Unifiable Preference Expressions for Pervasive Service Composition

Chun-Feng Liao, Hsueh-Hung Cheng, and Li-Chen Fu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan (R.O.C)
{liaocf, b96902201, lichen}@ntu.edu.tw

***Abstract*—Composing services in a pervasive environment usually involves user-in-the-loop adaption which is absent in most of the traditional enterprise service composition mechanisms. In such environment, the criteria for selecting and ranking services are usually specified by users, which tend to be vague and subjective. The criteria can be contradictory and the activated services can interfere with one another. This paper addresses these issues by defining a unifiable and negotiable expression language called the Preference Expression that is capable of specifying both enumerative/numeric as well as mandatory/negotiable user preferences. A set of unification rules for possible conflicting preferences is also derived. Experimental results show that the proposed approach is able to achieve high composition precision and maintains reasonable success rate at the same time.**

I. Introduction

Service composition, a technique for discovering, selecting, and activating service components to form a service that best fits pre-specified criteria, is an important issue in Service Computing and has received much interest in enterprise environments for a long time [3]. Enterprise services are usually composed based on a set of precisely specified service requests, which do not change frequently and are usually defined by domain experts to fit business requirements. In addition, the services components deployed in an enterprise environment are relatively well-defined and static.

Figure 1 depicts the process of typical service composition. In such process, service composition is driven by a user-specified Service Request that contains a set of Preference Descriptors for selecting qualified service components. After a Service Request is submitted, the process enters the Type-based Node Searching phase, in which a service composition manager searches for qualified candidates either from a centralized service registry or by broadcasting the type information as the searching criteria and waiting for responses. The procedure in this phase depends on the underlying service discovery infrastructure such as UDDI (Universal Description Discovery and Integration) [5] or Jini [4]. Usually, there are more than one qualified candidates, so that the manager selects the best one among candidates in the Candidate Scoring and Selection phase. After all of the most appropriate candidates are determined, each of them is then activated in the Service Activation phase.

Although it has been reported that service composition is also significant in pervasive environments such as smart homes [2], however, existing enterprise service composition techniques are not suitable for pervasive systems. Unlike in enterprise environments, the goal of pervasive service composition is to compose services that meet maximum satisfactions of users, which tend to be vague and subjective. Complexities arise when many users submit conflicting service requests at the same time. It follows that the core issue of pervasive service composition is twofold : 1) how users specify their preferences precisely, and 2) how to unify possible conflicting service requests submitted by multiple parties. Unfortunately, most of current works focus on designing sophisticated mechanisms in the Candidate Scoring and Selection phase [7], [8], [9], [6] instead of dealing with the request conflicting issue mentioned above. The objective of this paper is therefore to devise a set of mechanisms that are able to capture users' preferences precisely and to negotiate when there are conflicting preferences.

In this work, we append an additional phase, namely, the Preference Unification phase, before the Type-based Node Searching phase (see Fig. 2) so that before a service composition starts, users are able to adjust Preference Descriptors to make the composed service more satisfactory. In addition, a formal expression language, namely, the Preference Expression (PE), based on CC/PP (Composite Capability/Preference Profiles) standard [10] is also introduced. PE denotes user preferences from two perspectives: the enumerability and the necessity of preferences. A set of rules for unifying different types of potentially conflicting preferences are also presented

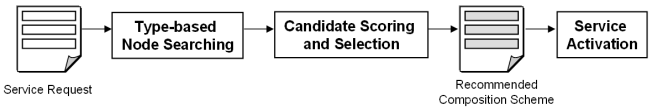


Fig. 1: A typical service composition architecture

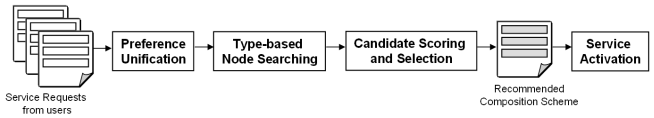


Fig. 2: The refined architecture that facilitates preference unification

978-0-7695-4624-7/11 $26.00 © 2011 IEEE
DOI 10.1109/APSCC.2011.11

(see Section IV). Note that one of the most distinguishing feature of the proposed PE is that it is negotiable so that users are able to indicate that there are spaces for compromise when their preferences are mutually exclusive. In this way, the proposed approach is able to reach high success rate of composition while maintaining user-acceptable quality of services, as will be discussed in Section V.

II. Related Work

Service composition has been one of the most active research issues in Service Computing [3]. In the last few years, a considerable number of studies have been made on designing a service composition system, but most of them focus on enterprise environments. According to a recent study conducted by Bronsted *et al.* [2], there are surprisingly few researches have been done on the service composition issues in pervasive environments such as the smart homes. They also observed that little research has been made on dealing with possible conflicting user preferences.

There are some approaches proposed to model user preferences in an intelligent information system such as Lee *et al.* [11] and Mandel *et al.*[12]. However, these approaches either do not provide a completed formal framework for user to represent their preference or does not work when multiple conflicting preferences are present. Existing pervasive service composition mechanisms deal with conflicting preferences either explicitly defining a precedence [13] or by attempting to seek a common ground among conflicting ones by using logic-based approaches [14]. Shankar *et al.* [15] proposes an Event-Condition-Action-Post-Condition (ECA-P) policy model. This work detects conflicts among policies by analyzing their semantic post-conditions and replaces conflicting ones with the one with preferred post-conditions. However, due to limited expressiveness of ECA-P model, the results are decisive: users either come to a common agreement or the service is not provided at all. However, in real cases, users tend to be negotiable. Obviously, more powerful representation techniques have to be developed in order to capture the negotiable user preferences. It follows from the above discussions that there is still much space for further investigation on the problem of conflicting preferences in pervasive service composition. This work therefore concentrates on dealing with this issue.

