \rangle \\
 & \varphi^{\downarrow D_{J \cap K}}
\end{array}
$$

□

Proposition A.11. *For $\varphi \in D_J$, we have $(\varphi^{\downarrow D_K})^{\downarrow D_L} = \varphi^{\downarrow D_{K \cap L}}$*

Proof.

$$
\begin{array}{ll}
 & (\varphi^{\downarrow D_K})^{\downarrow D_L} \\
= & \quad \langle \text{ Definition of } \downarrow_{D_L} \rangle \\
 & \mathcal{I}_L ; (\varphi^{\downarrow D_K}) \\
= & \quad \langle \text{ Definition of } \downarrow_{D_K} \rangle \\
 & \mathcal{I}_K ; \mathcal{I}_L ; \varphi \\
= & \quad \langle \text{ Proposition 1(2) } \rangle \\
 & \mathcal{I}_{K \cap L} ; \varphi \\
= & \quad \langle \text{ Definition of } \downarrow_{D_{K \cap L}} \rangle \\
 & \varphi^{\downarrow D_{K \cap L}}
\end{array}
$$

□

Proposition A.12. *For $J = \{j\}$, we have $\neg(J \subseteq K) \Rightarrow J \cap K = \emptyset$*

Proof. Our proof strategy for $p \Rightarrow q$ is to assume p and then prove q. We assume

$$
\begin{array}{ll}
 & \neg(J \subseteq K) \\
\Leftrightarrow & \quad \langle \text{ Subset axiom } \rangle \\
 & \neg \, \forall(i \mid i \in J \, : \, i \in K \,) \\
\Leftrightarrow & \quad \langle \text{ Singleton membership } \rangle \\
 & \neg \, \forall(i \mid i = j \, : \, i \in K \,) \\
\Leftrightarrow & \quad \langle \text{ Generalized De Morgan } \rangle \\
 & \exists(i \mid i = j \, : \, \neg(i \in K) \,) \\
\Leftrightarrow & \quad \langle \text{ Trading rule for } \exists \, \rangle \\
 & \exists(i \mid : \, i = j \ \wedge \ \neg(i \in K) \,) \\
\Leftrightarrow & \quad \langle \text{ Substitution axiom } \rangle \\
 & \exists(i \mid : \, i = j \ \wedge \ \neg(j \in K) \,)
\end{array}
$$

$\Leftrightarrow$ $\langle$ Distributivity of $\wedge$ over $\exists$ $\rangle$
$\neg(j \in K) \ \wedge \ \exists(i \ |: \ i = j)$
$\Rightarrow$ $\langle$ Weakening $\rangle$
$\neg(j \in K)$

Then, we prove $J \cap K = \emptyset$

$J \cap K$
$=$ $\langle$ Set intersection axiom (i.e., $i \in A \ \wedge \ i \in B \Leftrightarrow i \in A \cap B$) $\rangle$
$\{i \mid i \in J \ \wedge \ i \in K\}$
$=$ $\langle$ Singleton membership $\rangle$
$\{i \mid i = j \ \wedge \ i \in K\}$
$=$ $\langle$ Substitution axiom $\rangle$
$\{i \mid i = j \ \wedge \ j \in K\}$
$=$ $\langle$ Assumption $\neg(j \in K)$ $\rangle$
$\{i \mid i \in j \ \wedge \ \mathsf{false}\}$
$=$ $\langle$ Zero of $\wedge$ $\rangle$
$\{i \mid \mathsf{false}\}$
$=$ $\langle$ Empty range $\rangle$
$\emptyset$

$\square$

B The Detailed Proof of Proposition 4.1

Proposition 4.1(1)

For $d(\varphi) = D_J$ and $d(\psi) = D_K$, we have $d(\varphi - \psi) = d(\varphi)$.

Proof.

$$
\begin{array}{ll}
 & d(\varphi - \psi) = d(\varphi) \\
\Leftarrow & \quad \langle\ d(\varphi) = D_J\ \rangle \\
 & d(\varphi - \psi) = D_J \\
\Leftarrow & \quad \langle\ \text{Axiom A.1(1)}\ \rangle \\
 & \varphi - \psi \in D_J \\
\Leftarrow & \quad \langle\ \text{Definition of } D_J\ \rangle \\
 & \exists(f \mid f \in D_I : \varphi - \psi = \mathcal{I}_J{;}f\,) \\
\Leftarrow & \quad \langle\ \text{Let} \\
 & \qquad\qquad f(i) = \begin{cases} \varphi(i) - \psi(i) & \text{if } i = J \cap K \\ \varphi(i) & \text{if } i = J - K \\ \emptyset & \text{otherwise} \end{cases} \\
 & \quad \&\ \exists\text{-Split-off-term}\ \rangle \\
 & \exists(f \mid f \in D_I : \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \mathcal{I}_J{;}(\{(i,A) \mid i \in J \cap K \ \wedge\ A = \varphi(i) - \psi(i)\} \\
 & \quad \cup\ \{(i,A) \mid i \in J - K \ \wedge\ A = \varphi(i)\} \\
 & \quad \cup\ \{(i,\emptyset) \mid i \in \overline{J}\}) \\
\Leftarrow & \quad \langle\ \text{Distributivity of } ; \text{ over } \cup\ \rangle \\
 & \exists(f \mid f \in D_I : \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \mathcal{I}_J{;}\{(i,A) \mid i \in J \cap K \ \wedge\ A = \varphi(i) - \psi(i)\} \\
 & \quad \cup\ \mathcal{I}_J{;}\{(i,A) \mid i \in J - K \ \wedge\ A = \varphi(i)\} \\
 & \quad \cup\ \mathcal{I}_J{;}\{(i,\emptyset) \mid i \in \overline{J}\} \\
\Leftarrow & \quad \langle\ \text{Proposition A.1(1)}\ \rangle \\
 & \exists(f \mid f \in D_I : \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \{(i,A) \mid i \in (J \cap K) \cap J \ \wedge\ A = \varphi(i) - \psi(i)\} \\
 & \quad \cup\ \{(i,A) \mid i \in (J - K) \cap J \ \wedge\ A = \varphi(i)\}) \\
 & \quad \cup\ \{(i,\emptyset) \mid i \in \overline{J} \cap J\} \\
\Leftarrow & \quad \langle\ \text{Set theory}\ \rangle \\
 & \exists(f \mid f \in D_I : \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \{(i,A) \mid i \in J \cap K \ \wedge\ A = \varphi(i) - \psi(i)\} \\
 & \quad \cup\ \{(i,A) \mid i \in J - K \ \wedge\ A = \varphi(i)\})
\end{array}
$$

$$
\begin{array}{ll}
 & \quad \cup\ \{(i,\emptyset)\ \mid\ i \in \emptyset\} \\
\Leftarrow & \qquad \langle\ \text{Empty range axiom}\ \rangle \\
 & \exists(f\ \mid\ f \in D_I\ :\ \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \{(i,A)\ \mid\ i \in J \cap K\ \wedge\ A = \varphi(i) - \psi(i)\} \\
 & \quad \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\} \\
 & \quad \cup\ \emptyset \\
\Leftarrow & \qquad \langle\ \text{Identity of}\ \cup\ \rangle \\
 & \exists(f\ \mid\ f \in D_I\ :\ \varphi\psi = \mathcal{I}_J{;}f\,) \\
 & \vee\ \ \varphi - \psi = \{(i,A)\ \mid\ i \in J \cap K\ \wedge\ A = \varphi(i) \cup \psi(i)\} \\
 & \quad \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\} \\
\Leftarrow & \qquad \langle\ \text{Definition of}\ \varphi - \psi\ \rangle \\
 & \exists(f\ \mid\ f \in D_I\ :\ \varphi\psi = \mathcal{I}_J{;}f\,) \vee \mathsf{true} \\
\Leftarrow & \qquad \langle\ \text{Zero of}\ \vee\ \rangle \\
 & \mathsf{true}
\end{array}
$$

□

Proposition 4.1(2)

For $d(\varphi) = D_J$, we have $\varphi - e_K = \varphi$

Proof.

$$
\begin{array}{ll}
 & \varphi - e_K \\
= & \qquad \langle\ \text{Definition of}\ \varphi - e_K\ \rangle \\
 & \{(i,A)\ \mid\ i \in J \cap K\ \wedge\ A = (\varphi(i) - e_K(i))\}\ \ \cup\ \ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Definition of}\ e_K\ (e_K(i) = \emptyset)\ \rangle \\
 & \{(i,A)\ \mid\ i \in J \cap K\ \wedge\ A = (\varphi(i) - \emptyset)\}\ \ \cup\ \ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Properties of set difference}\ \rangle \\
 & \{(i,A)\ \mid\ i \in J \cap K\ \wedge\ A = \varphi(i)\}\ \ \cup\ \ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B)\ \rangle \\
 & \{(i,A)\ \mid\ (i \in J \cap K\ \wedge\ A = \varphi(i)) \vee (i \in J - K\ \wedge\ A = \varphi(i))\} \\
= & \qquad \langle\ \text{Distributivity of}\ \wedge\ \text{over}\ \vee\ \rangle \\
 & \{(i,A)\ \mid\ (i \in J \cap K \vee i \in J - K)\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B)\ \rangle \\
 & \{(i,A)\ \mid\ i \in (J \cap K) \cup (J - K)\ \wedge\ A = \varphi(i)\} \\
= & \qquad \langle\ \text{Set theory}\ \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & \{(i, A) \mid i \in J \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \text{ Rewriting } \rangle \\
 & \{(i, \varphi(i)) \mid i \in J\} \\
= & \quad \langle \ \varphi \in D_J \ \rangle \\
 & \varphi
\end{array}
$$

□

Proposition 4.1(3)

For $d(\varphi) = D_K$, we have $e_J - \varphi = e_J$

Proof.

$$
\begin{array}{ll}
 & e_J - \varphi \\
= & \quad \langle \text{ Definition of } e_J - \varphi \ \rangle \\
 & \{(i, A) \mid i \in J \cap K \ \wedge \ A = (e_J - \varphi(i))\} \ \cup \ \{(i, A) \mid i \in J - K \ \wedge \ A = e_J\} \\
= & \quad \langle \text{ Definition of } e_K \ (e_K(i) = \emptyset) \ \rangle \\
 & \{(i, A) \mid i \in J \cap K \ \wedge \ A = (\emptyset - \varphi(i))\} \ \cup \ \{(i, A) \mid i \in J - K \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Set theory } \rangle \\
 & \{(i, A) \mid i \in J \cap K \ \wedge \ A = \emptyset\} \ \cup \ \{(i, A) \mid i \in J - K \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \ \rangle \\
 & \{(i, A) \mid (i \in J \cap K \ \wedge \ A = \emptyset) \vee (i \in J - K \ \wedge \ A = \emptyset)\} \\
= & \quad \langle \text{ Distributivity of } \wedge \text{ over } \vee \ \rangle \\
 & \{(i, A) \mid (i \in J \cap K \vee i \in J - K) \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \ \rangle \\
 & \{(i, A) \mid i \in (J \cap K) \cup (J - K) \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Set theory } \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Renaming } \rangle \\
 & \{(i, \emptyset) \mid i \in J\} \\
= & \quad \langle \text{ Definition of } e_J \ \rangle \\
 & e_J
\end{array}
$$

□

Proposition 4.1(4)

Let $d(\varphi) = D_J$, $d(\psi) = D_K$, and $d(\chi) = D_L$ for $J, K, L \subseteq I$, we have
$\varphi \leq (\psi - \chi) \Rightarrow \varphi \leq \psi$

Proof.

$$
\begin{array}{ll}
& \varphi \leq (\psi - \chi) \\
\Leftrightarrow & \quad \langle \text{ Definition of } \leq \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle\ p \ \wedge\ \mathsf{true} \Leftrightarrow p\ \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \ \wedge\ \mathsf{true} : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle\ i \in I \Leftrightarrow \mathsf{true}\ \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \ \wedge\ i \in I : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle\ (K \cap L) \cup (K - L) \cup \overline{K} = I\ \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \ \wedge\ i \in (K \cap L) \cup (K - L) \cup \overline{K} : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle \text{ Set Union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \ \wedge\ (i \in K \cap L \vee i \in K - L \vee i \in \overline{K}) : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle \text{ Distributivity of } \wedge \text{ over } \vee \rangle \\
& J \subseteq K \ \wedge \\
& \forall(i \mid (i \in J \ \wedge\ i \in K \cap L) \vee (i \in J \ \wedge\ i \in K - L) \vee (i \in J \ \wedge\ i \in \overline{K}) : \\
& \quad \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle \text{ Set Intersection axiom (i.e., } i \in A \ \wedge\ i \in B \Leftrightarrow i \in A \cap B) \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \cap (K \cap L) \vee i \in J \cap (K - L) \vee i \in J \cap \overline{K} : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle \text{ Range split for } \forall \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \cap (K \cap L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid i \in J \cap (K - L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid i \in J \cap \overline{K} : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle\ J \subseteq K \Leftrightarrow J \cap \overline{K} = \emptyset\ \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \cap (K \cap L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid i \in J \cap (K - L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid i \in \emptyset : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle\ i \in \emptyset \Leftrightarrow \mathsf{false}\ \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \cap (K \cap L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid i \in J \cap (K - L) : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge \\
& \forall(i \mid \mathsf{false} : \varphi(i) \subseteq (\psi - \chi)(i)) \\
\Leftrightarrow & \quad \langle \text{ Empty range axiom } \rangle \\
& J \subseteq K \ \wedge \ \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq (\psi - \chi)(i)) \ \wedge
\end{array}
$$

