FORMALIZING TRUST REQUIREMENTS AND SPECIFICATION IN SERVICE WORKFLOW ENVIRONMENTS

Wattana Viriyasitavat[1,2] and Andrew Martin[1]
[1]*Computing Laboratory, Department of Computer Science, University of Oxford, Oxford, U.K.*

[2]*Information Technology Division, Department of Statistics, Chulalongkorn University, Bangkok, Thailand*

Keywords: Trust, Service, Workflows, Trust, Requirements, Specification, Formalism.

Abstract: The emergence of advance communication technologies such as Internet has changed the nature of face-to-face towards virtual interactions in the form of services. Proliferation of services has enabled the creation of new value-added services composed of several sub-services in a pre-specified manner, known as service workflows. There are a number of security issues as workflows require disparate services to dynamically collaborate and interact on demand. Trust is an enabling technology serving as an adaptive and platform-independent solution that fits in this context. However, the lack of consensus on a unified trust definition and the traditional mindset of treating trust requirements separately pose the difficulty in developing formal specification. This paper provides a formal framework to this problem. The central part of the paper is logic based formalism with algebraic expressions to formally specify trust requirements. A trust definition and three modes of trust are described with algebraic operators to form specification formulae. The contribution of the framework is to allow trust requirements to be formally and uniformly specified by each distributed autonomous service, serving as a core component for automatic compliance checking in service workflows.

1 INTRODUCTION

To date, many trust approaches have been proposed (Viriyasitavat, 2009, Guha et al., 2004, Jøsang et al., 2006 Galizia et al., 2007); however they neither provide sufficient formal frameworks nor effective models in specifying trust. Moreover, the lack of consensus on a trust definition (O'Donovan and Smyth, 2005) poses the difficulty in developing formal specification. It impedes (1) workflow scalability that tends to be limited in a certain domain, (2) dynamicity when each service acts in an autonomous manner, and (3) consistency to deal with disparate trust requirements.

This paper attempts to fulfill the lack of the formal specification required by autonomous services to consistently and uniformly specify their requirements. Hence, the development of a formal trust specification (TS) and semantics for reasoning about trust and service workflows permit services to automatically check for compliance and therefore assess trustworthiness from this result. This paper makes four contributions. (1) Our TS is a means to formally and uniformly express trust requirements across multiple domains. (2) With well-defined semantics, TS serves as a key driver for compliance checking. Since trust requirements are uniformly expressed, there is no need to devise distinct complicated algorithms to deal with unrelated formats. (3) In rapidly changing environments which may affect the willingness of a service to participate in a workflow, our TS provides a flexible means by allowing services the ability to relax or restrict trust requirements on-the-fly, subjecting to their preference. And (4) our TS can be used to facilitate the service selection processes, which regulates how a workflow is constructed in order to maximize utility. As a result, the solution enriches the proliferation of service provisions and consumptions over the Internet.

2 REQUIREMENTS FOR TRUST IN SERVICE WORKFLOWS

This section presents the requirements our work aim to meet. We propose unique characteristics of workflows and trust as follows.

Viriyasitavat W. and Martin A..
FORMALIZING TRUST REQUIREMENTS AND SPECIFICATION IN SERVICE WORKFLOW ENVIRONMENTS .
DOI: 10.5220/0003479501960206
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 196-206
ISBN: 978-989-8425-55-3

R1: Interoperability with Local Security Requirements

In decentralized workflows, no single point of control exists; instead, the workflow specification itself travels from service to service (Kuntze and Sch, 2008). Access to local resources is typically determined by local security policies. Integrating or transforming independent local security policies into global policy enforcement is a very complicated task; especially, when services dynamically join or leave the workflow with free will. As such, it is desirable that security should be maintained locally and the requirements corresponding to local policies are specified in a uniformed format and published among participating services in a workflow.

R2: Separation of Duty (SoD)

Services with complementary competencies are joined to carry out a special task. SoD ensures that conflicting services cannot be part of a workflow execution to circumvent conflict-of-interest situations. Typically, a service is assigned to a specific role for a particular task, where SoD policies are verified based on role assignment (Nyanchama and Osborn, 1999, Jaeger and Tidswell, 2001). However, this method only permits workflow owners to enforce policies from their perspectives, but preventing participating services to express their SoD requirements. Our TS allows arbitrary services to impose their own requirements through TS formulae.

R3: The Association between Tasks and Services

As noted, several services with complementary capabilities can be gathered to perform a sophisticated task. On the other hand one service is also allowed to execute several tasks in a workflow. For example, an online patient record service is responsible for acquiring patients' information and providing statistical analysis to the National Health Service. Privacy and confidentiality are two requirements of the first while the later concerns with data accuracy. Consequently, the trust level of such service is differently determined depending on the task in responsible.

R4: Flow- and Task-oriented Property Specification

There are two perspectives of trust in workflow executions. The first is flow-oriented perspective describing that services involved in a flow execution must possess certain properties. For example, to protect data secrecy, digital patient's records must be transmitted over SSL among services involved along the flow. Another focuses on task-oriented executions. Properties of services responsible for a particular task can be verified before trusting. Since one task can be connected by many services, the properties of other services may have an influence on trust of a target service. For instance, the insurance claiming task must be approved by two different services, one from an insurance company and another from a contracted hospital. To trust a hospital, an insurance service must be presented.

R5: Enforcement of Sequences

Security of a workflow also depends on the sequence of tasks that are dependent on one another. One particular task might require the results from others occurred before, as well as to provide its results to the services afterwards (Kuntze and Sch, 2008). From the service viewpoint, it is desirable that they are able to specify properties related to the sequence of tasks and service associated. For example, the task of issuing a check for a tax refund can be done after a financial and general manager have approved in order.

R6: Flexible Degrees of Restriction

The absence of the end-to-end visibility of a workflow has led workflow research to re-examine and to find the way for workflow cooperation (Falcone et al., 2003). Flexible degrees of visibility enable entities to retain the level of privacy and confidentiality of internal processes. This fact gives rise to security difficulties for one to accurately specify properties of other services in a workflow. In response to this, TS should be flexible to formally express requirements with several degrees of restriction based on visibility.

R7: Protection of Workflow Data

Since data is traversed from one service to another in a workflow path, protecting the data against security threats becomes necessary. This requirement has been sufficiently accommodated by protections offered by traditional security. For example, integrity refers to the prevention of unauthorized modification; authentication refers to verifying identity accessing to information; authorization refers to access control enforcement; and confidentiality is achieved by the use of cryptography. These requirements are essential to be specified as the required properties where TS should be developed in a way to address this requirement.

3 RELATED WORK

Although trust has long been investigated, one topic which has been treated less is in the area of

formalism.

Table 1: Comparison between Trust Formalisms.