III. Service Model

In this paper, the proposed approach is presented based on a message-oriented service model called PerSAM (Pervasive Service Application Model). To facilitate further discussions, this section briefly review PerSAM; details of the object models and algorithms can be found in [16].

A PerNode $p \in P$ is a basic logical software entity in a system, where P is the universe of PerNodes. PerNode has two subtypes: Worker Node and Manager Node. A Worker Node $w \in W$ is a PerNode that encapsulates a unit of application logic, where W is the universe of Worker Nodes in the system. Worker Nodes are basic service providing units of a Pervasive Service. In this work, the capabilities and selecting criteria for Worker Nodes are modeled by extending CC/PP, which is a W3C standard for specifying device capabilities and user preferences [10]. The capability of a Worker Node w is described by its type τ and a set of attributes $A = \{\alpha_i\}$, where $\alpha_i = (n_i, v_i)$ is a name-value pair, based on which the Worker Node can be described by arbitrary attributes. The description of a Worker Node's capability is called a Capability Descriptor and is formally defined as follows.

Definition 1. (Capability Descriptor) *The Capability Descriptor of a Worker Node w is a pair: $C(w) \triangleq (\tau, A)$, where τ is the type of w, and $A = \{\alpha_i\}$ is the attribute set of w.*

For example, a Worker Node w_1 that controls an 37 inches LCD monitor which locates at the room-1 can be described as follows:

$$(LCD, [(size, 37), (location, "room - 1")]).$$

In PerSAM, a smart home application is called a Pervasive Service, denoted s, which consists of one or more Worker Nodes that collectively provide a service to users. Each Pervasive Service is managed by a Pervasive Service Manager (PSM), which is responsible for composing, activating, and monitoring the corresponding Pervasive Service. All members of a Pervasive Service, but not PSM, are dynamic, and each PSM is responsible for selecting most appropriate Worker Nodes during the process of service composition. A Pervasive Service can be formally denoted as a tuple:

$$s = \langle m^s, W^s \rangle \in M \times 2^W, \tag{1}$$

where m^s is the PSM of s, M is the universe of PSMs, and W^s is the set of Worker Nodes belonging to s.

Typically, a service designer defines an application (i.e. a Pervasive Service) by specifying a Service Request. A Service Request specifies a Preference Descriptor $P(\widetilde{w})$ for each desired Worker Node $\widetilde{w}$ so that a PSM is able to search for qualified nodes according to the descriptor. The structure of a Preference Descriptor is defined below.

Definition 2. (Preference Descriptor) *The Preference Descriptor of a desired Worker Node $\widetilde{w}$ is a pair: $P(\widetilde{w}) \triangleq (\tau, \widetilde{A})$, where τ is the type of $\widetilde{w}$, and $\widetilde{A} = \{\widetilde{\alpha}_i\}$. Note that $\widetilde{\alpha}_i = (n_i, \epsilon_i)$, where n_i is the attribute name and ϵ_i is an expression that specifies selection criteria for attribute values.*

For example, the following Preference Descriptor $P(\widetilde{w}_1)$ directs the PSM to search for an LCD monitor that is more than 30 inches and is located at room-2:

$$(LCD, [(size, (\geq 30)), (location, (== "room - 2")]).$$

Depending on the characteristics of service composition mechanisms, the syntax of ϵ is usually different. To facilitate consistent smart home applications, we propose an unifiable and negotiable expression called the Preference Expression. The details of Preference Expression will be elaborated in the follow sections.

Listing 1: The BNF of MEPE

```
MEPE ::= PtList | NegationExpr
PtList ::= ' ( ' == STRING PtListTail* ' ) '
PtListTail ::= →== STRING
NegationExpr ::= ' ( ' ! = STRING(∧! = STRING)* ' ) '
```

IV. Unifiable Preference Expressions

This section presents the syntax and unification rules of a set of unifiable preference expressions. The approach taken by this work is first to distinguish different kinds of preferences from two perspectives and then propose approaches to deal with them separately. Specifically, we design Preference Expression from two aspects: 1) a preference can be specified either numerically or enumeratively, and 2) from the user's points of view, the preference can be either mandatory or negotiable. The following subsections are going to elaborate each of these combinations and their unification rules in detail.

A. Unifiable Enumerative Preference Expressions

The Enumerative Preference Expression is used to describe preferences for an enumerable attribute by specifying a list of preferred or un-preferred values. This type of expression is called a Mandatory Enumerative Preference Expression (MEPE). For instance, the preference to the composers of a music can be specified as (== "Bach" →== "Mozart").