$\forall(i \mid i \in (J \cap (K - L) : \varphi(i) \subseteq (\psi - \chi)(i)) \wedge$
true
$\Leftrightarrow$ $\langle\ p \wedge \mathsf{true} \Leftrightarrow \mathsf{true}\ \rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap (K \cap L) : \varphi(i) \subseteq (\psi - \chi)(i)) \wedge$
$\forall(i \mid i \in (J \cap (K - L) : \varphi(i) \subseteq (\psi - \chi)(i))$
$\Leftrightarrow$ $\langle$ Definition of $\varphi - \chi$ $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq (\psi(i) - \chi(i))) \wedge$
$\forall(i \mid i \in (J \cap (K - L)) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Definition of set difference $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq (\psi(i) \cap \overline{\chi(i)})) \wedge$
$\forall(i \mid i \in (J \cap (K - L)) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Set theory $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq \psi(i) \wedge \varphi \subseteq \overline{(\chi(i))}) \wedge$
$\forall(i \mid i \in (J \cap (K - L)) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Destributivity axiom $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq \psi(i)) \wedge$
$\forall(i \mid i \in J \cap K \cap L : \varphi \subseteq \overline{(\chi(i))}) \wedge$
$\forall(i \mid i \in (J \cap (K - L) : \varphi(i) \subseteq \psi(i))$
$\Rightarrow$ $\langle$ Weakening $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L : \varphi(i) \subseteq \psi(i)) \wedge$
$\forall(i \mid i \in (J \cap (K - L) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Range split (i.e., $\{x \mid r\} \cup \{x \mid p\} = \{x \mid r \vee p\}$) $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L \vee i \in (J \cap (K - L) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Set union axiom (i.e., $i \in A \vee i \in B \Leftrightarrow i \in A \cup B$) $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K \cap L \cup (J \cap (K - L)) : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Set theory $\rangle$
$J \subseteq K \wedge \forall(i \mid i \in J \cap K : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle\ J \subseteq K \Leftrightarrow J \cap K = J\ \rangle$
$J \subseteq K \wedge \forall(i \mid i \in J : \varphi(i) \subseteq \psi(i))$
$\Leftrightarrow$ $\langle$ Definition of $\leq$ $\rangle$
$\varphi \leq \psi$

□

Proposition 4.1(5)

Let $d(\varphi) = D_J$ and $d(\psi) = D_K$, we have $\varphi \leq \psi \Rightarrow \varphi - \psi = e_J$

Proof. Our proof strategy for $p \Rightarrow q$ is to assume p and then prove q. Assume

$$
\begin{array}{ll}
 & \varphi \leq \psi \\
\Leftrightarrow & \quad \langle \text{ Definition of } \leq \rangle \\
 & J \subseteq K \ \wedge \ \forall(i \mid i \in J : \varphi(i) \subseteq \psi(i))
\end{array}
$$

Then, prove

$$
\begin{array}{ll}
 & \varphi - \psi \\
= & \quad \langle \text{ Definition of } \varphi - \psi \rangle \\
 & \{(i, A) \mid i \in J \cap K \ \wedge \ A = \varphi(i) - \psi(i)\} \ \cup \ \{(i, A) \mid i \in J - K \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \ J \subseteq K \text{ from the assumption } \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \varphi(i) - \psi(i)\} \ \cup \ \{(i, A) \mid i \in \emptyset \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \ \forall(i \mid i \in J : \varphi(i) \subseteq \psi(i)) \text{ from the assumption \&} \\
 & \qquad S \subseteq T \Leftrightarrow S - T = \emptyset \ \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \emptyset\} \ \cup \ \{(i, A) \mid i \in \emptyset \ \wedge \ A = \varphi(i)\} \\
= & \quad \langle \ i \in \emptyset \Leftrightarrow \mathsf{false} \ \& \ p \ \wedge \ \mathsf{false} \Leftrightarrow \mathsf{false} \ \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \emptyset\} \ \cup \ \{(i, A) \mid \mathsf{false}\} \\
= & \quad \langle \text{ Empty range } \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \emptyset\} \ \cup \ \emptyset \\
= & \quad \langle \text{ Identity of } \cup \rangle \\
 & \{(i, A) \mid i \in J \ \wedge \ A = \emptyset\} \\
= & \quad \langle \text{ Definition of } e_J \rangle \\
 & e_J
\end{array}
$$

□

Proposition 4.1(6)

For $d(\varphi) = D_J$ and $d(\psi) = D_K$, we have $(\varphi\psi - \psi)^{\downarrow D_J} \leq \varphi$

Proof.

$$
\begin{array}{ll}
 & (\varphi\psi - \psi)^{\downarrow D_J} \leq \varphi \\
\Leftarrow & \quad \langle \text{ Definition of } \varphi\psi - \psi \rangle \\
 & (\{(i, A) \mid i \in (J \cup K) \cap K \ \wedge \ A = (\varphi\psi)(i) - \psi(i)\}
\end{array}
$$

$$
\begin{array}{ll}
 & \cup\ \{(i,A)\ \mid\ i \in (J \cup K) - K\ \wedge\ A = (\varphi\psi)(i)\})^{\downarrow_{D_J}} \leq \varphi \\
\Leftarrow & \quad \langle\ (J \cup K) \cap K = K\ \rangle \\
 & (\{(i,A)\ \mid\ i \in K\ \wedge\ A = (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (J \cup K) - K\ \wedge\ A = (\varphi\psi)(i)\})^{\downarrow_{D_J}} \leq \varphi \\
\Leftarrow & \quad \langle\ (J \cup K) - K = (J - K) \cup (K - K) = (J - K)\ \rangle \\
 & (\{(i,A)\ \mid\ i \in K\ \wedge\ A = (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = (\varphi\psi)(i)\})^{\downarrow_{D_J}} \leq \varphi \\
\Leftarrow & \quad \langle\ \text{Definition of } \downarrow_{D_J}\ \rangle \\
 & (\mathcal{I}_J;(\{(i,A)\ \mid\ i \in K\ \wedge\ A = (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = (\varphi\psi)(i)\})) \leq \varphi \\
\Leftarrow & \quad \langle\ \text{Distributivity of } ; \text{ over } \cup\ \rangle \\
 & \mathcal{I}_J;\{(i,A)\ \mid\ i \in K\ \wedge\ A = (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \mathcal{I}_J;\{(i,A)\ \mid\ i \in J - K\ \wedge\ A = (\varphi\psi)(i)\}) \leq \varphi \\
\Leftarrow & \quad \langle\ \text{Proposition A.1(1)}\ \rangle \\
 & (\{(i,A)\ \mid\ i \in K \cap J\ \wedge\ A = (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (J - K) \cap J\ \wedge\ A = (\varphi\psi)(i)\}) \leq \varphi \\
\Leftarrow & \quad \langle\ (J - K) \cap J = (J - K)\ \rangle \\
 & (\{(i,A)\ \mid\ i \in K \cap J\ \wedge\ (\varphi\psi)(i) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ (\varphi\psi)(i)\}) \leq \varphi \\
\Leftarrow & \quad \langle\ \text{Definition of combining information}\ \rangle \\
 & (\{(i,A)\ \mid\ i \in K \cap J\ \wedge\ A = (\varphi(i) \cup \psi(i)) - \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in J - K\ \wedge\ A = \varphi(i)\}) \leq \varphi \\
\Leftarrow & \quad \langle\ \text{Definition of } \leq\ \rangle \\
 & ((K \cap J) \cup (J - K)) \subseteq J\ \wedge \\
 & \forall(i\ \mid\ i \in K \cap J\ :\ (\varphi(i) \cup \psi(i) - \psi(i)) \subseteq \varphi(i)\ )\ \wedge \\
 & \forall(i\ \mid\ i \in J - K\ :\ \varphi(i) \subseteq \varphi(i)\ ) \\
\Leftarrow & \quad \langle\ (K \cap J) \cup (J - K) = J\ \rangle \\
 & J \subseteq J\ \wedge \\
 & \forall(i\ \mid\ i \in K \cap J\ :\ (\varphi(i) \cup \psi(i) - \psi(i)) \subseteq \varphi(i)\ )\ \wedge \\
 & \forall(i\ \mid\ i \in J - K\ :\ \varphi(i) \subseteq \varphi(i)\ ) \\
\Leftarrow & \quad \langle\ J \subseteq J \Leftrightarrow \mathsf{true}\ \&\ p\ \wedge\ \mathsf{true} \Leftrightarrow p\ \rangle \\
 & \forall(i\ \mid\ i \in K \cap J\ :\ (\varphi(i) \cup \psi(i) - \psi(i)) \subseteq \varphi(i)\ )\ \wedge \\
 & \forall(i\ \mid\ i \in J - K\ :\ \varphi(i) \subseteq \varphi(i)\ ) \\
\Leftarrow & \quad \langle\ (\varphi(i) \cup \psi(i) - \psi(i)) \subseteq \varphi \Leftrightarrow \mathsf{true}\ \rangle \\
 & \forall(i\ \mid\ i \in K \cap J\ :\ \mathsf{true}\ )\ \wedge \\
 & \forall(i\ \mid\ i \in J - K\ :\ \varphi(i) \subseteq \varphi(i)\ )
\end{array}
$$

$$
\begin{array}{ll}
\Leftarrow & \langle\ \varphi(i) \subseteq \varphi(i) \Leftrightarrow \mathsf{true}\ \rangle \\
& \forall(i\ \mid\ i \in K \cap J\ :\ \mathsf{true}\,)\ \wedge \\
& \forall(i\ \mid\ i \in J - K\ :\ \mathsf{true}\,) \\
\Leftarrow & \langle\ \forall\text{-True body}\ \rangle \\
& \mathsf{true}\ \wedge\ \mathsf{true} \\
\Leftarrow & \langle\ \mathsf{true}\ \wedge\ \mathsf{true} \Leftrightarrow \mathsf{true}\ \rangle \\
& \mathsf{true}
\end{array}
$$

□

Proposition 4.1(7)

Let $d(\varphi) = D_J$, $d(\psi) = D_K$, and $d(\chi) = D_L$ for $J, K, L \subseteq I$, we have $\varphi \leq \psi \Rightarrow (\chi - \varphi)\psi = \chi\psi$

Proof. Our proof strategy for $p \Rightarrow q$ is to assume p and then prove q. Assume

$$
\begin{array}{ll}
& \varphi \leq \psi \\
\Leftrightarrow & \langle\ \text{Definition of } \leq\ \rangle \\
& J \subseteq K\ \wedge\ \forall(i\ \mid\ i \in J\ :\ \varphi(i) \subseteq \psi(i)\,)
\end{array}
$$