Models	R1	R2	R3	R4	R5	R6	R7
Marsh	×	×	×	×	×	✓/×	×
Davulcu	×	✓/×	×	✓	✓	✓	✓/×
Altunay	✓	×	×	✓/×	×	✓	✓
HEN+	×	×	×	×	×	×	×
Our Model	✓	✓	✓	✓	✓	✓	✓

✓ Direct Support, ✓/× Not Obvious, × Not Support

The following described the related works in this area (summarized in tables 1)

Marsh (Marsh, 1994) addresses key aspects of trust in providing the social sciences with a tool where the quantitative trust value, between -1 and 1, is used to support trusting decisions. This work is widely regarded as the first introduction of trust formalism. The notion $T_a(b, s)^t$ denotes 'a trust b' in situation s at a specific point of time t. The main contribution is to encompass several aspects including situation, time, utility, importance and knowledge to quantify trust, providing those who study on trust with a means of discussion in a precise manner. However, this work is inadequate to capture the important aspects of trust requirements in service workflows, as its original purpose is not focused on workflow collaboration.

Davulcu et al. (Davulcu et al., 1999) devises a framework based on Concurrent Transaction Logic (CTR) for reasoning in virtual enterprises. Workflow is modelled by Direct Acyclic Graph representing task coordination. A set of CTR connectives enforces constraints on workflow structure. Although these connectives are sufficient to specify workflow constraints on tasks coordination, they are less expressive in term of capturing trust requirements in service workflows. For instance, it is not possible to realize SoD (R2), and fails to capture the association between services and tasks (R3).

The earliest work of integrating trust in service workflow is presented by Altunay et al. (Altunay et al., 2005). Trust relationships are examined in two categories: direct and indirect. Direct trust relationships occur between two services that are immediate neighbour, whereas indirect trust relationship describes when two services are not immediately connected. However, this work is only applicable to the workflow modelled by a simple graph, where in the real world it is far more complex. It is neither formally expressed nor addressed essential trust requirements such as Enforcement of Sequence (R5).

In our previous work (Viriyasitavat, 2009), a petri-net-based trust framework called HENS+ is used to support inter-domain workflow trust relationship, addressing service delegation and trust transitivity in dynamic workflow environments. This work identifies the necessity of mutual trust relationships between interacting domains. Despite addressing comprehensive relationship at the domain level, it lacks a formal approach for specifying trust of interacting services inside a workflow.

Despite several approaches being proposed, they are incomplete as one might be appropriate for expressing trust in general aspects, and another might be able to reason. In this paper, the formal TS is developed to fulfil this lack by focusing on the lower level at inter-service trust relationship.

4 WORKFLOW MODELLING

Since Petri Net is widely-accepted as a mathematical workflow modelling (Van der Aalst, 1998, Salimifard and Wright, 2001, Klai and Tatam 2005, we generalize the Petri Net by adding a new set of logic-based connectives, which we call this variant as Service Workflow Net (SWN).

Def. 1: SWN is a labelled Place/Transition Net, i.e., a tuple $\mathcal{M} = (P, T, F, C, r, i, o, l)$ is a SWN iff: (1) it has two specific places, input place (i) and output place (o), and (2) if a new transition (t) is added to connect place o and place i, i.e., $\cdot t = \{o\}, t \cdot = \{i\}$, where

1. P represents places (services) of a workflow (circle),
2. T represents transitions(tasks) (rectangle),
3. P and $T \neq \emptyset$; and $P \cap T = \emptyset$,
4. $F \subseteq (P \times T) \cup (T \times P)$ represents directed flows,
5. $C = \{AND, OR, XOR\}$ is a set of connectives that is opened to support other advanced constructs.
6. $l: P \rightarrow A \cup \{\tau\}$ is a labeling function where A is a set of attributes, and τ denote a null value. It is used for labelling a service with associated attributes.

Compared to others like π-calculus (Van der Aalst, 2004) and UML activity diagrams (Eshuis and Wieringa, 2003), Petri Net provides advantages in modelling workflows (Van der Aalst, 1998, Best et al, 2001) However, the best choice for workflow modelling is ongoing arguments. Different languages have different advantages depending on which aspects being approached. The main reason of using Petri Net is that the powerful mathematical

foundation makes it possible to set up mathematical models for reasoning about TS. It contains a set of places and transitions corresponding to services and tasks which provides a clear notation of the association between tasks and services (R3). Precisely, tasks in elementary form are atomic units of work that can be fulfilled by a single service, and in composite form require more than one service to complete. This assists security analysis such as SoD (R2). Although there are many arguments on Petri Net as a workflow language, it suffices in principles to model a workflow at the service level. However, our approach is not limited to Petri Net. It can be extended to any graph-based language.

5 SERVICE WORKFLOW TRUST

5.1 Defining Workflow Trust

The most related trust definition among existing literatures is provided by Olmedilla, et al. (Olmedilla, et al., 2005): "*Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X).*" This implies that trust is unilateral from A acquiring the service X provided by B in a given context. However, the definition fails to address trust in another direction, when B needs to trust A before providing the service X. In term of service workflows, B might ascertain that the outcome from the service X is used only by a trusted party A, and will not be maliciously disseminated outside trusted domains.

Despite being claimed that trust is considerable confusion around the terminology in multiple meanings, this term is being used effectively in many contexts. In our aspect, the importance of incorporating trust in service workflows is that trust is an enabling technology. We summarize the definition of trust of service workflows as (Viriyasitavat and Martin, 2010): *"Trust in a workflow is a subjective, possibly mutual measurable, relationship between (direct and indirect) interacting services to act autonomously, securely, and reliably, in a given situation with a specific context of a given time."* Direct interaction takes place between two adjacent services in a workflow path, while the indirect one occurs when they are not (immediately) connected. The mutual trust relationship describes bidirectional measurable trust exhibited among participating services. Establishing trust in both directions is crucial, as one service may need to evaluate trustworthiness of a subsequent service before passing information, while the subsequent one perhaps requires trust of the outcome that is originated from the trusted source.

5.2 Trust Formalization

In our previous work (Viriyasitavat and Martin, 2010), we formulated trust in service workflows into three modes: Henceforth Path Trust (HPT), Backward Path Trust (BPT), and Existence Trust (ET) (see Figure 1). HPT and BPT addresses the direction for preceding and

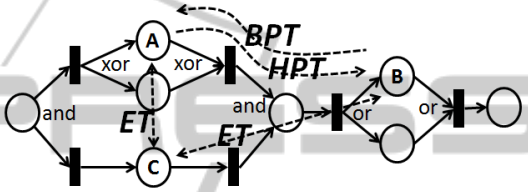

Figure 1: The Workflow Example with HPT, BPT and ET.

succeeding services, respectively, while ET aims at any service in the workflow.