The syntax of MEPE is presented in Listing 1 in BNF (Backus-Naur Form) [17]. In an MEPE, the preferred values are a listed of strings, delimited by arrowheads. Alternatively, one can enumerate the undesired values by a list of conjunctions (e.g. ! = "Bach"∧! = "Vivaldi"). The list is ordered by preferences in descending order so that one can easily conceive that the first qualified service component is the most preferable one. The expression is evaluated to be true as soon as the PSM finds a service component whose attribute value meets the criteria specified in the expression. Alternatively, one can enumerate the undesired values by a list of conjunctions (e.g. ! = "Bach"∧! = "Vivaldi"), as mentioned earlier, and then the expression is evaluated to be true if the attribute value of a service component matches none of the undesired values.

In a Preference Expression, the term with an operator is called a "preference term", or simply a "p-term". A set of p-terms is called a "preference term set", or called a "pt-set", which is denoted as $pt(\varepsilon)$, where ε is the Preference Expression. Assume that there are k possibly conflicting Preference Expressions $\{\varepsilon_i\}_{i=1}^{k}$, then the pt-sets of these expressions are denoted as $\{pt(\varepsilon_i)\}_{i=1}^{k}$. For example, if ε = (== "Bach" →== "Mozart"), then $pt(\varepsilon)$ = {== "Bach", == "Mozart"}. In MEPE, p-terms that are associated with the notations "==" and "! =" operators are respectively called positive p-terms (denoted ε^+) and negative p-terms (denoted ε^-). An MEPE is either composed of a set of positive p-terms, called Positive MEPE, or a set of negative p-terms, called Negative MEPE, but not a mixture of them. The pt-set of a Positive MEPE and a Negative MEPE are denoted as $pt(\varepsilon^+)$ and $pt(\varepsilon^-)$, respectively.

If there are more than one specified preference expressions, then these expressions have to be unified. The core idea of unifying Preference Expressions is to construct a new expression such that for all p-terms in the new expression satisfy all involved original expressions. If there is at least one Positive MEPE, then a set of possibly conflicting MEPEs can be unified as a single Positive MEPE. Assume that $\{\varepsilon_i\}_{i=1}^{k}$ are a set of possibly conflicting MEPEs, where $\exists\varepsilon^+ \in \{\varepsilon_i\}_{i=1}^{k}$ such that ε^+ is a Positive MEPE. Then, the set $\{\varepsilon_i\}_{i=1}^{k}$ can be integrated into a Positively Unified MEPE, denoted as ε^{u+}, where $\forall t \in pt(\varepsilon^{u+})$, $\wedge_{i=1}^{k_1}[t \in pt(\varepsilon_i^+)]$ and $\wedge_{j=k_1+1}^{k}[t \notin pt(\varepsilon_j^-)]$. The core idea is to construct an expression ε^{u+} such that all p-terms in ε^{u+} satisfy all of the involved original expressions.

The following theorem presents a general form of unifying MEPEs where there is a mixture of possibly conflicting positive and negative MEPEs.

Theorem 1. (Deriving the unified pt-set of a mixture of Positive and Negative MEPEs) *If there is a mixture of several possibly conflicting Positive and Negative MEPEs, then the pt-set of the Positively Unified MEPE $pt(\varepsilon^{u+})$ can be obtained by the following operations:*

$$pt(\varepsilon^{u+}) = \cap_{i=1}^{k'} pt(\varepsilon_i^+) - \cup_{j=k'+1}^{k} pt(\varepsilon_j^-), \tag{2}$$

where there are k' positive MEPEs and $k-k'$ negative MEPEs.

Proof. Based on the De Morgan's laws and the set difference operation, that is, $A - B = A \cap \bar{B}$, (2) can be transformed to intersections of pt-sets, specifically,

$$\begin{aligned} pt(\varepsilon^{u+}) &= \cap_{i=1}^{k'} pt(\varepsilon_i^+) - \cup_{j=k'+1}^{k} pt(\varepsilon_j^-) \\ &= \cap_{i=1}^{k'} pt(\varepsilon_i^+) \cap \overline{\cup_{j=k'+1}^{k} pt(\varepsilon_j^-)} \\ &= \cap_{i=1}^{k'} pt(\varepsilon_i^+) \cap \cap_{j=k'+1}^{k} \overline{pt(\varepsilon_j^-)} \end{aligned}$$

Thus, we have:

$$\begin{aligned} &[pt(\varepsilon^{u+}) \subseteq pt(\varepsilon_1^+)] \wedge ... \wedge [pt(\varepsilon^{u+}) \subseteq pt(\varepsilon_{k'}^+)] \wedge \\ &[pt(\varepsilon^{u+}) \subseteq \overline{pt(\varepsilon_{k'+1}^-)}] \wedge ... \wedge [pt(\varepsilon^{u+}) \subseteq \overline{pt(\varepsilon_k^-)}], \end{aligned}$$

which can be rewritten as

$$\begin{aligned} &[pt(\varepsilon^{u+}) \subseteq pt(\varepsilon_1^+)] \wedge ... \wedge [pt(\varepsilon^{u+}) \subseteq pt(\varepsilon_{k'}^+)] \wedge \\ &[pt(\varepsilon^{u+}) \nsubseteq pt(\varepsilon_{k'+1}^-)] \wedge ... \wedge [pt(\varepsilon^{u+}) \nsubseteq pt(\varepsilon_k^-)]. \end{aligned}$$