Then, prove

$$
\begin{array}{ll}
& (\chi - \varphi)\psi \\
= & \langle\ \text{The definition of combining information}\ \rangle \\
& \{(i, A)\ \mid\ i \in L \cap K\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
& \quad \cup\ \{(i, A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i, A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \langle\ p\ \wedge\ \mathsf{true} \Leftrightarrow p\ \rangle \\
& \{(i, A)\ \mid\ i \in L \cap K\ \wedge\ \mathsf{true}\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
& \quad \cup\ \{(i, A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i, A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \langle\ i \in I \Leftrightarrow \mathsf{true}\ \rangle \\
& \{(i, A)\ \mid\ i \in L \cap K\ \wedge\ i \in I\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
& \quad \cup\ \{(i, A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i, A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \langle\ J \cup \overline{J} = I\ \rangle \\
& \{(i, A)\ \mid\ i \in L \cap K\ \wedge\ (i \in J \cup \overline{J})\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
& \quad \cup\ \{(i, A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i, A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \langle\ \text{Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B)\ \rangle \\
& \{(i, A)\ \mid\ i \in L \cap K\ \wedge\ (i \in J\ \vee\ i \in \overline{J})\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\}
\end{array}
$$

$$
\begin{array}{ll}
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{The distributivity of } \wedge \text{ over } \vee\ \rangle \\
 & \{(i,A)\ \mid\ (i \in L \cap K\ \wedge\ i \in J\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i))\ \ \vee \\
 & \quad (i \in L \cap K\ \wedge\ i \in \overline{J}\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i))\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Set intersection axiom (i.e., } i \in A\ \wedge\ i \in B \Leftrightarrow i \in A \cap B)\ \rangle \\
 & \{(i,A)\ \mid\ (i \in (L \cap K) \cap J\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i))\ \ \vee \\
 & \quad (i \in (L \cap K) \cap \overline{J}\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i))\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Range split (i.e., } \{x\ \mid\ r\} \cup \{x\ \mid\ p\} = \{x\ \mid\ r \vee p\})\ \rangle \\
 & \{(i,A)\ \mid\ i \in (L \cap K) \cap J\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (L \cap K) \cap \overline{J}\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Definition of set difference}\ \rangle \\
 & \{(i,A)\ \mid\ i \in (L \cap K) \cap J\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (L \cap K) - J\ \wedge\ A = (\chi - \varphi)(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = (\chi - \varphi)(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Definition of } \chi - \varphi\ \rangle \\
 & \{(i,A)\ \mid\ i \in (L \cap K) \cap J\ \wedge\ A = (\chi(i) - \varphi(i)) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (L \cap K) - J\ \wedge\ A = \chi(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = \chi(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \forall(i\ \mid\ i \in J\ :\ \varphi(i) \subseteq \psi(i))\ \rangle \\
 & \{(i,A)\ \mid\ i \in (L \cap K) \cap J\ \wedge\ A = \chi(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in (L \cap K) - J\ \wedge\ A = \chi(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = \chi(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Range split (i.e., } \{x\ \mid\ r\} \cup \{x\ \mid\ p\} = \{x\ \mid\ r \vee p\})\ \rangle \\
 & \{(i,A)\ \mid \\
 & \quad (i \in (L \cap K) \cap J\ \wedge\ A = \chi(i) \cup \psi(i))\ \ \vee\ \ (i \in (L \cap K) - J\ \wedge\ A = \chi(i) \cup \psi(i))\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = \chi(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Distributivity of } \wedge \text{ over } \vee\ \rangle \\
 & \{(i,A)\ \mid\ (i \in (L \cap K) \cap J\ \vee\ i \in (L \cap K) - J)\ \ \wedge\ \ A = \chi(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = \chi(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ \text{Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B)\ \rangle \\
 & \{(i,A)\ \mid\ i \in ((L \cap K) \cap J) \cup ((L \cap K) - J)\ \wedge\ A = \chi(i) \cup \psi(i)\} \\
 & \cup\ \{(i,A)\ \mid\ i \in L - K\ \wedge\ A = \chi(i)\}\ \cup\ \{(i,A)\ \mid\ i \in K - L\ \wedge\ A = \psi(i)\} \\
= & \quad \langle\ ((L \cap K) \cap J) \cup ((L \cap K) - J) = L \cap K\ \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & \{(i, A) \mid i \in L \cap K \ \wedge \ A = \chi(i) \cup \psi(i)\} \\
 & \ \cup \ \{(i, A) \mid i \in L - K \ \wedge \ A = \chi(i)\} \ \cup \ \{(i, A) \mid i \in K - L \ \wedge \ A = \psi(i)\} \\
= & \qquad \langle \text{ Definition of combining information } \rangle \\
 & \chi\psi
\end{array}
$$

□

C The Detailed Proofs of Proposition 4.2

Proposition 4.2(1)

Let $J = \{j\}$, $K = \{k\}$, and $d(\varphi) = D_L$, we have
$D_J \preccurlyeq d(\varphi) \quad \vee \quad \varphi = fs(\varphi, D_J, D_K)$

Proof.

$$
\begin{array}{ll}
 & D_J \preccurlyeq d(\varphi) \quad \vee \quad \varphi = fs(\varphi, D_J, D_K) \\
\Leftrightarrow & \qquad \langle\ p \vee q \Leftrightarrow \neg p \Rightarrow q\ \rangle \\
 & \neg(D_J \preccurlyeq d(\varphi)) \Rightarrow \varphi = fs(\varphi, D_J, D_K)
\end{array}
$$

Our proof strategy for $p \Rightarrow q$ is to assume p and then prove q. We assume

$$
\begin{array}{ll}
 & \neg(D_J \preccurlyeq d(\varphi)) \\
\Leftrightarrow & \qquad \langle\ d(\varphi) = D_L\ \rangle \\
 & \neg(D_J \preccurlyeq D_L) \\
\Rightarrow & \qquad \langle\ \text{Proposition A.2(2)}\ \rangle \\
 & \neg(J \subseteq L) \\
\Rightarrow & \qquad \langle\ \text{Proposition A.12}\ \rangle \\
 & J \cap L = \emptyset
\end{array}
$$

Then we prove $fs(\varphi, D_J, D_K) = \varphi$

$$
\begin{array}{ll}
 & fs(\varphi, D_J, D_K) \\
= & \qquad \langle\ \text{Definition of } fs(\varphi, D_J, D_K)\ \rangle \\
 & \varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]) \\
= & \qquad \langle\ (J \cap L = \emptyset) \Rightarrow L - J = L\ \rangle \\
 & \varphi^{\downarrow D_L} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]) \\
= & \qquad \langle\ \text{Proposition A.3(4)}\ \rangle \\
 & \varphi \cdot (\varphi^{\downarrow D_J}[D_K/D_J]) \\
= & \qquad \langle\ \varphi \in D_L\ \rangle \\
 & \varphi \cdot (\{(i, A) \mid i \in L \ \wedge\ A = \varphi(i)\}^{\downarrow D_J}[D_K/D_J]) \\
= & \qquad \langle\ \text{Definition of } \downarrow_{D_J}\ \rangle \\
 & \varphi \cdot (\mathcal{I}_J{:}\{(i, A) \mid i \in L \ \wedge\ A = \varphi(i)\}[D_K/D_J]) \\
= & \qquad \langle\ \text{Proposition A.1(1)}\ \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & \varphi \cdot (\{(i, A) \mid i \in L \cap J \ \wedge \ A = \varphi(i)\}[D_K/D_J]) \\
= & \quad \langle\ J \cap L = \emptyset\ \rangle \\
 & \varphi \cdot (\{(i, A) \mid i \in \emptyset \ \wedge \ A = \varphi(i)\}[D_K/D_J]) \\
= & \quad \langle\ i \in \emptyset \Leftrightarrow \mathsf{false}\ \rangle \\
 & \varphi \cdot (\{(i, A) \mid \mathsf{false} \ \wedge \ A = \varphi(i)\}[D_K/D_J]) \\
= & \quad \langle\ p \ \wedge \ \mathsf{false} \Leftrightarrow \mathsf{false}\ \rangle \\
 & \varphi \cdot (\{(i, A) \mid \mathsf{false}\}[D_K/D_J]) \\
= & \quad \langle\ \text{Empty range}\ \rangle \\
 & \varphi \cdot (\emptyset[D_K/D_J]) \\
= & \quad \langle\ \text{Frame substitution of an empty set}\ \rangle \\
 & \varphi \cdot \emptyset \\
= & \quad \langle\ \text{Proposition A.7}\ \rangle \\
 & \varphi \cdot e_\emptyset \\
= & \quad \langle\ \text{Proposition A.6}\ \rangle \\
 & \varphi
\end{array}
$$

□

Proposition 4.2(2)

Let $J = \{j\}$, $K = \{k\}$, and $d(\varphi) = D_L$, we have
$D_K \preccurlyeq d(\varphi) \quad \vee \quad \varphi = fs(fs(\varphi, D_J, D_K), D_K, D_J)$

Proof.

$$
\begin{array}{ll}
 & D_K \preccurlyeq d(\varphi) \quad \vee \quad \varphi = fs(fs(\varphi, D_J, D_K), D_K, D_J) \\
\Leftrightarrow & \quad \langle\ p \vee q \Leftrightarrow \neg p \Rightarrow q\ \rangle \\
 & \neg(D_K \preccurlyeq d(\varphi)) \Rightarrow \varphi = fs(fs(\varphi, D_J, D_K), D_K, D_J)
\end{array}
$$

Our proof strategy for $p \Rightarrow q$ is to assume p and then prove q. We assume

$$
\begin{array}{ll}
 & \neg(D_K \preccurlyeq d(\varphi)) \\
\Leftrightarrow & \quad \langle\ d(\varphi) = D_L\ \rangle \\
 & \neg(D_K \preccurlyeq D_L) \\
\Rightarrow & \quad \langle\ \text{Proposition A.2(2)}\ \rangle \\
 & \neg(K \subseteq L) \\
\Rightarrow & \quad \langle\ \text{Proposition A.12}\ \rangle \\
 & K \cap L = \emptyset
\end{array}
$$

To prove $\varphi = fs(fs(\varphi, D_J, D_K), D_K, D_J)$, we have two cases:

1. $\neg(D_J \preccurlyeq d(\varphi))$
2. $D_J \preccurlyeq d(\varphi)$

1. Assume $\neg(D_J \preccurlyeq d(\varphi))$

$$
\begin{array}{ll}
& fs(fs(\varphi, D_J, D_K), D_K, D_J) \\
= & \quad \langle \text{ Proposition 4.2(1) \& } \neg D_J \preccurlyeq d(\varphi) \ \rangle \\
& fs(\varphi, D_K, D_J) \\
= & \quad \langle \text{ Proposition 4.2(1) \& } \neg D_K \preccurlyeq d(\varphi) \ \rangle \\
& \varphi
\end{array}
$$

2. Assume $D_J \preccurlyeq d(\varphi) \Rightarrow J \subseteq L \Rightarrow J \cap L = J$

$$
\begin{array}{ll}
& fs(fs(\varphi, D_J, D_K), D_K, D_J) \\
= & \quad \langle \text{ Definition of } fs \ \rangle \\
& fs(\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]), D_K, D_J) \\
= & \quad \langle \text{ Definition of } fs \ \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_{(L-K)}} \cdot ((\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_K}[D_J/D_K]) \\
= & \quad \langle \ (K \cap L = \emptyset) \Rightarrow L - K = L \ \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot ((\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_K}[D_J/D_K]) \\
= & \quad \langle \text{ Proposition 2.1(2, 9) \& } d(\varphi^{\downarrow D_J}[D_K/D_J]) = D_K \ \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot ((\varphi^{\downarrow D_{(L-J)\cap K}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))[D_J/D_K]) \\
= & \quad \langle \ K \cap L = \emptyset \ \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot ((\varphi^{\downarrow D_\emptyset} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))[D_J/D_K]) \\
= & \quad \langle \text{ Proposition A.4 } \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot ((e_\emptyset \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))[D_J/D_K]) \\
= & \quad \langle \text{ Proposition A.6 } \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot ((\varphi^{\downarrow D_J}[D_K/D_J])[D_J/D_K]) \\
= & \quad \langle \text{ Replace } J \text{ by } K \text{ and then } K \text{ by } J \text{ is equivelant to replace } J \text{ by } J \ \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot (\varphi^{\downarrow D_J}[D_J/D_J]) \\
= & \quad \langle \text{ Replacing } J \text{ by } J \text{ does not affect the information } \rangle \\
& (\varphi^{\downarrow D_{(L-J)}} \cdot (\varphi^{\downarrow D_J}[D_K/D_J]))^{\downarrow D_L} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle \text{ Proposition A.9 } \rangle \\
& (\varphi^{\downarrow D_{(L-J)}})^{\downarrow D_L} \cdot (\varphi^{\downarrow D_J}[D_K/D_J])^{\downarrow D_L} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle \text{ Proposition A.11 } \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & (\varphi^{\downarrow D_{(L-J)\cap L}}) \cdot (\varphi^{\downarrow D_J}[D_K/D_J])^{\downarrow D_L} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ (L-J)\cap L = L \cap \overline{J} \cap L = L \cap \overline{J} = L - J\ \rangle \\
 & (\varphi^{\downarrow D_{(L-J)}}) \cdot (\varphi^{\downarrow D_J}[D_K/D_J])^{\downarrow D_L} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ \text{Proposition A.10 \& } d(\varphi^{\downarrow D_J}[D_K/D_J]) = D_K\ \rangle \\
 & (\varphi^{\downarrow D_{(L-J)}}) \cdot (\varphi^{\downarrow D_J}[D_K/D_J])^{\downarrow D_{L\cap K}} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ K \cap L = \emptyset\ \rangle \\
 & (\varphi^{\downarrow D_{(L-J)}}) \cdot (\varphi^{\downarrow D_J}[D_K/D_J])^{\downarrow D_\emptyset} \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ \text{Proposition A.4}\ \rangle \\
 & (\varphi^{\downarrow D_{(L-J)}}) \cdot e_\emptyset \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ \text{Proposition A.6}\ \rangle \\
 & (\varphi^{\downarrow D_{(L-J)}}) \cdot \varphi^{\downarrow D_J} \\
= & \quad \langle\ \text{Proposition A.8}\ \rangle \\
 & \varphi^{\downarrow D_{(L-J)\cup J}} \\
= & \quad \langle\ D_J \preccurlyeq d(\varphi) \Rightarrow J \subseteq L\ \&\ (L-J)\cup J = L\ \rangle \\
 & \varphi^{\downarrow D_L} \\
= & \quad \langle\ \text{Proposition A.3(1)}\ \rangle \\
 & \varphi
\end{array}
$$

□

D The Detailed Proofs of Proposition 4.3

Proposition 4.3(1)

$\varphi \leq \psi \ \wedge \ \varphi \leq \chi \Rightarrow \mathit{update}(\mathit{update}(\mathcal{N},\ \varphi,\ \psi),\ \varphi,\ \chi) = \mathit{update}(\mathcal{N},\ \varphi,\ \psi \cdot \chi)$.