Def. 2: Let A and B be two services in a service workflow. TMC is a vector as its elements (tmc_i) indicate computational trust models in use. A set of all possible paths (π) originating from A to B is denoted by

$$\pi(A,B) = \{< p_0, t_0, p_1, t_1 \dots, t_{n-1}, p_n >\}, \text{ where } A = p_0, \text{ and } B = p_n.$$

$< p_0, t_0, p_1, t_1 \dots, t_{n-1}, p_n >$ is a sequence of services and task (p and t are services and tasks in SWN). $< p_0, t_0, p_1, t_1 \dots, t_{n-1}, p_n > \subseteq n_0.F$, where n_0 is a SWN. Path Trust is classified into two directions:

1. Henceforth Path Trust (HPT):

$$\alpha_{TMC}^{A\to B} = A \succcurlyeq_{TMC=[tmc_1,\dots,tmc_n]} B$$

2. Backward Path Trust (BPT):

$$\beta_{TMC}^{A\to B} = A \succ_{TMC=[tmc_1,\dots,tmc_n]} B$$

$\alpha_{C,TMC}^{A\to B}$ and $\beta_{TMC}^{B\to A}$ denote trust values resulted from binary relation $\succcurlyeq$ and $\succ$, indicating the trust values of A placing on B. TMC is a vector where its elements indicate the desired trust models. $\alpha_{TMC}^{A\to B}$ and $\beta_{TMC}^{A\to B}$ are valid if there is one path connecting from A to B.

Trust values from HPT and BPT aim at a single target service, not intermediate services along the path. However, a property of the intermediate services might have an influence on the trust value of the target. This can be precisely addressed by the following TS algebra that will be described in the next section. We note that trust relationship is subjective where trust from one might be different from others.

Def. 3: Existence Trust (ET): Let the terms A, B and TMC be as defined. The ET definition is given below.

1. Existence Trust (ET):

$$\mathrm{E}_{TMC}^{A\to B} = A \trianglerighteq_{TMC=[tmc_1,\ldots,tmc_n]} B$$

$\mathrm{E}_{TMC}^{A\to B}$ denotes the trust value from binary relation $\trianglerighteq$, indicating the trust value of service A placing on B. Due to there being no condition on paths, ET is a looser version of Path Trust. It is more expressive as being able to address trust in any service in a workflow. This is worth to be addresses as the presence of one service might affect trust of others to participate in a workflow. (Figure 1 demonstrates HPT and BPT in the forms of $\alpha_{\mathrm{TMC}}^{\mathrm{A}\to\mathrm{B}}$ and $\beta_{\mathrm{TMC}}^{\mathrm{B}\to\mathrm{A}}$, and the two ET trust values are denoted by $\mathrm{E}_{\mathrm{TMC}}^{\mathrm{A}\to\mathrm{D}}$ and $\mathrm{E}_{\mathrm{TMC}}^{\mathrm{B}\to\mathrm{D}}$).

TMC is a vector referring to the use of the desired trust models, for instance, trust based on reputation, experience and behaviour, credential, policies, quality of service, provenance, etc. As noted, since a trust value can vary from different perspectives, and some of the models are not designed to be computable, **Def. 4** describes conversion functions to address this issue.

Def. 4: Let the terms A, B and TMC are similar to **Def 2**. The Conversion Function is given as follows:

$$\mathrm{F}: [\mathrm{tmc}_1, \ldots, \mathrm{tmc}_\mathrm{n}] \to [\mathrm{a}_1, \ldots, \mathrm{a}_\mathrm{n}], \text{ such that}$$
$$\mathrm{F}([\mathrm{tmc}_1, \ldots, \mathrm{tmc}_\mathrm{n}]) = [\mathrm{f}_1(\mathrm{tmc}_1), \ldots, \mathrm{f}_\mathrm{n}(\mathrm{tmc}_\mathrm{n})], \text{ and}$$
$$\mathrm{f}_\mathrm{i}(\mathrm{tmc}_\mathrm{i}) \in [-1..1].$$

Some trust schemes use discreet and continue value to measure trust. For example, some approaches use a discrete value [0, 1], while many others derive trust into a continuous value [0..1]. After analyzing the various metrics, it can be concluded that there is no universal metric generic for all applications. The reason to reduce trust to a single numeric value is that trust is a relative factor. It makes sense to use a unitless ratio value normalized in the interval [−1..1]. The analysis of using this approach can be found in (Marsh, 1994). At this stage, this metrics are left loose and opened for different characteristics, merely observing that TMC (rather than a single value) is needed to capture them.

6 TS ALGEBRA

6.1 Syntax of TS

Three categories of TS operators are presented: Composition, Path, and Direction operators. Composite formulae are built up based on the association between tasks and services; Path formulae based on Computational Tree Logic (CTL) describe sequences of events; and Direction formulae indicate the direction. The grammars are presented in the Backus–Naur Form:

1. Direction Formulas

$$\mathrm{W} ::= \top \mid \bot \mid \mathcal{H}\mathrm{R} \mid \mathcal{B}\mathrm{R} \mid \sim\mathrm{W} \mid (\mathrm{W} \wedge \mathrm{W}) \mid (\mathrm{W} \vee \mathrm{W})$$

2. Path Formulas

$$\mathrm{R} ::= \top \mid \bot \mid \mathrm{S} \mid \sim\mathrm{R} \mid (\mathrm{R} \wedge \mathrm{R}) \mid (\mathrm{R} \vee \mathrm{R}) \mid (\mathrm{R} \oplus \mathrm{R}) \mid \exists_\mathrm{t}\odot\mathrm{sR} \mid \exists_\mathrm{t} \Diamond \mathrm{R} \mid \exists_\mathrm{t}\Box\mathrm{R} \mid \exists_\mathrm{t}(\mathrm{R} \Cup \mathrm{R}) \mid \exists_\mathrm{t}(\mathrm{R} \cup \mathrm{R}) \mid \forall_\mathrm{t}\odot\mathrm{R} \mid \forall_\mathrm{t} \Diamond \mathrm{R} \mid \forall_\mathrm{t}\Box\mathrm{R} \mid \forall_\mathrm{t}(\mathrm{R} \Cup \mathrm{R}) \mid \forall_\mathrm{t}(\mathrm{R} \cup \mathrm{R})$$

3. Composite Formulas

$$\mathrm{S} ::= \mathcal{F}\mathrm{E}_\mathrm{t}\mathrm{Z} \mid \mathcal{P}\mathrm{E}_\mathrm{t}\mathrm{Z} \mid \mathcal{F}\mathrm{A}_\mathrm{t}\mathrm{Z} \mid \mathcal{P}\mathrm{A}_\mathrm{t}\mathrm{Z}$$
$$\mathrm{Z} ::= \varepsilon \mid (\mathrm{s}, \mathrm{t}, \mathrm{o}, \mathrm{A}) \mid \sim\mathrm{Z} \mid (\mathrm{Z} \sqcap \mathrm{Z}) \mid (\mathrm{Z} \sqcup \mathrm{Z}) \mid (\mathrm{Z} \boxplus \mathrm{Z})$$

Direction Formulas

The Henceforth ($\mathcal{H}$) and Backward ($\mathcal{B}$) operators specify the directions from the preceding to succeeding, and succeeding to preceding services respectively.