As a result, $\forall t \in pt(\varepsilon^{u+})$, we see that

$$\begin{aligned} &[t \in pt(\varepsilon_1^+)] \wedge [t \in pt(\varepsilon_2^+)] \wedge ... \wedge [t \in pt(\varepsilon_{k'}^+)] \wedge \\ &[t \notin pt(\varepsilon_{k'+1}^-)] \wedge [t \notin pt(\varepsilon_{k'+2}^-)] \wedge ... \wedge [t \notin pt(\varepsilon_k^-)]. \end{aligned}$$

In other words, $\forall t \in pt(\varepsilon^{u+})$, we have

$$\wedge_{i=1}^{k'}[t \in pt(\varepsilon_i^+)] \wedge \wedge_{j=k'+1}^{k}[t \notin pt(\varepsilon_j^-)].$$

□

After deriving unified pt-sets, the order of p-terms have to be determined if the result is a Positively Unified MEPE. To determine the order of p-terms, users have to designate one

Listing 2: The BNF of NEPE

```
NEPE ::= PtList ? ( : ' ( ' NegationExpr ' ) ' )?
PtList ::= ' ( ' == STRING PtListTail* ' ) '
PtListTail ::= →== STRING
NegationExpr ::= ' ( ' ! = STRING(∧! = STRING)* ' ) '
```

Positive MEPE as the master expression. For example, if we want to unify the following MEPEs:

$$\varepsilon_1^+ = (== \text{"Bach"} \rightarrow== \text{"Mozart"} \rightarrow== \text{"Vivaldi"})$$
$$\varepsilon_2^+ = (== \text{"Mozart"} \rightarrow== \text{"Bach"} \rightarrow== \text{"Vivaldi"})$$
$$\varepsilon_3^- = (! = \text{"A. Vivaldi"}).$$

, where ε_1^+ is the master expression. The pt-set of the unified MEPE can be obtained by applying Theorem 1:

$$\begin{aligned} pt(\varepsilon^u) &= pt(\varepsilon_1^+) \cap pt(\varepsilon_2^+) - pt(\varepsilon_3^-) \\ &= \{== \text{"Mozart"}, \text{"Bach"}, \text{"Vivaldi"}\} - \{\text{"Vivaldi"}\} \\ &= \{== \text{"Mozart"}, == \text{"Bach"}\}. \end{aligned}$$

Next, $pt(\varepsilon^u)$ is ordered according to ε_1^+ which is chosen as the master expression. Hence, the Unified MEPE, denoted as ε^u, can be obtained after attaching the operators:

$$\varepsilon^u = (== \text{"Bach"} \rightarrow== \text{"Mozart"}).$$

As mentioned earlier, representing preferences by mandatory expressions is decisive, that is, users either come to an agreement or no service is provided at all. However, users are usually willing to negotiate: they do not always insist on the criteria and may want to give up some desired service quality if the criteria can not be met in the first place. The Negotiable Enumerative Preference Expression (NEPE) is designed for this purpose, which is used to specify the "good to have" criteria for an enumerative attribute. The BNF of NEPE is presented in Listing 2.

An NEPE has the form $\mathbb{P} : \mathbb{N}$, where the $\mathbb{P}$ segment is a list of positive p-terms, whereas the $\mathbb{N}$ segment is a set of negative p-terms. For example, in the expression $(==$ "Mozart" $:! =$ "Vivaldi"), the $\mathbb{P}$ segment is $==$ "Mozart", and the $\mathbb{N}$ segment is $! =$ "Vivaldi". The p-terms in $\mathbb{P}$ specify all "good to have" options. If the p-terms in $\mathbb{P}$ can not be satisfied, then the expression can be considered satisfied as long as the criteria specified in $\mathbb{N}$ are evaluated to be True. If follows that if one of the Preference Expressions to be unified is NEPE, then the expressions are first treated as Positive MEPEs that are composed of the p-terms in $\mathbb{P}$. If the unification fails, namely, $pt(\varepsilon^u) = \phi$, then the p-terms in $\mathbb{P}$ are replaced by those in $\mathbb{N}$ and then they are unified again. In this way, the NEPE provide an additional chance for unification, since $\mathbb{N}$ has weaker constraint than $\mathbb{P}$. For example, if we want to derive the Unified MEPE from the following MEPEs and NEPEs, where ε_1 is the master expression:

$$\varepsilon_1 = (== \text{"Bach"} \rightarrow== \text{"Vivaldi"} \rightarrow== \text{"Haydn"})$$
$$\varepsilon_2 = (== \text{"Mozart"} :! = \text{"Vivaldi"}).$$

First, ε_2 has to be converted into an MEPE based on $\mathbb{P}$, so that we have $\varepsilon_2^{\mathbb{P}} = (== \text{"Mozart"})$, where $\varepsilon_2^{\mathbb{P}}$ is an MEPE that are composed of all p-terms in $\mathbb{P}$. However, the unification between ε_1 and $\varepsilon_2^{\mathbb{P}}$ fails, since

$$\begin{aligned} pt(\varepsilon^u) &= pt(\varepsilon_1) \cap \varepsilon_2^{\mathbb{P}} \\ &= \{== \text{"Bach"}, == \text{"Vivaldi"}, == \text{"Haydn"}\} \\ &\cap \{== \text{"Mozart"}\} = \phi. \end{aligned}$$