Proof. Let $\mathit{update}(\mathcal{N},\ \varphi,\ \psi) = (\Omega, D)$ where $\Omega = \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \tau = (\chi_2 - \varphi) \cdot \psi)\} \ \cup \ \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \tau = \chi_2)\}$. Also, let $\mathit{update}((\Omega, D),\ \varphi,\ \chi) = (\Psi, D)$. We prove that $\Psi = \Omega[\psi \cdot \chi/\psi]$

$$
\begin{array}{ll}
 & \Psi \\
 & \qquad \langle \text{ Definition of the function update } \rangle \\
= & \{\tau \mid \exists(\chi_1 \mid \chi_1 \in \Omega : \varphi \leq \chi_1 \ \wedge \ \tau = (\chi_1 - \varphi) \cdot \chi)\} \\
 & \cup \ \{\tau \mid \exists(\chi_1 \mid \chi_1 \in \Omega : \neg(\varphi \leq \chi_1) \ \wedge \ \tau = \chi_1)\} \\
 & \qquad \langle \text{ Definition of the function update } \rangle \\
= & \{\tau \mid \exists(\chi_1 \mid \chi_1 \in \\
 & \qquad \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \tau_1 = (\chi_2 - \varphi) \cdot \psi)\} \\
 & \qquad \cup \ \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \tau_1 = \chi_2)\} : \\
 & \qquad \varphi \leq \chi_1 \ \wedge \ \tau = (\chi_1 - \varphi) \cdot \chi)\} \\
 & \cup \ \{\tau \mid \exists(\chi_1 \mid \chi_1 \in \\
 & \qquad \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \tau_1 = (\chi_2 - \varphi) \cdot \psi)\} \\
 & \qquad \cup \ \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \tau_1 = \chi_2)\} : \\
 & \qquad \neg(\varphi \leq \chi_1) \ \wedge \ \tau = \chi_1)\} \\
= & \qquad \langle \text{ Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \rangle \\
 & \{\tau \mid \exists(\chi_1 \mid \\
 & \qquad \chi_1 \in \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \tau_1 = (\chi_2 - \varphi) \cdot \psi)\} \\
 & \qquad \vee \ \ \chi_1 \in \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \tau_1 = \chi_2)\} : \\
 & \qquad \varphi \leq \chi_1 \ \wedge \ \tau = (\chi_1 - \varphi) \cdot \chi)\} \\
 & \cup \ \{\tau \mid \exists(\chi_1 \mid \\
 & \qquad \chi_1 \in \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \tau_1 = (\chi_2 - \varphi) \cdot \psi)\} \\
 & \qquad \vee \ \ \chi_1 \in \{\tau_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \tau_1 = \chi_2)\} : \\
 & \qquad \neg(\varphi \leq \chi_1) \ \wedge \ \tau = \chi_1)\} \\
= & \qquad \langle \ y \in \{x \mid r\} \Leftrightarrow r[x := y] \ \rangle \\
 & \{\tau \mid \exists(\chi_1 \mid \\
 & \qquad \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge \ \chi_1 = (\chi_2 - \varphi) \cdot \psi) \\
 & \qquad \vee \ \ \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2) : \\
 & \qquad \varphi \leq \chi_1 \ \wedge \ \tau = (\chi_1 - \varphi) \cdot \chi)\}
\end{array}
$$

$$
\begin{array}{ll}
& \cup\ \{\tau \mid \exists(\chi_1 \mid \\
& \qquad \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi\,) \\
& \qquad \vee\ \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2\,) : \\
& \qquad \neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1\,)\} \\
= & \qquad \langle\ \text{Distributivity axiom}\ \rangle \\
& \{\tau \mid \exists(\chi_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \\
& \qquad (\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\,) : \\
& \qquad \varphi \le \chi_1 \ \wedge\ \tau = (\chi_1 - \varphi)\cdot\chi\,)\} \\
& \cup\ \{\tau \mid \exists(\chi_1 \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \\
& \qquad (\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\,) : \\
& \qquad \neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1\,)\} \\
= & \qquad \langle\ \text{Trading rule for}\ \exists\ \rangle \\
& \{\tau \mid \exists(\chi_1 \mid\!: \exists(\chi_2 \mid \chi_2 \in \Phi : \\
& \qquad (\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\,) \\
& \qquad \wedge\ \ \varphi \le \chi_1 \ \wedge\ \tau = (\chi_1 - \varphi)\cdot\chi\,)\} \\
& \cup\ \{\tau \mid \exists(\chi_1 \mid\!: \exists(\chi_2 \mid \chi_2 \in \Phi : \\
& \qquad (\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\,) \\
& \qquad \wedge\ \ \neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1\,)\} \\
= & \qquad \langle\ \text{Nesting axiom}\ \rangle \\
& \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi : \\
& \qquad \Big((\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\Big) \\
& \qquad \wedge\ \ (\varphi \le \chi_1 \ \wedge\ \tau = (\chi_1 - \varphi)\cdot\chi)\,)\} \\
& \cup\ \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi : \\
& \qquad \Big((\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \vee\ (\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2)\Big) \\
& \qquad \wedge\ \ (\neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1)\,)\} \\
= & \qquad \langle\ \text{Distributivity of}\ \wedge\ \text{over}\ \vee\ \rangle \\
& \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi : \\
& \qquad \Big(\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi \ \wedge\ \varphi \le \chi_1 \ \wedge\ \tau = (\chi_1 - \varphi)\cdot\chi\Big) \\
& \qquad \vee\ \Big(\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2 \ \wedge\ \varphi \le \chi_1 \ \wedge\ \tau = (\chi_1 - \varphi)\cdot\chi\Big)\,)\} \\
& \cup\ \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi : \\
& \qquad \Big(\varphi \le \chi_2 \ \wedge\ \chi_1 = (\chi_2 - \varphi)\cdot\psi) \ \wedge\ \neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1\Big) \\
& \qquad \vee\ \Big(\neg(\varphi \le \chi_2) \ \wedge\ \chi_1 = \chi_2 \ \wedge\ \neg(\varphi \le \chi_1) \ \wedge\ \tau = \chi_1\Big)\,)\} \\
= & \qquad \langle\ \text{Proposition 4.1(7)}\ \rangle \\
& \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi : \\
& \qquad \Big(\varphi \le \chi_2 \ \wedge\ \chi_1 = \chi_2\cdot\psi \ \wedge\ \varphi \le \chi_1 \ \wedge\ \tau = \chi_1\cdot\chi\Big)
\end{array}
$$

$$
\begin{array}{ll}
 & \quad \vee \ \Big(\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \varphi \leq \chi_1 \ \wedge \ \tau = \chi_1 \cdot \chi\Big))\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad \Big(\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \neg(\varphi \leq \chi_1) \ \wedge \ \tau = \chi_1\Big) \\
 & \qquad \vee \ \Big(\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \neg(\varphi \leq \chi_1) \ \wedge \ \tau = \chi_1\Big))\} \\
= & \qquad \langle \text{ Substitution axiom } \rangle \\
 & \quad \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad \Big(\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \varphi \leq \chi_2 \cdot \psi \ \wedge \ \tau = \chi_2 \cdot \psi \cdot \chi\Big) \\
 & \qquad \vee \ \Big(\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \varphi \leq \chi_2 \ \wedge \ \tau = \chi_2 \cdot \chi\Big))\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad \Big(\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \neg(\varphi \leq \chi_2 \cdot \psi) \ \wedge \ \tau = \chi_2 \cdot \psi\Big) \\
 & \qquad \vee \ \Big(\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \neg(\varphi \leq \chi_2) \ \wedge \ \tau = \chi_2\Big))\} \\
= & \qquad \langle \text{ Contradiction } \rangle \\
 & \quad \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad \Big(\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \varphi \leq \chi_2 \cdot \psi \ \wedge \ \tau = \chi_2 \cdot \psi \cdot \chi\Big) \\
 & \qquad \vee \ \mathsf{false})\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad \mathsf{false} \\
 & \qquad \vee \ \Big(\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \neg(\varphi \leq \chi_2) \ \wedge \ \tau = \chi_2\Big))\} \\
= & \qquad \langle \text{ Zero for } \vee \ \rangle \\
 & \quad \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \varphi \leq \chi_2 \cdot \psi \ \wedge \ \tau = \chi_2 \cdot \psi \cdot \chi))\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \neg(\varphi \leq \chi_2) \ \wedge \ \tau = \chi_2))\} \\
= & \qquad \langle \ \varphi \leq \chi_2 \Rightarrow \varphi \leq \chi_2 \cdot \psi \ \rangle \\
 & \quad \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \tau = \chi_2 \cdot \psi \cdot \chi))\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \neg(\varphi \leq \chi_2) \ \wedge \ \tau = \chi_2))\} \\
= & \qquad \langle \text{ Idempotency of } \wedge \ \rangle \\
 & \quad \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\varphi \leq \chi_2 \ \wedge \ \chi_1 = \chi_2 \cdot \psi \ \wedge \ \tau = \chi_2 \cdot \psi \cdot \chi))\} \\
 & \cup \ \{\tau \mid \ \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ : \\
 & \qquad (\neg(\varphi \leq \chi_2) \ \wedge \ \chi_1 = \chi_2 \ \wedge \ \tau = \chi_2))\}, \ D) \\
= & \qquad \langle \text{ Trading rule for } \exists \text{ \& Symmetry of } \wedge \ \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & (\{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ \wedge\ \varphi \leq \chi_2 \ \wedge\ \tau = \chi_2 \cdot \psi \cdot \chi) : \\
 & \quad (\chi_1 = \chi_2 \cdot \psi)\} \\
 & \cup\ \{\tau \mid \exists(\chi_1, \chi_2 \mid \chi_2 \in \Phi \ \wedge\ \neg(\varphi \leq \chi_2) \ \wedge\ \tau = \chi_2) : \\
 & \quad (\chi_1 = \chi_2)\} \\
= & \quad \langle \text{ Nesting axiom } \rangle \\
 & \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi \ \wedge\ \varphi \leq \chi_2 \ \wedge\ \tau = \chi_2 \cdot \psi \cdot \chi) : \\
 & \quad \exists(\chi_1 \mid: (\chi_1 = \chi_2 \cdot \psi))\} \\
 & \cup\ \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi \ \wedge\ \neg(\varphi \leq \chi_2) \ \wedge\ \tau = \chi_2) : \\
 & \quad \exists(\chi_1 \mid: (\chi_1 = \chi_2))\} \\
= & \quad \langle \text{ The information } \chi_1 \text{ and combining information is always defined } \rangle \\
 & \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi \ \wedge\ \varphi \leq \chi_2 \ \wedge\ \tau = \chi_2 \cdot \psi \cdot \chi) : \\
 & \quad \mathsf{true})\} \\
 & \cup\ \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi \ \wedge\ \neg(\varphi \leq \chi_2) \ \wedge\ \tau = \chi_2) : \\
 & \quad \mathsf{true})\} \\
= & \quad \langle \text{ Tradong rule for } \exists \rangle \\
 & \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge\ \tau = \chi_2 \cdot \psi \cdot \chi)\} \\
 & \cup\ \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge\ \tau = \chi_2))\} \\
= & \quad \langle \text{ Proposition 4.1(7) } \rangle \\
 & \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \varphi \leq \chi_2 \ \wedge\ \tau = (\chi_2 - \varphi) \cdot \psi \cdot \chi)\} \\
 & \cup\ \{\tau \mid \exists(\chi_2 \mid \chi_2 \in \Phi : \neg(\varphi \leq \chi_2) \ \wedge\ \tau = \chi_2))\} \\
= & \quad \langle \text{ The definition of } \Omega \rangle \\
 & \Omega[\psi \cdot \chi / \psi]
\end{array}
$$