Path Formulas

The Temporal operators consist of a pair of symbols. The first part is one of $\exists_\mathrm{t}$ or $\forall_\mathrm{t}$ and the second part is one of $\odot$, $\Diamond$, $\Box$, $\Cup$, or $\cup$. The Next ($\odot$), Future ($\Diamond$), and Global ($\Box$)are similarly defined as in CTL. The Strong Until ($\mathrm{R}_1 \Cup \mathrm{R}_2$) specifies that R_1 must hold until the presence of R_2 and R_2 must hold in the future. The Weak Until ($\mathrm{R}_1 \cup \mathrm{R}_2$) is similar to the strong until, but R_2 is not required to hold. The For Some Path ($\exists_\mathrm{t}$) specifies that there must be some paths through a set of connected services $\mathrm{S}(\mathrm{t})$ where $\mathrm{t} \in \mathrm{T}$ in SWN. If t is omitted, it means there is no condition on paths specific to a particular task.

$$S_0(t) \subseteq \begin{cases} \{S_n | S_n \in G(S_0)\}, G(S_0): (S_0 \times t) \times (t \times P) \rightarrow P \\ \{S_n | S_n \in H(S_0)\}, H(S_0): (P \times t) \times (t \times S_0) \rightarrow P \end{cases}$$

The first line is a set of connected services if $\mathcal{H}$ is part of the formula; and the second indicates connected services in $\mathcal{B}$. Finally, the For All Path ($\forall_t$) specifies for all paths through a task t.

Composite Formulas

In the first (quantifier) part, the Forward ($\mathcal{F}$) and Previous ($\mathcal{P}$) address the target services immediately connected in the forward direction (service separation) and previous direction (service composition), through the task(s) t in E_t or A_t. Composite For Some (E_t) indicates there is at least one services immediately connected, and Composite For All (A_t) restricts to all services immediately connected through the task(s) t.

In second (property) part, ε represents a null element. (s,t,o,A) is an atomic element where s, t and o are a service name, type, and owner, respectively. A set of attributes A is used to indicate properties. For example, one can specify that services with CA certificate are trusted. Note that s and t can be null (ε) and A can be empty which means that there are no properties required. However, this element is flexible and opened for extension and implementation. The negation (∼) represents the negation of an expression, such as $\mathcal{HF}\forall_t{\sim}(\varepsilon, t_1, \varepsilon, \emptyset)$ means that there must be no subsequent service type t_1. The Composite Conjunction (⊓) allows trust requirements to be specified a service
composition in the Previous operator ($\mathcal{P}$), and a service separation ($\mathcal{F}$) in the Forward operator. The Composite Disjunction (⊔) indicates that in one or both services containing a desired property is trusted. The Composite Exclusive Disjunction (⊞) indicates only one service with a certain property is trusted. These are restricted to a task(s) t indicated by $\exists_t$ or $\forall_t$.

The remaining operators (∧, ∨, ∼) in Direction and Path formulae are similar to the definitions in propositional logic. The graphical explanation for each operator is illustrated in Figure 2.

6.2 Well-formed Formulas

Def. 5: Well-formed formulas of TS are defined below:

Atom: A propositional atom (s, t, o, A) is

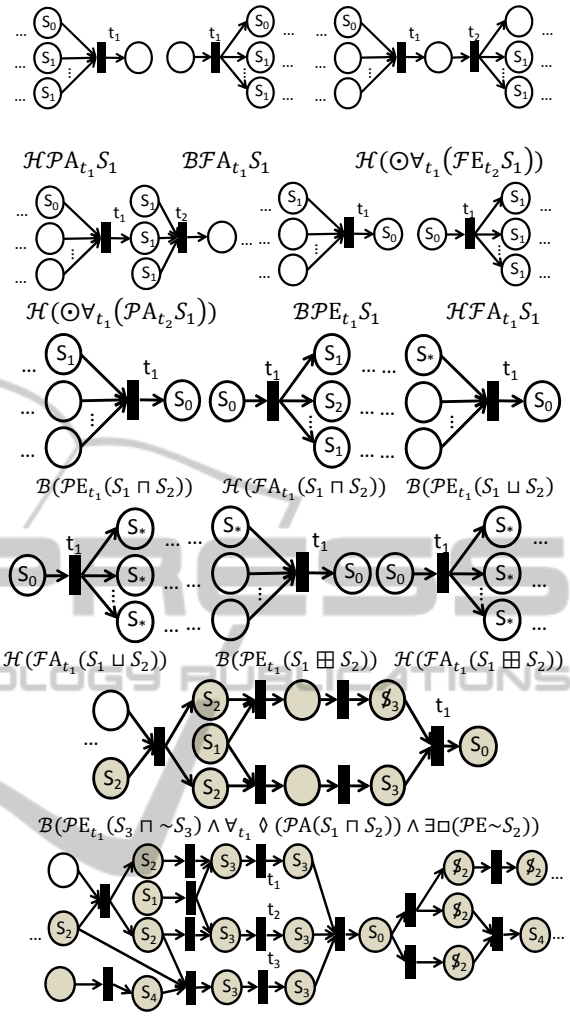

Figure 2: Graphical Illustration of Algebra Operators.

a Composite formula if it is preceded by Composite quantifier operators, $\mathcal{F}E_t, \mathcal{P}E_t, \mathcal{F}A_t$, or $\mathcal{P}A_t$ and a well-formed formula if further preceded by the Direction operators, $\mathcal{H}$, or $\mathcal{B}$, for example, $\mathcal{HF}E_t(s, t, o, A)$.

Composite Formulas: (Let Δ be an abbreviation of $\mathcal{F}E_t, \mathcal{P}E_t, \mathcal{F}A_t$, or $\mathcal{P}A_t$) If ΔZ_1 and ΔZ_2 are Composite formulas, then so are $\Delta{\sim}Z_1$, $\Delta(Z_1 \sqcap Z_2)$, $\Delta(Z_1 \sqcup Z_2)$, $\Delta(Z_1 \boxplus Z_2)$, $\Delta(Z_1 \Longrightarrow Z_2)$, and $\Delta(Z_1 \Leftrightarrow Z_2)$. Every composite formula is also a Path formula.

Path Formulas: If R_1 and R_2 are Path formulas, then so are ${\sim}R_1$, $(R_1 \wedge R_2)$, $(R_1 \vee R_2)$, $(R_1 \oplus R_2)$, $(R_1 \rightarrow R_2)$, $(R_1 \leftrightarrow R_2)$, $\exists_t \odot R_1$, $\exists_t \Diamond R_1$, $\exists_t \Box R_1$,

$\exists_t(R_1 \Cup R_2)$, $\exists_t(R_1 \cup R_2)$, $\forall_t \odot R_1$, $\forall_t \Diamond R_1$, $\forall_t \Box R_1$, $\forall_t(R_1 \Cup R_2)$, and $\forall_t(R_1 \cup R_2)$. These are well-formed if preceded by the Direction operators, $\mathcal{H}$, or $\mathcal{B}$.