Next, because ε_2 is negotiable, $\varepsilon_2^{\mathbb{P}}$ is replaced by $\varepsilon_2^{\mathbb{N}}$ so that the unification is performed again. Hence,

$$\begin{aligned} pt(\varepsilon^u) &= pt(\varepsilon_1) - \varepsilon_2^{\mathbb{N}} \\ &= \{== \text{"Bach"}, == \text{"Haydn"}\}. \end{aligned}$$

Finally, $pt(\varepsilon^u)$ is ordered according to ε_1 which is chosen as the master expression. Hence, the Unified MEPE can be obtained as follows:

$$\varepsilon^u = (== \text{"Bach"} \rightarrow== \text{"Haydn"}).$$

It is important to observe that the result of unifying a set of MEPEs and NEPEs must be an MEPE. The reason is that the outcome has to be a consensus (and also the most constrained). If there is at least one of them which is not negotiable, the outcome must not be negotiable. But for a special case where all expressions to be unified are NEPEs, the outcome will be an NEPE. When unifying NEPEs, the p-terms in $\mathbb{P}$ and in $\mathbb{N}$ segments are converted into MEPEs and are unified correspondingly. If either the unification result of $\mathbb{P}$ segments or that of the $\mathbb{N}$ segments is ϕ, then the final result have to be transformed to an MEPE. For example, if ε_2 is replace by $(==$ "Mozart" $:! =$ "Vivaldi"), causing $pt(\varepsilon^{u_1}) = \phi$, then the final result becomes $\phi : (! =$ "Haydn"$\wedge ! =$ "Vivaldi"), which can be rewritten as a negative MEPE $(! =$ "Haydn"$\wedge ! =$ "Vivaldi").

B. Unifiable Numeric Preference Expressions

Numeric attributes are different from enumerative ones in that they are numerically comparable and that they can be constrained by specifying intervals (i.e. upper and lower bounds). As a result, numeric expressions must support more operators than that are supported in enumerative ones. Specifically, there are only two operators supported in Enumerative Preference Expressions: "==" and "!=", whereas Numeric Preference Expressions uses additional operators such as ">", "<", "≤", "≥", and "¬". Numeric Preference Expressions can also be mandatory or negotiable. A Mandatory Numeric Preference Expression (MNPE) is a numeric preference expression whose criteria must be met. For example, the MNPE:

$$((> 20 \wedge \leq 30) \vee < 10) \tag{3}$$

can be used to specify the selection criteria of size of an LCD display whose size is either between 20 to 30 inches or smaller than 10 inches.

On the other hand, the Negotiable Numeric Preference Expression (NNPE) is the numeric preference expression that contains "negotiable" semantics. Similar to NEPE, an NNPE

TABLE I: Reducing $< x \vee < y$ when (a) $x > y$, (b) $x < y$, and (c) $x = y$.

No.	Case	Rule
(1)	$< x \vee < y$	if $x \geq y$ then $< x$ else $< y$
(2)	$> x \vee < y$	if $x \leq y$ then $True$
(3)	$> x \vee > y$	if $x \geq y$ then $> y$ else $> x$
(4)	$== x \vee < y$	if $x \leq y$ then $< y$
(5)	$== x \vee > y$	if $x \geq y$ then $> y$
(6)	$== x \vee == y$	if $x == y$ then $== x$
(7)	$! = x \vee < y$	if $x < y$ then $True$ else $! = x$
(8)	$! = x \vee > y$	if $x > y$ then $True$ else $! = x$
(9)	$! = x \vee == y$	if $x == y$ then $True$ else $! = x$
(10)	$! = x \vee ! = y$	if $x == y$ then $! = x$ else $True$

is also composed of a $\mathbb{P}$ segment and an $\mathbb{N}$ segment which are delimited by a colon mark. For example, the following expression

$$((> 20 \vee < 10) \wedge ! = 25 : \gg) \tag{4}$$

is capable of specifying a selection criteria and a negotiable expression for an LCD display, where the former is that the size should be either larger than 20 inches or less than 10 inches and must not equal 25 inches, whereas the later is the expression after the colon mark (":"), i.e. the $\mathbb{N}$ segment with the notation "$\gg$", which means the size is the greater the better. Note that the $\mathbb{N}$ segment of an NNPE is useful when the user only wants to specify a vague constraint for an attribute. For instance, $\gg$, $\ll$, and $\approx$ denote "the greater the better", "the less the better", and "the closer to a specified value the better".

Not surprisingly, the unification rules for Numeric Preference Expressions are different from enumerative ones because they are now integration of numerical interval as well as comparative operators rather than lists of strings. However, it can be shown that the integration of numerical intervals and operators can actually be reduced to a few types of compact forms so that specific unification rules for these compact forms can still be derived to integrate Numeric Preference Expressions efficiently.

The first step is to convert the expressions into Conjunctive Normal Forms (CNF), where a clause is a disjunction of logical terms (e.g. $> 30 \vee < 20$). Theoretically, every logical expression can be converted into an equivalent CNF expression by repeatedly applying distributive law and De Morgan's laws. The purpose for converting expressions into CNF is that both $\cap$ and $\cup$ satisfy the associativity property so that the logical terms can be unified pairwise. Specifically, after an MNPE is converted into a CNF, all clauses are connected by $\cap$, and all logical terms are connected by $\cup$. Therefore, logical terms within a clause can be unified pairwise, and the order in which they are unified does not affect the outcome. The same principle holds for clauses within an MNPE. Taking the MNPE in (3) as an example, it can be converted in to the following CNF by applying De Morgan's laws, that is, $((> 20 \wedge \leq 30) \vee < 10) \equiv (((> 20 \vee < 10) \wedge (\leq 30 \vee < 10))$.