□

Proposition 4.3(2)

$isInKnowledge(\mathcal{N},\ d(\varphi),\ \varphi) \quad \vee \quad update(\mathcal{N},\ \varphi,\ \psi) = \mathcal{N}$

Proof.

$$
\begin{array}{ll}
 & isInKnowledge(\mathcal{N},\ d(\varphi),\ \varphi) \quad \vee \quad update(\mathcal{N},\ \varphi,\ \psi) = \mathcal{N} \\
\Leftrightarrow & \quad \langle\ p \Rightarrow q \Leftrightarrow \neg p \vee q\ \rangle \\
 & \neg isInKnowledge(\mathcal{N},\ d(\varphi),\ \varphi) \Rightarrow update(\mathcal{N},\ \varphi,\ \psi) = \mathcal{N}
\end{array}
$$

First, we assume that

$$
\begin{array}{ll}
& \neg \mathit{isInKnowledge}(\mathcal{N},\ d(\varphi),\ \varphi) \\
\Leftrightarrow & \quad \langle \text{ Definition of } \mathit{isInKnow} \ \rangle \\
& \neg\ \exists(\chi \mid \chi \in \Phi : d(\varphi) \in D \ \wedge\ \varphi \leq \chi \ \wedge\ d(\varphi) \preccurlyeq d(\chi)\,) \\
\Leftrightarrow & \quad \langle \text{ De Morgan } \rangle \\
& \forall(\chi \mid \chi \in \Phi : \neg(d(\varphi) \in D \ \wedge\ \varphi \leq \psi \ \wedge\ d(\varphi) \preccurlyeq d(\chi))\,) \\
\Leftrightarrow & \quad \langle \text{ De Morgan } \rangle \\
& \forall(\chi \mid \chi \in \Phi : \neg(d(\varphi) \in D) \ \vee\ \neg(\varphi \leq \chi) \ \vee\ (\neg d(\varphi) \preccurlyeq d(\chi))\,)
\end{array}
$$

$\{\chi \mid \chi \in \Phi \ \wedge\ \neg(\varphi \leq \chi)\},\ D)$.

Based on the assumption, we prove that for $d(\varphi) = D_J$ and $d(\psi) = D_K$, we have $\exists(\chi \mid \chi \in \Phi : \varphi \leq \chi\,) \Rightarrow \mathsf{false}$

$$
\begin{array}{ll}
& \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi\,) \\
\Rightarrow & \quad \langle\ \varphi \leq \chi \Rightarrow J \subseteq K \text{ from the definition of } \leq \rangle \\
& \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi \ \wedge\ J \subseteq K\,) \\
\Rightarrow & \quad \langle \text{ Proposition A.2(1) } \rangle \\
& \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi \ \wedge\ D_J \preccurlyeq D_K\,) \\
\Rightarrow & \quad \langle\ d(\varphi) = D_J \text{ and } d(\psi) = D_K \ \rangle \\
& \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi \ \wedge\ d(\varphi) \preccurlyeq d(\chi)\,) \\
\Rightarrow & \quad \langle\ d(\chi) \in D \ \wedge\ J \subseteq K \Rightarrow d(\varphi) \in D \ \rangle \\
& \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi \ \wedge\ d(\varphi) \preccurlyeq d(\chi) \ \wedge\ d(\varphi) \in D\,) \\
\Rightarrow & \quad \langle \text{ The assumption } \rangle \\
& \mathsf{false}
\end{array}
$$

Therefore, we have $\neg\ \exists(\chi \mid \chi \in \Phi : \varphi \leq \chi\,) \Leftrightarrow\ \forall(\chi \mid \chi \in \Phi : \neg(\varphi \leq \chi)\,) \Leftrightarrow \mathsf{true}$. Then we prove that $\mathit{update}(\mathcal{N},\ \varphi,\ \psi) = \mathcal{N}$

$$
\begin{array}{ll}
& \mathit{update}(\mathcal{N},\ \varphi,\ \psi) \\
= & \quad \langle \text{ Definition of } \mathit{update} \ \rangle \\
& (\{(\chi - \varphi) \cdot \psi \mid \chi \in \Phi \ \wedge\ \varphi \leq \chi\} \ \cup\ \{\chi \mid \chi \in \Phi \ \wedge\ \neg(\varphi \leq \chi)\},\ D) \\
= & \quad \langle \text{ From (1) } \rangle \\
& (\{(\chi - \varphi) \cdot \psi \mid \chi \in \Phi \ \wedge\ \mathsf{false}\} \ \cup\ \{\chi \mid \chi \in \Phi \ \wedge\ \mathsf{true}\},\ D) \\
= & \quad \langle \text{ Zero of } \wedge \text{ and } \vee \ \rangle \\
& (\{(\chi - \varphi) \cdot \psi \mid \mathsf{false}\} \ \cup\ \{\chi \mid \chi \in \Phi\},\ D) \\
= & \quad \langle \text{ Empty range } \rangle \\
& \emptyset \ \cup\ \{\chi \mid \chi \in \Phi\},\ D)
\end{array}
$$

$=$ $\langle$ Identity of $\cup$ $\rangle$
$\{\chi \mid \chi \in \Phi\},\ D)$
$=$ $\langle$ Definition of $\mathcal{N}$ $\rangle$
$\mathcal{N}$

□

E The Detailed Proofs of Section 4.3

Proposition 4.4

let $\mathcal{N} = (\{e_\emptyset\}, D)$ be an empty knowledge, we have:

1. $update(\mathcal{N},\ \varphi,\ e_\emptyset) = \mathcal{N}$
2. $insert(\mathcal{N}, e_\emptyset) = \mathcal{N}$
3. $choose(extract(\mathcal{N},\ x,\ \varphi)) = e_\emptyset$

Proof. 1. For $\mathcal{N} = (\{e_\emptyset\}, D)$, we have $update(\mathcal{N},\ \varphi,\ e_\emptyset) = (\Psi, D)$ where

$$
\begin{array}{ll}
& \Psi \\
= & \quad \langle \text{ Definition of } update \ \rangle \\
& \{(\chi - \varphi) \cdot e_\emptyset \mid \chi \in \Phi \ \wedge\ \varphi \leq \chi\} \ \cup\ \{\chi \mid \chi \in \Phi \ \wedge\ \neg(\varphi \leq \chi)\} \\
= & \quad \langle \text{ Proposition A.6 } \rangle \\
& \{\chi - \varphi \mid \chi \in \Phi \ \wedge\ \varphi \leq \chi\} \ \cup\ \{\chi \mid \chi \in \Phi \ \wedge\ \neg(\varphi \leq \chi)\} \\
= & \quad \langle \text{ Singleton membership } \rangle \\
& \{\chi - \varphi \mid \chi = e_\emptyset \ \wedge\ \varphi \leq \chi\} \ \cup\ \{\chi \mid \chi = e_\emptyset \ \wedge\ \neg(\varphi \leq \chi)\} \\
= & \quad \langle \text{ Rewriting } \rangle \\
& \{e_\emptyset - \varphi \mid \chi = e_\emptyset \ \wedge\ \varphi \leq \chi\} \ \cup\ \{e_\emptyset \mid \chi = e_\emptyset \ \wedge\ \neg(\varphi \leq \chi)\} \\
= & \quad \langle \text{ Proposition 4.1(3) } \rangle \\
& \{e_\emptyset \mid \chi = e_\emptyset \ \wedge\ \varphi \leq \chi\} \ \cup\ \{e_\emptyset \mid \chi = e_\emptyset \ \wedge\ \neg(\varphi \leq \chi)\} \\
= & \quad \langle \text{ Set union axiom (i.e., } i \in A \vee i \in B \Leftrightarrow i \in A \cup B) \ \rangle \\
& \{e_\emptyset \mid (\chi = e_\emptyset \ \wedge\ \varphi \leq \chi) \ \vee\ (\chi = e_\emptyset \ \wedge\ \neg(\varphi \leq \chi))\} \\
= & \quad \langle \text{ Distributivity of } \wedge \text{ over } \vee \ \rangle \\
& \{e_\emptyset \mid \chi = e_\emptyset \quad \wedge \quad (\varphi \leq \chi \vee \neg(\varphi \leq \chi))\} \\
= & \quad \langle \text{ Excluded middle axiom } \rangle \\
& \{e_\emptyset \mid \chi = e_\emptyset \quad \wedge \quad \mathsf{true}\} \\
= & \quad \langle \text{ Zero of } \vee \ \rangle \\
& \{e_\emptyset \mid \chi = e_\emptyset\} \\
= & \quad \langle \text{ The set contains only } e_\emptyset \ \rangle \\
& \{e_\emptyset\}
\end{array}
$$

2. $insert(\mathcal{N}, e_\emptyset) = \mathcal{N}$

$$
\begin{array}{ll}
& insert((\{e_\emptyset\}, D), e_\emptyset) \\
= & \quad \langle \text{ Definition of } insert \ \rangle \\
& (\{e_\emptyset\} \cup \{e_\emptyset\}, D) \\
= & \quad \langle \text{ Idempotency of } \cup \ \rangle \\
& (\{e_\emptyset\}, D) \\
= & \quad \langle \text{ Definition of } \mathcal{N} \ \rangle
\end{array}
$$

$\mathcal{N}$

3. $choose(extract(\mathcal{N},\ x,\ \varphi)) = e_\emptyset$

$$
\begin{array}{ll}
 & choose(extract(\mathcal{N},\ x,\ \varphi)) \\
= & \quad \langle \text{ Definition of } extract \ \rangle \\
 & choose(\{\psi^{\downarrow x} \mid x \in D \ \wedge\ \psi \in \Phi \ \wedge\ \varphi \leq \psi \ \wedge\ x \preccurlyeq d(\psi)\}) \\
= & \quad \langle \text{ Singleton membership } \rangle \\
 & choose(\{\psi^{\downarrow x} \mid x \in D \ \wedge\ \psi = e_\emptyset \ \wedge\ \varphi \leq \psi \ \wedge\ x \preccurlyeq d(\psi)\}) \\
= & \quad \langle \text{ Rewriting } \rangle \\
 & choose(\{e_\emptyset^{\downarrow x} \mid x \in D \ \wedge\ \psi = e_\emptyset \ \wedge\ \varphi \leq e_\emptyset \ \wedge\ x \preccurlyeq d(\psi)\}) \\
= & \quad \langle \text{ Proposition A.5 } \rangle \\
 & choose(\{e_\emptyset \mid x \in D \ \wedge\ \psi = e_\emptyset \ \wedge\ \varphi \leq \psi \ \wedge\ x \preccurlyeq d(\psi)\}) \\
= & \quad \langle \text{ Definition of } choose \text{ (i.e., choose selects an element randomly from a set and it returns } e_\emptyset \text{ from an empty set) } \rangle \\
 & e_\emptyset
\end{array}
$$

□

Proposition 4.6

let $\mathcal{N} = (\Phi, D)$ where $\Phi = \{\varphi \mid \varphi \in D_J \text{ for } J \subseteq I\}$. We have, $insert(\mathcal{N}, \varphi) = \mathcal{N}$

Proof.

$$
\begin{array}{ll}
 & insert((\Phi, D), \varphi) \\
= & \quad \langle \text{ Definition of } insert \ \rangle \\
 & (\Phi \cup \{\varphi\}, D) \\
= & \quad \langle \ \varphi \in \Phi \text{ from the definition of } \Phi \ \rangle \\
 & (\Phi, D) \\
= & \quad \langle \text{ Definition of } \mathcal{N} \ \rangle \\
 & \mathcal{N}
\end{array}
$$