Direction Formulas: If W_1 and W_2 are well-formed formulas, then so are $\sim W_1$, $(W_1 \wedge W_2)$, $(W_1 \vee W_2)$, $(W_1 \rightarrow W_2)$, and $(W_1 \leftrightarrow W_2)$ (Please note that the syntax applied to $\mathcal{H}$R are similar to $\mathcal{B}$R). Every Direction formula is well-formed.

6.3 Semantics of TS

This section provides more comprehensive detail on semantics serving as a key element for reasoning about service workflows. With well-defined semantics, reasoning algorithms can be developed based on this property to facilitate automatic interoperation. We also show that the semantics can be easily and concisely expressed in term of logic. In what follows, we write the path as $p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow \cdots \rightarrow p_n$. Regardless of tasks, we use $\prec$ for one-step connected services $p_0 \prec p_1 \prec \cdots \prec p_n$ and $<^n$ for abbreviation of $p_0 \prec p_1 \prec \cdots \prec p_n$ as $p_0 <^n p_n$.

Def. 6: Each formula ω is interpreted over a SWN $\mathcal{M}$. Let q_i be an abbreviation of (s, t, o, A) and p_0 is a service imposing a formula. Direction operators are explicitly integrated into each clause. The semantics of $\mathcal{M} \vDash \omega$ can be understood as follows: (Note that the following clauses are not exhaustively listed. It only shows important semantics of the TS)

Composite Formulas:

1. $\mathcal{M}, p_0 \vDash \top$, and $\mathcal{M}, p_0 \nvDash \bot$.
2. $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{P}A_t q_i)$ iff from p_0, for all p_i through a task t that $p_i \prec p_0$ ($p_0 \prec p_i$ in $\mathcal{H}(\mathcal{F}A_t q_i)$), p_i satisfies q_i.
3. $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{F}A_t q_i)$ iff from p_0, for all p_i through a task t that $t \rightarrow p_i, p_0$ ($p_i, p_0 \rightarrow t$ in $\mathcal{H}(\mathcal{P}A_t q_i)$) and $p_i \neq p_0$, p_i satisfies q_i.
4. $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{P}A_t(q_1 \sqcap q_2))$ iff from p_0, for all p_i through a task t that $p_i \prec p_0$ ($p_0 \prec p_i$ in $\mathcal{H}(\mathcal{F}A_t(q_1 \sqcap q_2))$), p_i satisfies q_1 and p_j satisfies q_2. p_i and p_j can be the same service.
5. $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{P}A_t(q_1 \sqcup q_2))$ iff from p_0, for all p_i through a task t that $p_i \prec p_0$ ($p_0 \prec p_i$ in $\mathcal{H}(\mathcal{F}A_t(q_1 \sqcup q_2))$), p_i satisfies q_1 or q_2.
6. $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{P}A_t(q_1 \boxplus q_2))$ iff from p_0, for all p_i through a task t that $p_i \prec p_0$ ($p_0 \prec p_i$ in $\mathcal{H}(\mathcal{F}A_t(q_1 \boxplus q_2))$), p_i satisfies only one property, either q_1 or q_2.
7. $\mathcal{M}, p_0 \vDash \sim \mathcal{B}(\mathcal{P}A_t q_i)$ iff $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{P}E_t \sim q_i)$.
8. $\mathcal{M}, p_0 \vDash \sim \mathcal{B}(\mathcal{F}A_t q_i)$ iff $\mathcal{M}, p_0 \vDash \mathcal{B}(\mathcal{F}E_t \sim q_i)$.
9. $\mathcal{M}, p_0 \vDash \sim \mathcal{H}(\mathcal{F}E_t q_i)$ iff $\mathcal{M}, p_0 \vDash \mathcal{H}(\mathcal{F}A_t \sim q_i)$.
10. $\mathcal{M}, p_0 \vDash \sim \mathcal{H}(\mathcal{P}E_t q_i)$ iff $\mathcal{M}, p_0 \vDash \mathcal{H}(\mathcal{P}A_t \sim q_i)$.

(Note that if E_t is presented instead of A_t, it indicates only some p_i instead of for all p_i)

Clause 1 reflects that $\top$ is always true and $\bot$ is always false. Clauses 2-3 mean that Composite atoms are evaluated either in Henceforth ($\mathcal{H}$) or Backward ($\mathcal{B}$) directions, in service composition ($\mathcal{P}$) or separation ($\mathcal{F}$) restricted to the task t. Clauses 4-6 extend the Composite atoms with $\sqcap$, $\sqcup$, and $\boxplus$. $\sqcap$ restricts two properties must be satisfied, $\sqcup$ indicate at least one of the two properties must be satisfied, and $\boxplus$ restricts that only one of properties is satisfied, but not both. Finally clauses 7-10 explain how the negation operator from a Direction part can propagate into the Composite part of the formula. Notice that the $\mathcal{H}, \mathcal{B}, \mathcal{P}$, and $\mathcal{F}$ are not affected by the negation propagation.

Path Formulas:

(Let S be the Composite part without Direction operator, for example, $S = \mathcal{F}A_t q_i$, or $S = \mathcal{P}E_t(q_1 \sqcap q_2)$ and ∇ be a substitution of $\mathcal{H}$ or $\mathcal{B}$)

1. $\mathcal{M}, p_0 \vDash \nabla(S_1 \wedge S_2)$ iff $\mathcal{M}, p_0 \vDash \nabla(S_1)$ and $\nabla(S_2)$.
2. $\mathcal{M}, p_0 \vDash \nabla(S_1 \vee S_2)$ iff $\mathcal{M}, p_0 \vDash \nabla(S_1)$ or $\nabla(S_2)$.
3. $\mathcal{M}, p_0 \vDash \mathcal{B}\forall_t(S_1 \Cup S_2)$ iff for all i where $p_{n_i} <^n p_0$ ($p_0 <^n p_{n_i}$ in the Henceforth direction, $\mathcal{H}$), $\mathcal{M}, p_{n_i} \vDash \mathcal{B}(S_2)$ and for all p_{m_i} through a task t where $m = 0, \ldots, n-1$, we have $\mathcal{M}, p_{m_i} \vDash \mathcal{B}(S_1)$.
4. $\mathcal{M}, p_0 \vDash \nabla(\forall_t \Diamond (S_1))$ iff $\nabla(\forall_t(\top \Cup S_1))$.
5. $\mathcal{M}, p_0 \vDash \nabla(\forall_t \Box(S_1))$ iff $\nabla(\sim\exists_t \Diamond (\sim S_1))$.
6. $\mathcal{M}, p_0 \vDash \nabla(\exists_t \Box(S_1))$ iff $\nabla(\sim\forall_t \Diamond (\sim S_1))$.
7. $\mathcal{M}, p_0 \vDash \nabla(\forall_t(S_1 \cup S_2))$ iff $\nabla(\forall_t(S_1 \Cup S_2) \vee \forall_t \Box(S_1))$.
8. $\mathcal{M}, p_0 \vDash \nabla(\exists_t(S_1 \cup S_2))$ iff $\nabla\big(\exists_t(S_1 \Cup S_2) \vee \exists t \Box S1$.
9. $\mathcal{M}, p_0 \vDash \mathcal{B}\forall_t \odot(S_1)$ iff for all i where $p_i \prec p_0$ ($p_0 \prec p_i$ in Henceforth direction, $\mathcal{H}$) through a task t, such that $\mathcal{M}, p_i \vDash \mathcal{B}(S_1)$.
10. $\mathcal{M}, p_0 \vDash \mathcal{B}\forall_t \odot(S_1)$ iff for some i where $p_i \prec p_0$ ($p_0 \prec p_i$ in Henceforth direction, $\mathcal{H}$) through a task t, such that $\mathcal{M}, p_i \vDash \mathcal{B}(S_1)$.