The next step is to derive the most compact form for each disjunctive clause. In fact, all disjunctive clause can be reduced to one of the eight compact forms shown in Table II by repeatedly applying the reduction rules shown in Table I. Among six different operators defined in MNPE, $\leq$ and $\geq$ is semantically equivalent to $(< \vee ==)$ and $(> \vee ==)$, respectively. In this way, the number of different operators can be reduced to four: $>$, $<$, $==$, and $! =$. Consequently, there are 10 possible pairwise combinations among numeric p-terms in a disjunctive clauses (see Table I). Because the logical operator for connecting p-terms in distinctive clauses is $\vee$, the outcomes of unifications should be with fewer constraints, that is, with greater possible coverage. The reduction rules for the cases (1) to (10) listed in Table I can be diagrammatically and intuitively derived.

TABLE II: Compact forms of Numeric Preference Expressions

No.	Type	Compact Form
(1)	Negation	$! = s$
(2)	Disjoint intervals or Disjunctions of positive terms	$> a \vee < b \vee \bigvee_i (== x_i)$, where $a \neq b$
(3)	Disjoint intervals or Disjunctions of positive terms	$> a \vee \bigvee_i (== x_i)$
(4)	Disjoint intervals or Disjunctions of positive terms	$< b \vee \bigvee_i (== x_i)$
(5)	Disjoint intervals	$> a \vee < b$, where $a \neq b$
(6)	Disjoint intervals	$> a$
(7)	Disjoint intervals	$< b$
(8)	Disjunctions of positive terms	$\bigvee_i (== x_i)$

Now let us prove that all disjunctive clauses can be reduced to one of the eight compact forms shown in Table II.

Lemma 1. *If a p-term of the form $! = s$ appears in a disjunctive clause, then either $! = s$ is the only p-term in the disjunctive clause or the clause is resolved to be True.*

Proof. The numeric unification is performed pairwise throughout the disjunctive clause. According to cases (7)-(10) shown in Table I, the results of integrating "$! = s$" with another p-term is either $True$ or "$! = s$". If the result is $True$, then the whole disjunctive clause are immediately evaluated as being $True$; otherwise, only "$! = s$" is derived and then it is integrates with the next term in the disjunctive clause. Finally, the clause contains either solely "$! = s$" or the whole clause is evaluated as being $True$. □

Lemma 2. *There is at most one p-term of the form $> a$ and at most one p-term of the form $< b$ in a disjunctive clause.*

Proof. This lemma can be directly proved by using case (1) and case (3) shown in Table I. Assuming there are two different p-terms of the form $> a$, (for instance, $> x$ and $> y$), then according to case (1) in Table I, the p-terms can be integrated into single term, namely, either $> x$ or $> y$. Similar results hold for for $< b$ according to case (3). □

Lemma 3. *All disjunctive clauses can be reduced to the general forms shown in Table I, namely, either $! = s$ or $> a \vee < b \vee \bigvee_i (== x_i)$, where $a \neq b$.*

TABLE III: Unification rules for $NegotiationExpr$

Master	Slave	Outcome
$\gg$	$\gg$	$\gg$
$\ll$	$\ll$	$\ll$
$\gg$	$\ll$	ϕ
$\ll$	$\gg$	ϕ
other cases	other cases	Master $NegotiationExpr$

Proof. Recall that the general form of MNPE is

$$\bigvee_{i_1}(!=s_{i_1}) \vee \bigvee_{i_2}(>a_{i_2}) \vee \bigvee_{i_3}(<b_{i_3}) \vee \bigvee_{i_4}(==x_{i_4}).$$

This form can be further reduced to one of the following form based on Lemma 1:

$$\bigvee_{i_2}(>a_{i_2}) \vee \bigvee_{i_3}(<b_{i_3}) \vee \bigvee_{i_4}(==x_{i_4}), \tag{5}$$

or

$$\vee(!=s). \tag{6}$$

Note that (5) can be further reduced to the following form based on Lemma 2:

$$>a \vee <b \vee \bigvee_{i}(==x_i). \tag{7}$$

As a result, this lemma can be proved by combining (6) and (7). □

Theorem 2. (The compact forms of disjunctive clauses) *All disjunctive clauses can be reduced to one of the eight compact forms shown in Table II.*

Proof. The compact forms (1) and (2) listed in Table II can be directly obtained from the two general forms derived in Lemma 3, namely, (6) and (7). The compact forms (3) to (8) are special cases of general forms, which can be obtained by assigning ∞, $-\infty$, and 0 to the variable a and b in (7) and i in (6), respectively, as shown in Table II. □

After each disjunctive clauses are reduced to the most compact forms, the final step is to connect disjunctive clauses by conjunctive logical operator ($\wedge$) and applying unification rules to each pair.