□

F Analyzing a Policy Using the Prototype Tool and PVS

To verify a security policy, we need the protocol specification, the precondition, and the postcondition. The specification of the protocol $I_1; I_2; I_3; I_4$ that is provided to the tool is:

```
Init (Agent "C") (Agent "S") "ch1" "s"

Comm (Agent "C") (Agent "S") "s" ([], T)
  (ChooseExtract (Agent "C") ["mission"] (Know ([("officerID",["JohnDo"])],["officerID"])) "Sx"

Comm (Agent "S") (Agent "C") "s" ([("data", "country")], T)
  (ChooseExtract (Agent "S") ["data"] (Know ([("data",["France"])],["data"])) "Cx1"

Comm (Agent "S") (Agent "C") "s" ([("data", "company")], T)
  (ChooseExtract (Agent "S") ["data"] (Know ([("data",["AirFrance"])],["data"])) "Cx2"

Comm (Agent "S") (Agent "C") "s" ([("data", "employee")], T)
  (ChooseExtract (Agent "S") ["data"] (Know ([("data",["Manager"])],["data"])) "Cx3"

Update (Agent "C") (Know ([("officerID",["JohnDo"])],["officerID"]))
  (Combine (Variable "Cx3") (Combine (Variable "Cx1")
  (Combine (Variable "Cx2") (Know ([("officerID",["JohnDo"])],["officerID"])))))

Init (Agent "C") (Agent "A") "ch2" "t"

Comm (Agent "C") (Agent "A") "t" ([("officerID","data"), ("country","data"), ("company","data")], T)
  (ChooseExtract (Agent "C") ["officerID","country","company","mission"]
  (Know ([("officerID",["JohnDo"])],["officerID"])) "Ax1"

Assign (Agent "A") (Combine (Variable "Ax1") (Know ([("analyzedData",["Performance"])],["analyzedData"])))

Comm (Agent "A") (Agent "C") "t" ([], T)
  (ChooseExtract (Agent "A") ["analyzedData"] (Focus (Variable "Ax1") "mission")) "Cx4"

Update (Agent "C") (Know ([("officerID",["JohnDo"])],["officerID"]))
  (Combine (Variable "Cx4") (Know ([("officerID",["JohnDo"])],["officerID"])) )

Init (Agent "C") (Agent "PR") "ch3" "v"

Comm (Agent "PR") (Agent "C") "v" ([], T) (ChooseExtract (Agent "PR") ["topic"]
  (Know ([("topic",["Economy"])],["topic"])) "Cx5"

Comm (Agent "C") (Agent "PR") "v" ([("country", "data")], T)
  (ChooseExtract (Agent "C") ["mission","country"] (Focus (Variable "Cx1") "topic")) "Px1"

Update (Agent "PR") (Know ([("topic",["Economy"])],["topic"]))
  (Combine (Variable "Px1") (Know ([("topic",["Economy"])],["topic"])) )

Init (Agent "PR") (Agent "A") "ch4" "u"

Comm (Agent "PR") (Agent "A") "u" ([], T)
  (ChooseExtract (Agent "PR") ["mission"] (Know ([("topic",["economy"])],["topic"])) "Ax2"

Comm (Agent "A") (Agent "PR") "u" ([], T)
  (ChooseExtract (Agent "A") ["data"] (Focus (Variable "Cx1") "mission")) "Px2"

Update (Agent "PR") (Know ([("topic",["Economy"])],["topic"]))
  (Combine (Variable "Px2") (Know ([("topic",["Economy"])],["topic"])))
```

```
Inaction
```

This postcondition verifies whether the Agent (PR) knows a piece of information classified as data is also classified as company in the Agent (CA)

```
p (Agent "C") ["company"]  "company" "data" (Agent "PR")
```

The first parameter of the postcondition is the predicate name. The second parameter is the agent name. The third parameter is the classification of the information. The fourth and fifth parameter represents the parameters of frame substitution. The last parameter is the agent name. The postcondition states that there exists an information classified as company is Agent C is also exists in Agent PR after substituting the frame company by data.

The tool computes the weakest precondition automatically and writes using the syntax of PVS language as shown below.

```
p
(
 update
 (
  update
  (
    NC,
    ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
      ELSE  s1 |false  ENDIF)), add(officerID,emptyset)) ,
    combineInf
    (
      fs(select(extract
       (
         NO,
         add(data,emptyset),
         ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
           ELSE  s1 |false  ENDIF) ),add(data,emptyset))
       )) ,data,employee) ,
      combineInf
      (
        fs(select(extract
        (
          NO,
          add(data,emptyset),
          ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
            ELSE  s1 |false  ENDIF)),add(data,emptyset))
        )) ,data,country) ,
        combineInf
        (
           fs(select(extract
           (
             NO,
             add(data,emptyset),
             ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
               ELSE s1 |false  ENDIF) ),add(data,emptyset))
            )) ,data,company) ,
           ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
             ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
        )
      )
    )
```

```
) ,
((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
  ELSE  s1 |false  ENDIF) ),add(officerID,emptyset)),
combineInf(select(extract
(
  insert
  (
    NA,
    combineInf
    (
      fs(fs(fs(select(extract
      (
        update
        (
          NC,
          ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
            ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
          combineInf
          (
            fs(select(extract
            (
              NO,
              add(data,emptyset),
             ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
                ELSE s1 |false  ENDIF) ),add(data,emptyset))
            )) ,data,employee) ,
            combineInf
            (
              fs(select(extract
              (
                NO,
                add(data,emptyset),
                ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
                  ELSE s1 |false  ENDIF) ),add(data,emptyset))
              )) ,data,country) ,
              combineInf
              (
                fs(select(extract
                (
                  NO,
                  add(data,emptyset),
                  ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
                     ELSE s1 |false  ENDIF) ),add(data,emptyset))
                )) ,data,company) ,
                ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                  ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
              )
            )
          )
        ) ,
        add(officerID,add(country,add(company,add(mission,emptyset)))),
        ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
          ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
        )) ,company,data) ,country,data) ,officerID,data) ,
        ((lambda i:(if member(i,add(analyzedData,emptyset)) THEN add(Performance,emptyset)
          ELSE s1 |false  ENDIF) ),add(analyzedData,emptyset))
      )
    ) ,
    add(analyzedData,emptyset),
    restrictInf(fs(fs(fs(select(extract
    (
      update
      (
        NC,
       ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
```

```
            ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
          combineInf
          (
             fs(select(extract
             (
               NO,
               add(data,emptyset),
               ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
                 ELSE s1 |false  ENDIF) ),add(data,emptyset))
             )) ,data,employee) ,
             combineInf
             (
               fs(select(extract
               (
                 NO,
                 add(data,emptyset),
                 ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
                   ELSE s1 |false  ENDIF) ),add(data,emptyset))
               )) ,data,country) ,
               combineInf
               (
                 fs(select(extract
                 (
                   NO,
                   add(data,emptyset),
                   ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
                     ELSE  s1 |false  ENDIF) ),add(data,emptyset))
                 )) ,data,company) ,
                 ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                     ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
               )
             )
           )
         ) ,
         add(officerID,add(country,add(company,add(mission,emptyset)))),
         ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
           ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
       )) ,company,data) ,country,data) ,officerID,data) ,add(mission,emptyset))
    )) ,
    ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
      ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
  )
) ,

add(company,emptyset),

update
(
  update
  (
    NPR,
    ((lambda i:(if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
      ELSE s1 |false  ENDIF) ),add(topic,emptyset)) ,
    combineInf
    (
      fs(select(extract
      (
        update
        (
          update
          (
            NC,
            ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
              ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
            combineInf
```

```
  (
    fs(select(extract
    (
      NO,
      add(data,emptyset),
      ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
        ELSE s1 |false  ENDIF) ),add(data,emptyset))
    )) ,data,employee) ,
    combineInf
    (
      fs(select(extract
      (
         NO,
         add(data,emptyset),
         ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
           ELSE s1 |false  ENDIF) ),add(data,emptyset))
      )) ,data,country) ,
      combineInf
      (
         fs(select(extract
         (
            NO,
            add(data,emptyset),
            ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
              ELSE s1 |false  ENDIF) ),add(data,emptyset))
         )) ,data,company) ,
         ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
            ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
      )
    )
  )
) ,
((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
  ELSE s1 |false  ENDIF) ),add(officerID,emptyset)),
combineInf
(
  select(extract
  (
     insert
     (
        NA,
        combineInf
        (
           fs(fs(fs(select(extract
           (
              update
              (
                NC,
                ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                  ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
                combineInf
                (
                   fs(select(extract
                   (
                      NO,
                      add(data,emptyset),
                     ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
                       ELSE s1 |false  ENDIF) ),add(data,emptyset))
                   )) ,data,employee) ,
                   combineInf
                   (
                     fs(select(extract
                     (
                        NO,
                        add(data,emptyset),
```

```
                    ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
                      ELSE s1 |false  ENDIF) ),add(data,emptyset))
                )) ,data,country) ,
                combineInf
                (
                  fs(select(extract
                  (
                    NO,
                    add(data,emptyset),
                    ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
                      ELSE s1 |false  ENDIF) ),add(data,emptyset))
                  )) ,data,company) ,
                 ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                   ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
                )
              )
            )
        ) ,
        add(officerID,add(country,add(company,add(mission,emptyset)))),
        ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
          ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
      )) ,company,data) ,country,data) ,officerID,data) ,
     ((lambda i:(if member(i,add(analyzedData,emptyset)) THEN add(Performance,emptyset)
       ELSE s1 |false  ENDIF) ),add(analyzedData,emptyset))
  )
) ,
add(analyzedData,emptyset),
restrictInf(fs(fs(fs(select(extract
(
  update
  (
     NC,
     ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
       ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
     combineInf
     (
       fs(select(extract
       (
         NO,
         add(data,emptyset),
         ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
           ELSE s1 |false  ENDIF) ),add(data,emptyset))
       )) ,data,employee) ,
       combineInf
       (
         fs(select(extract
         (
           NO,
           add(data,emptyset),
           ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
             ELSE s1 |false  ENDIF) ),add(data,emptyset))
         )) ,data,country) ,
         combineInf
         (
           fs(select(extract
           (
             NO,
             add(data,emptyset),
             ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
               ELSE s1 |false  ENDIF) ),add(data,emptyset))
           )) ,data,company) ,
           ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
             ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
         )
       )
```

```
                  )
                ) ,
               add(officerID,add(country,add(company,add(mission,emptyset)))),
               ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                 ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
             )) ,company,data) ,country,data) ,officerID,data) ,add(mission,emptyset))
           )
         ) ,
         ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
           ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
       )
    ) ,
    add(mission,add(country,emptyset)),
    restrictInf(fs(select(extract
     (
       NO,
       add(data,emptyset),
       ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
         ELSE s1 |false  ENDIF) ),add(data,emptyset))
        )) ,data,country) ,add(topic,emptyset))
    )) ,country,data) ,
    ((lambda i:(if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
       ELSE s1 |false  ENDIF) ),add(topic,emptyset))
   )
 ) ,
((lambda i:(if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
  ELSE s1 |false  ENDIF) ),add(topic,emptyset)) ,
combineInf
(
  select(extract
  (
    insert
    (
      NA,
      combineInf
      (
         fs(fs(fs(select(extract
         (
           update
           (
             NC,
             ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
               ELSE s1 |false  ENDIF) ),add(officerID,emptyset)) ,
             combineInf
             (
                fs(select(extract
                (
                  NO,
                  add(data,emptyset),
                  ((lambda i:(if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
                     ELSE s1 |false  ENDIF) ),add(data,emptyset))
                )) ,data,employee) ,
                combineInf
                (
                  fs(select(extract
                  (
                    NO,
                    add(data,emptyset),
                    ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
                     ELSE s1 |false  ENDIF) ),add(data,emptyset))
                  )) ,data,country) ,
                  combineInf
                  (
                    fs(select(extract
                    (
```

```
                          NO,
                          add(data,emptyset),
                          ((lambda i:(if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
                            ELSE s1 |false  ENDIF) ),add(data,emptyset))
                        )) ,data,company) ,
                        ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                          ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
                      )
                    )
                  )
                ) ,
                add(officerID,add(country,add(company,add(mission,emptyset)))),
               ((lambda i:(if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
                 ELSE s1 |false  ENDIF) ),add(officerID,emptyset))
            )) ,company,data) ,country,data) ,officerID,data) ,
            ((lambda i:(if member(i,add(analyzedData,emptyset)) THEN add(Performance,emptyset)
              ELSE s1 |false  ENDIF) ),add(analyzedData,emptyset))
          )
        ) ,
        add(data,emptyset),
        restrictInf(fs(select(extract
        (
          NO,
          add(data,emptyset),
          ((lambda i:(if member(i,add(data,emptyset)) THEN add(France,emptyset)
            ELSE s1 |false  ENDIF) ),add(data,emptyset))
        )) ,data,country) ,add(mission,emptyset))
       )) ,
      ((lambda i:(if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
        ELSE s1 |false  ENDIF) ),add(topic,emptyset))
    )
  ) ,

  company,data
)
```