(Note that if $\exists_t$ is presented instead of $\forall_t$, it indicates only some i instead of for all i)

Clauses 1 and 2 are similar to the semantics in propositional logic and the remaining clauses are similarly described in CTL. The only difference is that all types of operators must be preceded by one of the Direction operators indicating the direction of

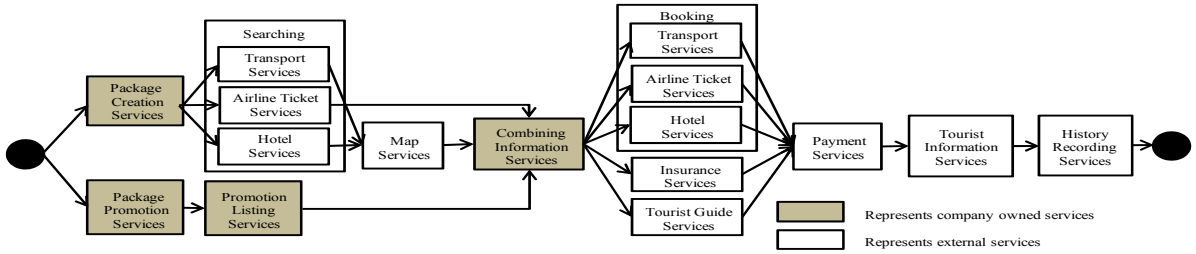

Figure 3: Travel Planning Processes.

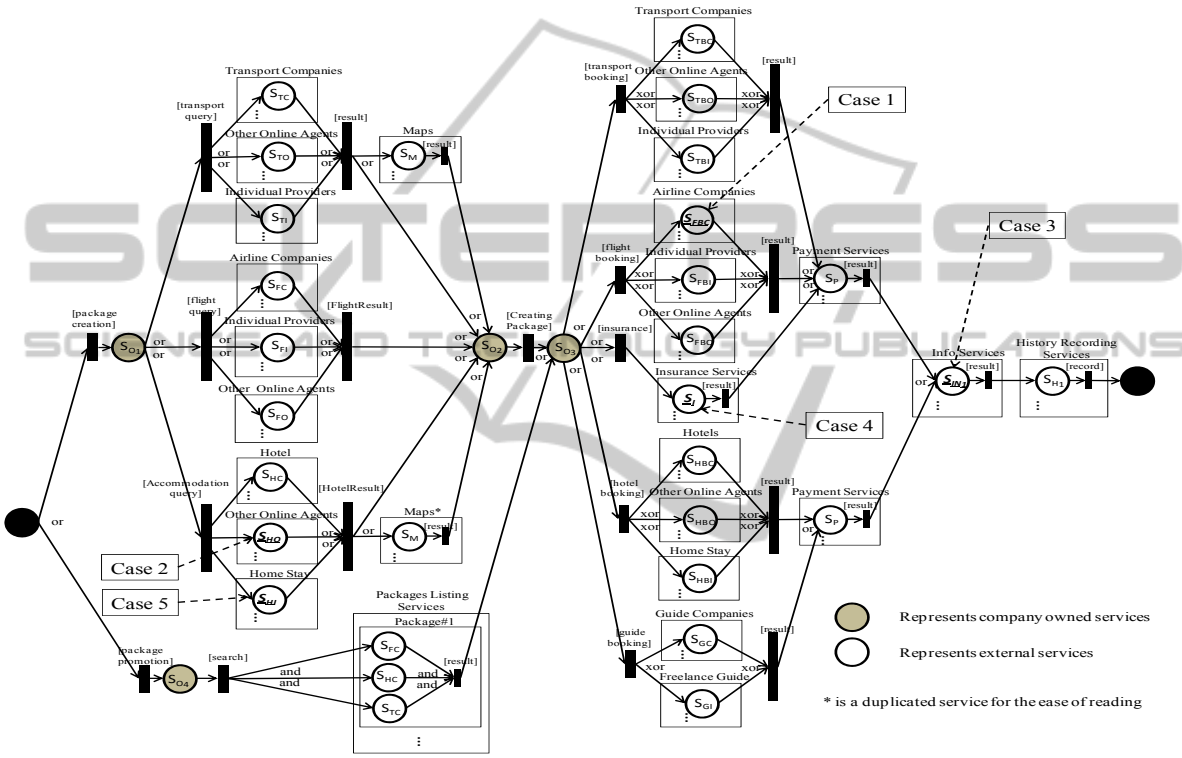

Figure 4: The Petri-net-based Workflow for All-in-one Travel Planning Processes.

paths in a workflow. For example, the formula $\mathcal{B}\forall_t(S_1 \Cup S_2)$ holds on for all paths in backward direction if it is the case that S_1 holds continuously until S_2 holds.

It is essential to extend the specification semantics with a formal analysis of satisfiability property. In the light of **Def 6**, this is used to determine whether a given TS formula is satisfied by a workflow.

Def. 7: Let ω be a TS formula and $\mathcal{M}$ be the formal representation of SWN. Whether TS satisfies, partially satisfies, or contradict with $\mathcal{M}$ is defined by the following relations.

1. $\mathcal{M} \models \omega$ is called Satisfiability, when TS issatisfied in every case in $\mathcal{M}$,
2. $\mathcal{M} \vdash \omega$ is called Partial Satisfiability, when TS $\models \mathcal{M}'$ and $\mathcal{M}' \subseteq \mathcal{M}$, and
3. $\mathcal{M} \nvdash \omega$ is called Contradictory, when TS does not satisfy any part of $\mathcal{M}$

7 APPLICATION EXAMPLE

7.1 Motivating Scenario

Suppose there is a travel planning workflow offering travel services by incorporating with multiple services including: (1) *Searching Services*: Transport, Airline, and Hotel Companies (S_{TC},S_{FC},S_{HC}), Individual Transport, Ticket, and

Hotel Providers (S_{TI},S_{FI},S_{HI}), and Online Transport, Ticket, and Hotel Agents (S_{TO}, S_{FO},S_{HO}), (2) *Map Services*: S_M providing hotels or transports locations, (3) *Booking Services*: Transport, Airline, and Hotel Companies (S_{TBC},S_{FBC},S_{HBC}), Individual Transport,

Ticket, and Hotel Providers (S_{TBI},S_{FBI},S_{HBI}) and Other Online Transport, Ticket, and Hotel Agents (S_{TBO},S_{FBO},S_{HBO}), and (4) *Others*: Insurance Services S_I. Payment S_P, Information Services S_{IN}, History Recording Services S_H, and the services $S_{O_1}..S_{O_n}$ belong to the workflow owner.