The unification procedure of NNPE is the same as that of MNPE except for the $\mathbb{N}$ segment. The rules for unifying $\mathbb{N}$ segments are listed in Table III. If the semantics of terms in $\mathbb{N}$ are the same, then the segment $\mathbb{N}$ is directly adopted. On the contrary, the terms are removed if any conflict exists. If it is neither of the two cases, then the segment $\mathbb{N}$ of the master expression is chosen.

V. Evaluation

According to a recent survey of 24 existing service composition frameworks in pervasive environments [2], 17 of them are categorized as Type-based Service Composition (TBSC), since they only compose the service by simply matching node types; the remaining 7 of them match the values of attributes against a set of user-specified expression, which are called Expression-driven Service Composition (EDSC). In the following, the proposed approach is called the Negotiable-Expression-driven Service Composition (NESC). This work evaluates the quality of the above mentioned approaches based on several metrics which will be reported in the following sub-sections.

All experiments are conducted on P4 1GHz CPU PCs with 1GB memory and all input data are randomly generated to simulate the real world situation. In each experiment, the number of Service Request is set to 1000, the lengths of services are randomly distributed from 3 to 5, and there are totally 15 node types in the system. Each composition method is performed to select candidates among a group of Worker Nodes ranging from 500 to 1500 instances and each node consists of 7 to 11 attributes. Among these attributes, 50% of them are constrained by user preferences. By default, the number of mandatory preferences is equal to the number of negotiable preferences.

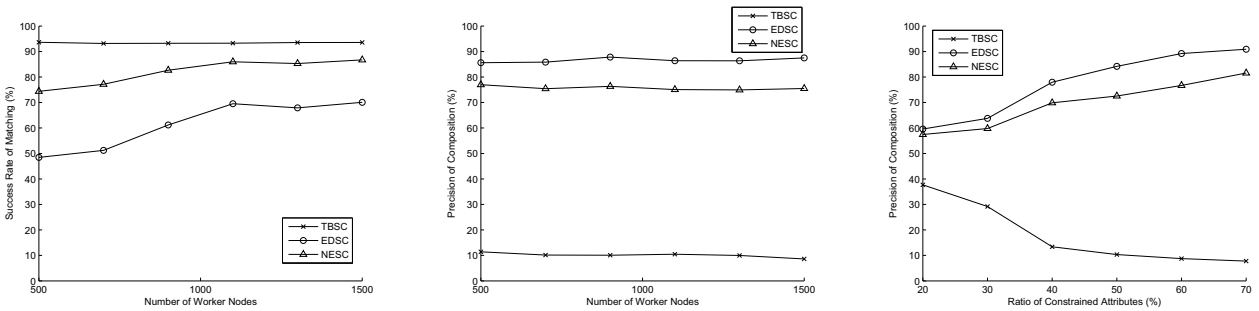
1) Success Rate of Matching: The first metric is called the Success Rate of Matching (SRM), which is defined below:

$$SRM = \frac{\sum_{s\in S} n(W_s^{valid})}{\sum_{s\in S} n(W_s^{requested})} = \frac{\sum_{s\in S} n(W_s{}^{valid})}{\sum_{s\in S} \ell(s)}, \tag{8}$$

where W_s^{valid} is set of Worker Nodes that are successfully found and matched for the service s, S is the set of services to be composed, and $\ell(s)$ is the average number of components in a service.

The core idea of SRM is to measure the success rate based on the number of successfully found nodes instead of the number of successfully composed services. When a service needs to be composed and if there are $n-1$ out of n nodes which are found, then $\frac{n-1}{n}$ is given to SRM instead of 0. Figure 3a shows SRM with different number of Worker Nodes. Both the SRMs of EDSC and NESC slightly increase when the number of nodes is increased. TBSC has the highest SRM. However, a composition mechanism with high SRM does not guarantee high quality of service. In an extreme case, a composition mechanism can achieve high SRM by simply reporting that all nodes are candidates. As a result, a metric that measures the precision of the timing for reporting candidates is required. Recently, many researchers found that "recall" and "precision", which are common evaluation measures in the information retrieval field [18], are very useful metrics for evaluating the quality of service composition [19], [20]. NESC only returns the best node, so that $n(W_s^{requested})$ represents the total number of relevant nodes of a Service Request. Therefore, the semantics of SRM is identical to the concept of "recall" which is the number of relevant items retrieved (i.e. $\sum_{s\in S} n(W_s^{valid})$) over the number of total number of relevant items (i.e. $\sum_{s\in S} n(W_s^{requested})$). In the following, the metrics for measuring the precision of a composed service will be presented.

2) Precision of Composition: Precision is the number of relevant items retrieved over the number of total retrieved items [18]. Hence, the Precision of Composition (PoC) can



(a) (b) (c)

Fig. 3: (a) Success Rate of Matching with number of nodes; (b) Precision of Composition with number of nodes; (c) Precision of Composition with the ratio of constrained attributes ;

be defined as the number of valid nodes retrieved over the number of total retrieved nodes, namely,

$$PoC = \frac{\sum_{s \in S} n(W_s^{valid})}{\sum_{s \in S} n(W_s^{found})}, \tag{9}$$

where $n(W_s^{valid})$ is the number of nodes that fulfills the corresponding Node Preference Descriptors and $n(W_s^{found})$ is the number of nodes found by the PSM.