The precondition is a predicate on the initial knowledge of agent. We write the precondition using the syntax of PVS language:

```
isInKnow
(
   NC,
   add(officerID,add(mission,add(topic,emptyset))),
   ((lambda i: (if member(i,add(mission,emptyset)) THEN add(Cobra,emptyset)
      ELSE (if member(i,add(officerID,emptyset)) THEN add(JohnDo,emptyset)
      ELSE (if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
      ELSE  s1 |false ENDIF) ENDIF) ENDIF)), add(mission,add(officerID,add(topic,emptyset))))
)
AND
isInKnow
(
  NO,
  add(mission,add(data,emptyset)),
  ((lambda i: (if member(i,add(mission,emptyset)) THEN add(Cobra,emptyset)
     ELSE (if member(i,add(data,emptyset)) THEN add(France,emptyset)
     ELSE  s1 |false ENDIF) ENDIF)), add(mission,add(data,emptyset)))
)
AND
isInKnow
(
  NO,
  add(mission,add(data,emptyset)),
  ((lambda i: (if member(i,add(mission,emptyset)) THEN add(Cobra,emptyset)
```

```
     ELSE (if member(i,add(data,emptyset)) THEN add(Manager,emptyset)
     ELSE  s1 |false ENDIF) ENDIF)), add(mission,add(data,emptyset)))
)
AND
isInKnow
(
  NO,
  add(mission,add(data,emptyset)),
  ((lambda i: (if member(i,add(mission,emptyset)) THEN add(Cobra,emptyset)
     ELSE (if member(i,add(data,emptyset)) THEN add(AirFrance,emptyset)
     ELSE  s1 |false ENDIF) ENDIF)), add(mission,add(data,emptyset)))
)
AND
isInKnow
(
  NPR,
  add(topic,emptyset),
  ((lambda i: (if member(i,add(topic,emptyset)) THEN add(Economy,emptyset)
     ELSE  s1 |false ENDIF)), add(topic,emptyset))
)
AND PROJ_2(NC)=add(topic,add(mission,add(officerID,add(country,
               add(company,add(employee,add(analyzedData,emptyset)))))))
AND PROJ_2(NO)=add(mission,add(officerID,emptyset))
AND PROJ_2(NPR)=add(mission,add(data,emptyset))
AND PROJ_2(NA)= add(mission,add(data,add(analyzedData,emptyset)))
```