The workflow is preliminarily depicted in Figure 3, and the detailed workflow with its connectives is illustrated in Figure 4. Two options are available to the customers. To serve accurate demands, the first product, originated by the Package Creation, allows users flexibility in selecting flight tickets, hotel nights, and transports, while the Package Promotion provides more convenience by catering promotion packages in bundle. The Transport, Airlie Ticket, and Hotel services can perform in parallel and are optional. All transactions will be processed through the Booking service, Payment service, Tourist Info service, and History recording service consecutively. The Insurance service and Tourist Guide service are two additional purchasing depending on the customers' demands.

7.2 Applying TS

Five examples corresponding to the trust requirements (section 2) are described and summarized in table 2.

CASE 1: *Credential-based Trust*. In the situation when the Flight Booking (S_{FBC}) processes transactions of flight booking requests, S_{FBC} trusts the transactions originated from the airline companies certified by Business Airline Registry (C_{BAR}). The formal specification in Table 2 demonstrates the certificate requirement by the labeling function (**Def 1**), $C_{BAR} \in l(S_{FC}).A$ as the property of S_{FC}. The formula $\mathcal{B}(\forall \Diamond (\mathcal{P}A_{\text{FlightResult}}(\varepsilon, S_{FC}, \varepsilon, \{C_{BAR}\})))$ is imposed to restrict for Backward ($\mathcal{B}$) direction and for all path in future ($\forall \Diamond$) to reach all target services executing the FlightResult task. Because the formula satisfies only some paths (e.g. $\mathbf{S_{FC}}(\text{target}) \rightarrow S_{O_2} \rightarrow S_{O_3} \rightarrow \mathbf{S_{FBC}}(\text{source})$), it is partially satisfied by the workflow. The formula falls into the BPT mode where the trust value is presented as $\beta_{TMC}^{S_{FBC} \rightarrow S_{FC}}$. This example demonstrates the Association between Tasks and Services (R3) by explicitly restricting the services responsible for FlightResult task. (2) Flexible Degrees of Restriction (R6) can be described when parts of the workflow is not visible making the FlightResult invisible to S_{FBC}. The formula can be relaxed as $\mathcal{B}(\forall \Diamond (\mathcal{P}A(\varepsilon, S_{FC}, \varepsilon, \{C_{BAR}\})))$ by not specifying FlightResult task in A. (3) Task-oriented Property (R4) is addressed as the formula concern with trust of services executing FlightResult task.

CASE 2: *Trust in Providing Information*. The online hotel provider S_{HO} wants to protect the secret of the information of a price promotion. In order to pass such information along a workflow path until reaching the Booking service S_{HBO}, it trusts only the services owned by the workflow owner (S_{O_i}), expressed by $\mathcal{H}(\forall(\mathcal{F}A(\varepsilon, S_{O_i}, \varepsilon, \emptyset) \sqcup \mathcal{F}A(S_{HBO}, \varepsilon, \varepsilon, \emptyset)))$. Since the specification satisfies only some paths (e.g. $\mathbf{S_{HO}}(\text{source}) \rightarrow S_{O_2} \rightarrow S_{O_3} \rightarrow \mathbf{S_{HBO}}(\text{target})$), it is partially satisfied by the workflow. This specification falls into the HPT mode, and the trust value is presented by $\alpha_{TMC}^{S_{HO} \rightarrow S_{HBO}}$. This case demonstrates that the relative property (services owned by the workflow owner) of other services has an influence on the trust value of a target service (S_{HBO}). From this example, the TS is able to

Table 2: Examples of Trust Specification of Case 1 to 5.

Case	*Source*	*Target*	*Trust Specification (Formulas)*	*Trust Modes (source to target)*	*Satisfiability check ($\mathcal{M}$)*
1	S_{FBC}	S_{FC}	$\mathcal{B}(\forall \Diamond (\mathcal{P}A_{\text{FlightResult}}(\varepsilon, S_{FC}, \varepsilon, \{C_{BAR}\})))$ where $C_{BAR} \in l(S_{FC}).A$	$\beta_{TMC}^{S_{FBC} \rightarrow S_{FC}}$	Partially Satisfy
2	S_{HO}	S_{HBO}	$\mathcal{H}(\forall(\mathcal{F}A(\varepsilon, S_{O_i}, \varepsilon, \emptyset)) \sqcup (\mathcal{F}A(S_{HBO}, \varepsilon, \varepsilon, \emptyset))))$	$\alpha_{TMC}^{S_{HO} \rightarrow S_{HBO}}$	Partially Satisfy
3	S_{IN}	S_{HO},S_{TO},S_P	$\mathcal{B}(\mathcal{P}A(\varepsilon, S_P, \varepsilon, \emptyset)) \wedge \mathcal{B}(\forall \Diamond (\mathcal{P}A(\varepsilon, S_{HO}, \varepsilon, \emptyset))) \wedge ((\forall \Diamond (\mathcal{P}A(\varepsilon, S_{TO}, \varepsilon, \emptyset))))$	$\beta_{TMC}^{S_{IN} \rightarrow S_{HO}}, \beta_{TMC}^{S_{IN} \rightarrow S_{TO}}, \beta_{TMC}^{S_{IN} \rightarrow S_P}$	Partially Satisfy
4	S_I	S_{PFO}, S_{PHO}, S_{PTO}	$\mathcal{B}(\forall \Diamond (\mathcal{P}AS_{PFO} \vee \mathcal{P}AS_{PHO} \vee \mathcal{P}AS_{PTO}))$	$\beta_{TMC}^{S_I \rightarrow S_{PFO}}, \beta_{TMC}^{S_I \rightarrow S_{PHO}}, \beta_{TMC}^{S_I \rightarrow S_{PTO}}$	Contradict
5	S_{HI_1}	$S_{HBI} \rightarrow S_P$	$\mathcal{H}(\forall \odot (\mathcal{F}AS_P))$	$E_{TMC}^{S_{HI} \rightarrow S_{HBI}}$	Satisfy

address Flow-oriented Property Specification (R4) and Protection of Data (R7).