We develop the following theory to be used in proving $p \Rightarrow wp(S, Q)$

```
LatticeM: THEORY

BEGIN
I : TYPE+ = {private, officerID, data, mission, topic, analyzedData, country, company, employee}
frames: TYPE+ = setof[I]
d1,d2: VAR frames

framejoin(d1,d2):frames = union(d1,d2)
framemeet(d1,d2):frames = intersection(d1,d2)
framelower(d1,d2):bool = subset?(d1,d2)
frameeq(d1,d2):bool = subset?(d1,d2) AND subset?(d2,d1)

END LatticeM


InformationAlgebraModel: THEORY

BEGIN
IMPORTING LatticeM
st: TYPE+ = {France, AirFrance, Manager, Economy, Performance, JohnDo, Cobra}
information: TYPE+ = [I -> setof[st]]
know: TYPE+ = [information, frames]


inf1,inf2,inf3: VAR know
i,j: VAR I
d1,d2: VAR frames
s: string
s1: VAR st

topFrame: frames = {i | TRUE}

restrictInf(inf1,d1): know  =  (lambda i: (if member(i,d1) THEN PROJ_1(inf1)(i)
                                ELSE  s1 |false ENDIF),framemeet(d1,PROJ_2(inf1)))
```

```
infdom(inf1):frames = PROJ_2(inf1)

emptyInf(d1):know = (lambda i: {s1|false},d1)

combineInf(inf1,inf2) : know  = ((lambda i:
  (if member(i,PROJ_2(inf1)) AND member(i,PROJ_2(inf2))  THEN union(PROJ_1(inf1)(i),PROJ_1(inf2)(i))
  ELSE (if member(i,PROJ_2(inf1)) AND NOT member(i,PROJ_2(inf2)) THEN PROJ_1(inf1)(i)
  ELSE (if member(i,PROJ_2(inf2)) AND NOT member(i,PROJ_2(inf1)) THEN PROJ_1(inf2)(i)
  ELSE s1 |false ENDIF) ENDIF) ENDIF)) ,union(PROJ_2(inf1),PROJ_2(inf2)))

inflower(inf1,inf2): bool =
  (FORALL (i):  member(i,PROJ_2(inf1)) IMPLIES subset?(PROJ_1(inf1)(i),PROJ_1(inf2)(i)))
   AND (subset?(PROJ_2(inf1),PROJ_2(inf2)))


rmv(inf1,inf2) : know  = ((lambda i:
  (if member(i,PROJ_2(inf1)) AND member(i,PROJ_2(inf2))  THEN difference(PROJ_1(inf1)(i),PROJ_1(inf2)(i))
  ELSE (if member(i,PROJ_2(inf1))  THEN PROJ_1(inf1)(i)
  ELSE s1 |false ENDIF)  ENDIF)) ,PROJ_2(inf1))


END InformationAlgebraModel


ExplicitKnow: THEORY
BEGIN
IMPORTING InformationAlgebraModel
N: TYPE+ = setof[know]
T: TYPE+ = [N,frames]
t1,t2, NC,NA,NPR,NO : VAR T
n1,n2: VAR N
d1,d2,d3,d4: VAR frames
inf1, inf2, inf3, inf4, inf5: VAR know
i,j,k, i1: VAR I
s1: VAR st



update(t1,inf1,inf2): T =
  if (subset?(infdom(inf2),PROJ_2(t1))) THEN (inf4 | EXISTS(inf3) : member(inf3,PROJ_1(t1)) AND
  ((inflower(inf1,inf3) AND inf4=combineInf(rmv(inf3,inf1),inf2)) OR
  (NOT inflower(inf1,inf3) AND inf4=inf3)),PROJ_2(t1)) ELSE t1 ENDIF


extract(t1,d1,inf1): N = {inf3 | EXISTS(inf2): subset?(d1,PROJ_2(t1)) AND
  member(inf2,PROJ_1(t1)) AND inflower(inf1,inf2) AND
  framelower(d1,infdom(inf2)) AND inf3=restrictInf(inf2,d1)}


isInKnow(t1,d1,inf1): bool = EXISTS (inf2):
  subset?(d1,PROJ_2(t1)) AND member(inf2,PROJ_1(t1)) AND
  inflower(inf1,inf2) AND framelower(d1,infdom(inf2))


insert(t1,inf1):T =
  if (subset?(infdom(inf1),PROJ_2(t1))) THEN (add(inf1,PROJ_1(t1)),PROJ_2(t1))
  ELSE t1 ENDIF


select(n1):know =
  if (nonempty?(n1)) then choose(n1) ELSE emptyInf(emptyset) ENDIF


fs1(inf1,i,j): know = ((lambda k:(if (k=j) THEN PROJ_1(inf1)(i) ELSE emptyset ENDIF)),i1|i1=j)
```

```
fs(inf1,i,j): know = combineInf(rmv(inf1,restrictInf(inf1,k|k=i)),fs1(restrictInf(inf1,k|k=i),i,j))

ds(d1,i,j): frames = k | EXISTS(i1): member(i1,d1) AND ((NOT i1=i AND k=i1) OR (i1=i AND k=j))

p(t1,d1,t2,i,j): bool =
  EXISTS(inf1): infdom(inf1)=d1 AND NOT(inflower(inf1,emptyInf(d1))) AND
  isInKnow(t2,infdom(fs(inf1,i,j)),fs(inf1,i,j)) AND  isInKnow(t1,d1,inf1)

p1(t1,d1,inf2): bool =
  EXISTS(inf1): infdom(inf1)=d1 AND NOT(inflower(inf1,emptyInf(d1))) AND isInKnow(t1,d1,inf1)

p2(inf2,d1,t2,i,j): bool =
  EXISTS(inf1): infdom(inf1)=d1 AND NOT(inflower(inf1,emptyInf(d1))) AND inflower(inf1,inf2)
  AND isInKnow(t2,infdom(fs(inf1,i,j)),fs(inf1,i,j))

p3(inf2,d1,inf3,i,j):bool =
  EXISTS(inf1): infdom(inf1)=d1 AND NOT(inflower(inf1,emptyInf(d1))) AND
  inflower(inf1,inf2) AND inflower(fs(inf1,i,j),inf3)


inflem1:  LEMMA inf1=inf2 IFF (inflower(inf1,inf2) and inflower(inf2,inf1))

inflem2: LEMMA d1=d2 IFF framelower(d1,d2) AND framelower(d2,d1)

inflem3: LEMMA combineInf(inf1,inf2) = combineInf(inf2,inf1)

inflem4: LEMMA combineInf(combineInf(inf1,inf2),inf3)=combineInf(inf1,combineInf(inf2,inf3))

inflem5: LEMMA restrictInf(inf1,topFrame)=inf1

inflem6:LEMMA restrictInf(inf1,infdom(inf1))=inf1

inflem7: LEMMA inflower(inf1, combineInf(inf1,inf2))

inflem8:LEMMA inflower(inf2, combineInf(inf1,inf2))

inflem9:LEMMA infdom(combineInf(inf1,inf2))= framejoin(infdom(inf1),infdom(inf2))

inflem10:LEMMA framelower(infdom(select(extract(t1,d1,inf1))),d1)

inflem11: LEMMA framelower(d1,infdom(inf1)) IMPLIES infdom(restrictInf(inf1,d1))=d1

inflem12: LEMMA framelower(d1,infdom(inf1)) IMPLIES subset?(infdom(restrictInf(inf1,d1)),d1)

inflem13: LEMMA framelower(d1,infdom(inf1)) IMPLIES subset?(d1,infdom(restrictInf(inf1,d1)))

inflem14: LEMMA framelower(infdom(inf1), infdom(combineInf(inf1,inf2)))

inflem15: LEMMA infdom(combineInf(inf1,inf2))=framejoin(infdom(inf1),infdom(inf2))

inflem16: LEMMA framelower(d1, framejoin(d1,d2))

inflem17: LEMMA framejoin(d1,d2)=framejoin(d2,d1)

inflem18: LEMMA member(inf1,PROJ_1(t1)) IMPLIES subset?(infdom(inf1),PROJ_2(t1))

inflem19: LEMMA intersection(d1,d2)=intersection(d2,d1)

inflem20: LEMMA inflower(inf2, inf3) AND inflower(inf1, inf3) AND
  inf4 = combineInf(rmv(inf3, inf1), inf2) IMPLIES inflower(inf4, inf3)

inflem21: LEMMA inflower(inf1,inf2) AND inflower(inf2,inf3) IMPLIES inflower(inf1,inf3)

inflem22: LEMMA inflower(inf1,rmv(inf2,inf3)) IMPLIES inflower(inf1,inf2)
```

```
inflem23:LEMMA inflower(restrictInf(rmv(combineInf(inf1,inf2),inf2),infdom(inf1)),inf1)

inflem24: LEMMA member(i,infdom(inf1)) IMPLIES inflower(fs(fs(inf1,i,j),j,i),inf1)

inflem25: LEMMA member(i,infdom(inf1)) IMPLIES inflower(inf1,fs(fs(inf1,i,j),j,i))

inflem26: LEMMA NOT member(j,infdom(inf1)) IMPLIES  inflower(fs(fs(inf1,i,j),j,i),inf1)

inflem27: LEMMA NOT member(j,infdom(inf1)) IMPLIES  inflower(inf1,fs(fs(inf1,i,j),j,i))

inflem28:  LEMMA NOT member(j,infdom(inf1)) IMPLIES  inf1= fs(fs(inf1,i,j),j,i)

inflem29: LEMMA subset?(infdom(inf1),infdom(inf2)) AND
  subset?(infdom(inf2),infdom(inf1)) IMPLIES (inflower(fs(inf1,i,j),fs(inf2,i,j)) IMPLIES
  inflower(inf1,inf2))

inflem30: LEMMA subset?(infdom(inf1),infdom(inf2)) AND
  subset?(infdom(inf2),infdom(inf1)) IMPLIES (inflower(fs(inf1,i,j),fs(inf2,i,j)) AND
  inflower(fs(inf2,i,j),fs(inf1,i,j)) IMPLIES
  (inflower(inf1,inf2) AND inflower(inf2,inf1)))

inflem31: LEMMA inflower(inf1,inf2) IMPLIES inflower(inf1,combineInf(inf2,inf3))

inflem32: LEMMA rmv(emptyInf(d1),inf1)=emptyInf(d1)

inflem33: LEMMA rmv(inf1,emptyInf(d1))=inf1

inflem34: LEMMA inflower(inf1,inf2) IMPLIES
  combineInf(rmv(inf3,inf1),inf2)=combineInf(inf3,inf2)

inflem35: LEMMA inflower(inf1, combineInf(inf2,inf3)) AND inf1=rmv(inf1,inf2) AND
  (NOT EXISTS (i): subset?(PROJ_1(inf1)(i),emptyset)) IMPLIES inflower(inf1,inf3)

inflem36: LEMMA inflower(inf1,inf2) IFF combineInf(inf1,inf2)=inf2

inflem37: LEMMA combineInf(rmv(inf1,inf2),inf2)=combineInf(inf1,inf2)

inflem38: LEMMA inflower(inf1,inf2) IMPLIES rmv(inf1,inf2)=emptyInf(infdom(inf1))

inflem39: LEMMA inflower(inf1,inf2) AND inflower(inf1,inf3) IMPLIES
  subset?(PROJ_1(update(update(t1,inf1,inf2),inf1,inf3)),
          PROJ_1(update(t1,inf1,combineInf(inf2,inf3))))

inflem40: LEMMA inflower(inf1,inf2) AND inflower(inf1,inf3) IMPLIES
  subset?(PROJ_1(update((n1,d1),inf1,combineInf(inf2,inf3))),
          PROJ_1(update(update((n1,d1),inf1,inf2),inf1,inf3)))

inflem41: LEMMA empty?(PROJ_1(t1)) IMPLIES subset?(PROJ_1(update(t1,inf1,inf2)),PROJ_1(t1))

inflem42: LEMMA empty?(n1) IMPLIES subset?(n1,PROJ_1(update(t1,inf1,inf2)))

inflem43: LEMMA NOT isInKnow(t1,infdom(inf1), inf1) IMPLIES (update(t1,inf1,inf2)=t1)

inflem44: LEMMA NOT isInKnow(t1,infdom(inf1),inf1) IMPLIES
  subset?(PROJ_1(update(t1,inf1,inf2)),PROJ_1(t1))

inflem45: LEMMA NOT isInKnow(t1,infdom(inf1),inf1) IMPLIES
  subset?(PROJ_1(t1),PROJ_1(update(t1,inf1,inf2)))

inflem46: LEMMA NOT empty?(extract(t1,d1,inf1)) IMPLIES
  isInKnow(t1,d1,select(extract(t1,d1,inf1)))

inflem47: LEMMA isInKnow(t1,infdom(inf1),inf1) AND inflower(inf3,inf2) AND
  framelower(d1,infdom(inf2)) IMPLIES infisInKnow(update(t1,inf1,inf2),d1,inf3)
```

```
inflem48: LEMMA isInKnow(t1,infdom(inf1),inf1) AND isInKnow(t1,d1,inf3) AND
  inflower(inf1,inf2) IMPLIES isInKnow(update(t1,inf1,inf2),d1,inf3)

inflem49: LEMMA inflower(inf3,inf2) AND isInKnow(t1, infdom(inf1), inf1) IMPLIES
  isInKnow(update(t1,inf1,inf2), infdom(inf3), inf3)

inflem50: LEMMA inflower(inf1, combineInf(rmv(inf2,inf3),inf4)) AND
  subset?(intersection(infdom(inf1),infdom(inf4)),emptyset) IMPLIES inflower(inf1,inf2)

inflem51: LEMMA isInKnow(update(t1,inf1,inf2),d1, inf3) AND
  subset?(intersection(infdom(inf3),infdom(inf2)),emptyset) AND
  subset?(intersection(d1,infdom(inf2)),emptyset) IMPLIES isInKnow(t1,d1,inf3)

inflem52: LEMMA isInKnow(insert(t1,inf2),d1,inf1) AND
  (NOT inflower(inf1,inf2) OR NOT framelower(d1,infdom(inf2))) IMPLIES isInKnow(t1,d1,inf1)

inflem53: LEMMA isInKnow(t1,infdom(inf1), inf1) AND inflower(inf3,inf2) AND
  framelower(d2,infdom(inf2)) AND subset?(infdom(inf2),PROJ_2(t1)) IMPLIES
  NOT empty?(extract(update(t1,inf1,inf2),d2,inf3))


policylem1: LEMMA NOT empty?(extract(t1,d1,inf2)) AND NOT member(j,d1) AND
  isInKnow(t2,infdom(inf1),inf1) IMPLIES
  p(t1,d1,update(t2,inf1,fs(select(extract(t1,d1,inf2)),i,j)),i,j)

policylem2: LEMMA p(update(t1,inf1,inf2),d1,t2,i,j) AND
  subset?(intersection(infdom(inf2),d1),emptyset) IMPLIES p(t1,d1,t2,i,j)

policylem3: LEMMA p(t1,d1,update(t2,inf1,inf2),i,j) AND
  subset?(intersection(infdom(inf2),ds(d1,i,j)),emptyset) IMPLIES p(t1,d1,t2,i,j)

policylem4: LEMMA p(update(t1,inf1,inf2),d1,t2,i,j) AND
  (FORALL(inf3): infdom(inf3)=d1 AND isInKnow(t1, ds(d1,i,j), fs(inf3,i,j)) AND
  inf3=rmv(inf3,inf2) AND NOT (EXISTS(i): subset?(PROJ_1(inf3)(i),emptyset))) IMPLIES
  p(t1,d1,t2,i,j)

policylem5: LEMMA p(update(t1,inf1,inf2),d1,t2,i,j) AND
  (FORALL(inf3): isInKnow(t2, infdom(inf3), inf3) AND inf3=rmv(inf3,fs(inf2,i,j)) AND
  NOT (EXISTS(k): subset?(PROJ_1(inf3)(k),emptyset))) IMPLIES p(t1,d1,t2,i,j)

policylem6: LEMMA p(t1,d1,insert(t2,inf1),i,j) AND
  subset?(intersection(infdom(inf1),ds(d1,i,j)),emptyset) IMPLIES p(t1,d1,t2,i,j)

policylem7: LEMMA p(insert(t1,inf1),d1,t2,i,j) AND
  subset?(intersection(infdom(inf1),d1),emptyset) IMPLIES p(t1,d1,t2,i,j)

policylem8: LEMMA  NOT empty?(extract(t1,d1,inf3)) AND
  (FORALL (inf4): member(inf4,extract(t1,d1,inf3)) IMPLIES NOT inflower(inf4,emptyInf(d1))) AND
  inflower(fs(select(extract(t1,d1,inf3)),i,j),inf2) AND isInKnow(t2,infdom(inf1),inf1) IMPLIES
  p(t1,d1,update(t2,inf1,inf2),i,j)

policylem9: LEMMA  isInKnow(t1,d1,restrictInf(fs(inf2,j,i),d1)) AND d1=infdom(fs(inf2,j,i)) AND
  isInKnow(t2,infdom(inf1),inf1) AND NOT inflower(restrictInf(fs(inf2,j,i),d1),emptyInf(d1)) AND
  NOT member(i,infdom(inf2)) AND framelower(infdom(inf2),PROJ_2(t2)) AND infdom(inf2)=ds(d1,i,j)
  IMPLIES  p(t1,d1,update(t2,inf1,inf2),i,j)

policylem10: LEMMA  p1(t1,d1,inf2) AND isInKnow(t2,infdom(inf1),inf1) AND
  NOT member(i,infdom(inf2)) AND framelower(infdom(inf2),PROJ_2(t2)) AND
  infdom(inf2)=ds(d1,i,j) IMPLIES  p(t1,d1,update(t2,inf1,inf2),i,j)

policylem11: LEMMA isInKnow(t1,infdom(inf1),inf1) AND
  framelower(infdom(inf2),PROJ_2(t1)) AND p2(inf2,d1,t2,i,j) IMPLIES
  p(update(t1,inf1,inf2),d1,t2,i,j)

policylem12: LEMMA p2(inf1,d1,t2,i,j) IMPLIES  p2(combineInf(inf1,inf2),d1,t2,i,j)
```

```
policylem13: LEMMA p2(inf2,d1,t2,i,j) IMPLIES  p2(combineInf(inf1,inf2),d1,t2,i,j)

policylem14: LEMMA isInKnow(t1,infdom(inf2), inf2) AND
  framelower(infdom(inf3),PROJ_2(t1)) AND p3(inf1,d1,inf3,i,j) IMPLIES
  p2(inf1,d1,update(t1,inf2,inf3),i,j)

policylem15: LEMMA p3(inf1,d1,inf2,i,j) IMPLIES p3(inf1,d1,combineInf(inf2,inf3),i,j)

policylem16: LEMMA p3(inf1,d1,inf3,i,j) IMPLIES p3(inf1,d1,combineInf(inf2,inf3),i,j)

policylem17: LEMMA inflower(inf3,inf2) AND framelower(d2,infdom(inf2)) AND
  subset?(extract(t1,d2,inf3),emptyset) AND framelower(infdom(inf2),PROJ_2(t1)) AND
  p3(inf1,d1,restrictInf(inf2,d2),i,j) IMPLIES p3(inf1,d1,select(extract(insert(t1,inf2),d2,inf3)),i,j)

policylem18: LEMMA  subset?(intersection(infdom(inf1),d1),emptyset) AND
  subset?(intersection(infdom(inf2),d1),emptyset) AND p(t1,d1,t2,i,j) IMPLIES
  p(update(t1,inf1,inf2),d1,t2,i,j)

policylem19: LEMMA subset?(intersection(infdom(inf1),ds(d1,i,j)),emptyset) AND
  subset?(intersection(infdom(inf2),ds(d1,i,j)),emptyset) AND p(t1,d1,t2,i,j) IMPLIES
  p(t1,d1,update(t2,inf1,inf2),i,j)

policylem20: LEMMA isInKnow(t1,infdom(inf1),inf1) AND subset?(infdom(inf2), PROJ_2(t1)) AND
  framelower(d1,infdom(inf2)) AND NOT inflower(restrictInf(inf2,d1),emptyInf(d1)) AND
  isInKnow(t2,ds(d1,i,j),fs(restrictInf(inf2,d1),i,j)) IMPLIES  p(update(t1,inf1,inf2),d1,t2,i,j)
```

References

[1] Khaled Alghathbar, Csilla Farkas, and Duminda Wijesekera. Securing UML information flow using FlowUML. *Journal of Research and Practice in Information Technology*, 38(1):111–120, February 2006.

[2] D.E. Bell and L.J. La Padula. Secure computer system: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation, March 1976.

[3] David F.C. Brewer and Michael J. Nash. The chinese wall security policy. In *IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[4] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured Communication-Centred Programming for Web Services. In Rocco De Nicola, editor, *Programming Languages and Systems*, volume 4421 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 2007.

[5] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge university press, second edition, 2002.

[6] Riccardo Focardi and Roberto Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, September 1997.

[7] W3C WS-CDL Working Group. Web services choreography description language version 1.0. http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/ accessed on September 2007.

[8] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[9] Katia Hristova, Tom Rothamel, Yanhong A. Liu, and Scott D. Stoller. Efficient type inference for secure information flow. In *PLAS '06: Proceedings of the 2006 workshop on Programming languages and analysis for security*, pages 85–94, New York, NY, USA, 2006. ACM.

[10] Naoki Kobayashi. Type-based information flow analysis for the π-calculus. *Acta Informatica*, 42(4):291–347, 2005.

[11] Jürg Kohlas and Robert F. Stärk. Information Algebras and Consequence Operators. *Logica Universalis*, 1(1):139–165, January 2007.

[12] Khair Eddin Sabri and Ridha Khedri. A mathematical framework to capture agent explicit knowledge in cryptographic protocols. Technical Report CAS-07-04-RK, department of Computing and Software, Faculty of Engineering, McMaster University, 2007. http://www.cas.mcmaster.ca/cas/0template1.php?601 (accessed on January 19, 2008).

[13] Khair Eddin Sabri, Ridha Khedri, and Jason Jaskolka. Specification of agent explicit knowledge in cryptographic protocols. In *CESSE 2008 : International Conference on Computer, Electrical, and Systems Science, and Engineering*, volume 35, Venice, Canada, October 2008. World Academy of Science, Engineering and Technology.

[14] Vijay Varadharajan. Petri net based modelling of information flow security requirements. In *Computer Security Foundations Workshop III*, pages 51–61, 1990.

[15] Dennis Volpano, Cynthia Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *J. Comput. Secur.*, 4(2-3):167–187, 1996.