CASE 3: *Trust in Service Provider*. The Information service S_{IN} is willing to provide the service right after making a payment, and requires that the transactions must be originated from S_{HO} and S_{TO}. The formal specification is given as $\mathcal{B}(\mathcal{P}A(\varepsilon, S_P, \varepsilon, \emptyset)) \wedge \mathcal{B}(\forall \Diamond (\mathcal{P}A(\varepsilon, S_{HO}, \varepsilon, \emptyset)))) \wedge ((\forall \Diamond (\mathcal{P}A(\varepsilon, S_{TO}, \varepsilon, \emptyset))))$. The first term $\mathcal{B}(\mathcal{P}A(\varepsilon, S_P, \varepsilon, \emptyset))$ describes all services directly connected to S_{IN} must be the Payment service type (S_P), while the later two terms restrict the flow to be originated from S_{HO} and S_{TO}. This falls in the BPT mode where the trust values are $\beta_{TMC}^{S_{IN} \to S_P}$, $\beta_{TMC}^{S_{IN} \to S_{HO}}$ and $\beta_{TMC}^{S_{IN} \to S_{TO}}$. The first term illustrates our approach can address Enforcement of Sequences (R5). (Due to the page limits, the explanation of the remaining cases is discussed briefly)

CASE 4: *Trust in Service Provisions.* The Insurance service S_I requires that the Private Online Agent services must be involved in all paths. In this case, there are no private agents presented in the workflow which make this requirement contradicts with the satisfiability check. This can result in two consequences either the service S_I changes the requirement in order to participate in the workflow or decides not to participate. Alternatively, the workflow owner might replace the service S_I to avoid the conflict, or adjust the workflow to comply with this requirement.

CASE 5: *Existence Trust.* Suppose that Tom wants to participate in the workflow by letting one of his apartments. He creates the service S_{HI} and purchases the service S_{HBI} for the Room Booking process. Tom trusts the payment service S_P that have to be connected right after the service S_{HBI}. He has no concern with path from S_{HI} to S_{HBI}. This formula falls into the ET mode and represents Enforcement of Sequences (R5).

8 ANALYSIS OF TS

Interoperability with Local Security Requirements (R1): Obviously, the TS formula is a formal approach enabling Interoperability with Local Security Requirements since it allows each service to uniformly express its own requirements to other services.

Separation of Duty (R2): Although not illustrated by the example cases, our TS can address this specific requirement. For example, in a financial audit scenario, the annual financial statement must be audited by two different auditing companies. In this case, it is not necessary to precisely identify the specific companies, but instead have to make sure that they are not the same. The atomic proposition can be extended by introducing a dummy variable d_1 as $\mathcal{BP}A_{audit}((d_1, \varepsilon, , \varepsilon, \{audit\}) \sqcap ((\sim d_1, \varepsilon, , \varepsilon, \{audit\})))$. It means that two different services must be present to execute the audit task. The remaining trust requirements (R3-R7) are explicitly discussed along the way in the example cases.

Mutual Relationship: In some cases the relationship exists only in one direction. For example, if A trusts B, it is not necessary B to trust A. However, based on our definition of trust, the lack of trust relationship does not imply that there is no trust value in the computational sense. According to Conversion Function, if there is no trust, the function will return "0" as a default value.

9 CONCLUSIONS

This paper presents formal trust specification in service workflow environments. Three modes of trust and algebraic operators are developed to formally and uniformly express trust requirements from each autonomous service. The specification is also discussed with its syntax and semantics. TS formulas are incrementally built-up from Direction, Path, and Composite operators. The binding convention is described for operator priorities. To be able to reason about a service workflow, satisfiability relations are defined. Our solution provides advantages for the success of secure workflow interoperation in compliance with local trust requirements and grounds for automatic reasoning processes.

REFERENCES

Altunay, M., Brown, D., Byrd, G., Dean, R., 2005. Trust-Based Secure Workflow Path Construction, *In Proc. Of Intl. Conf. on Service Oriented Computing.*

Best, E., Devillers, R., Koutny, M., 2001. Petri Net Algebra, *EATCS Monographs on Theoretical Computer Science. Springer-Verlag.*

Davulcu, H., Kifer, M., Pokorny, L., Ramakrishnan, C. R., Ramakrishnan, I. V., Dawson, S., 1999. *Modeling and Analysis of Interactions in Virtual Enterprises.*

Falcone, R., Pezzulo, G., Castelfranchi, C., 2003. A fuzzy approach to a belief-based trust computation, *In Lecture Notes on Artificial Intelligence.*

Guha, R., Kumar, R., Raghavan, P., Tomkins, A., 2004.

Propagation of trust and distrust, *In Proc. of the 13th Intl. conf. on World Wide Web, ACM.*

Galizia, S., Gugliotta, A., Domingue, J., 2007. A trust based methodology for web service selection, *In Proc. of the Intl. Conf. on Semantic Computing.*

Jaeger, T., Tidswell, J. E., 2001. Practical Safety in Flexible Access Control Models, ACM Transactions on Information and System Security.

Jøsang, A., Marsh, S., Pope, S., 2006. Exploring different types of trust propagation, *In Proc. of the 4th Intl. Conf. on Trust Management.*

Klai, K., Tata, S., 2005. Abstraction-based Workflow Cooperation Using Petri Net Theory, In Proc of the *14th IEEE Intl. Workshops on Enabling Technologies: infrastructure For Collaborative Enterprise.*

Kuntze, N., Sch, J., 2008. Securing Decentralized Workflows in Ambient Environments, *In Intl. Conf. on Embedded and Ubiquitous Computing, IEEE/IFIP.*

Marsh S., 1994. Formalising Trust as a Computational Concept. *PhD thesis, University of Stirling\.*

Nyanchama, M., Osborn, S., 1999. The Role Graph Model and Conflict of Interest, *ACM Transaction on Information System Security.*

O'Donovan, J., Smyth, B., 2005. Trust in recommender systems, *In Proc. of the 10th Intl. Conf. on Intelligent User Interfaces.*

Salimifard, K., Wright, M., 2001. Petri-Net based Modeling of Workflow Systems: An Overview. *European Journal of Operational Research.*

Olmedilla, D., Rana, O., Matthews, B., Nejdl, W., 2005. Security and trust issues in semantic grids, In Proc. of the Dagsthul Seminar, Semantic Grid: The Convergence of Technologies.

Van der Aalst, W. M. P., 1998. Chapter 10: Three Good reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama, S. Kannapan, C. M. Khoong, S. Navathe, and J. Yates, editors, Information and Process Integration in Enterprises: Rethinking Documents, Vol 428 of the *Kluwer Intl. Series in Engineering and Computer Science.*

Van der Aalst, W. M. P., 2004. *Pi calculus versus petri nets: Let us eat"humble pie" rather than further inflate the "pi hype".*

Viriyasitavat, W., 2009. Modeling Delegation in Requirements-Driven Trust Framework. *In Congress on Services- I.*

Viriyasitavat, W., Martin, A., 2010. Formal trust specification in Service Workflows, *In IEEE/IFIP International Conference on Embedded and Ubiquitous Computing.*