Figure 3b illustrates the PoC of the three approaches with increasing number of nodes. The PoCs of TBSC, EDSC, and NESC are steady at 10%, 78%, and 86%, respectively. It is important to note that the although TBSC gets high SRM in the previous experiments, it suffers from extremely low PoC. In other words, TBSC tends to retrieve too many candidates causing the precision being extremely low. On the contrary, EDSC is too restrictive so that, although it gets the highest PoC, the SRM is poor. Figure 3a and 3b show that NESC is able to maintain high score both in SRM and PoC. Specifically, the NESC is precise enough so that it is able to compose high quality services while maintaining reasonable success rate of composition. From Fig. 3b, given that the number of constraints on node attributes are the same, PoC is independent of the number of the Worker Nodes. Therefore, additional experiments are performed to observe the relationship between PoC and the ratios of constrained attributes. The outcomes are shown in Fig. 3c. Observe that the PoCs of TBSC drop rapidly whereas PoCs of other approaches increase gradually. The results show that TBSC is more inappropriate if there are more constraints on attributes.

3) User Satisfaction Index (USI): Finally, we use the F_β-score of SRM and PoC to indicate the overall quality of composition. F_β-score is a popular method used to combine the precision and recall metrics [21], where β is a weight parameter used to adjust the importance between precision and recall. In fact, F_1-score is the harmonic mean of precision and recall. The following equation defines a F_β-based metric called User Satisfaction Index (USI) by integrating the outcomes of SRM and PoC:

$$USI(F_\beta) = (1 + \beta^2) \cdot \frac{PoC \cdot SRM}{(\beta^2 \cdot PoC) + SRM}. \tag{10}$$

In real cases, a composition method with low success rate (SRM) usually leads to frustrating user experiences. Users' preferences are usually adjustable, dynamic, and vague so that they are usually willing to negotiate, that is, to adjust their preferences, in order to prevent the composition from failing. When evaluating composition methods, one can put more emphases on success rate (SRM) by increasing β.

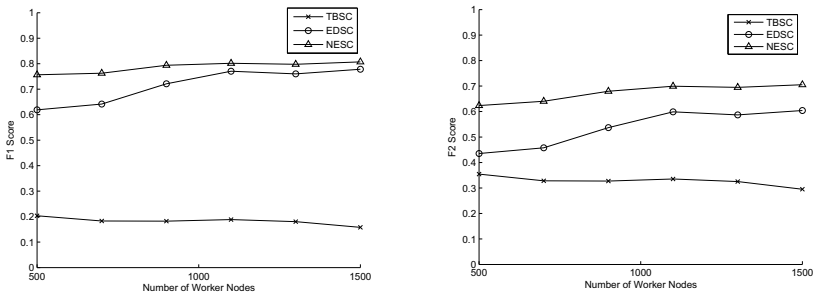
Figures 4a and 4b show the USIs of the three methods when $\beta = 1$ and $\beta = 2$, respectively. The results show that the proposed approach obtains the highest score both in $QoC(F_1)$ and $QoC(F_2)$. When $\beta = 2$, where the metric is in favor of the approaches with higher success rate, The USI score of NESC is obviously much higher than that of EDSC.

It can be concluded that the proposed approach, namely, NESC, is able to achieve high composition precision and maintains reasonable success rate of composition at the same time so that it outperforms the other methods in both $USI(F_1)$ and $USI(F_2)$ metrics.

VI. Conclusion

In a pervasive environment, the criteria for selecting and ranking services are usually specified by users, which tend to be vague and subjective, causing the criteria for selecting services can be contradictory. This paper proposes a negotiable and unifiable expression language, namely, Preference Expression, along with a set of unification rules for integrating conflicting preferences. Experimental results show that the proposed approach is able to greatly increase the success rate of composition especially under strict constraints. Although the proposed approach is presented based on PerSAM, the the core techniques such as Preference Expression and its unification rules are generally applicable to other service models with minor modifications.

Ongoing research will be under way to address the following issues. First, the constraints on preferences may involve more than one variable at the same time, but the current version



(a) (b)

Fig. 4: (a) User Satisfaction Index ($\beta = 1$); (b) User Satisfaction Index ($\beta = 2$)

of unification rules does not consider the co-relation among attributes. Besides, many services in Smart Homes contain "contents", that is, digitized media such as texts, images, videos and voices that are able to be processed by computers. Currently, the "contents" is modeled as an attribute of a service. Nevertheless, from a user's point of view, services with different digital contents should be distinguished from one another. For instance, a media player playing different movies provide different user experiences. In other words, the information of contents should be taken into account when selecting and ranking services besides types and QoS attributes of services. Further research is also under way to investigate this type of content-aware services.

ACKNOWLEDGMENT

This work is supported by National Science Council, National Taiwan University and Intel Corporation under Grants NSC99-2911-I-002-001, 99R70600, and 10R70500, respectively.

REFERENCES

[1] L. J. Zhang, J. Zhang, and H. Cai., *Service Computing*. Springer and Tsinghua University Press, 2007.

[2] J. Bronsted, K. M. Hansen, and M. Ingstrup, "Service Composition Issues in Pervasive Computing," in *IEEE Pervasive Computing*, 2010.

[3] L. J. Zhang, J. Zhang, and H. Cai. Service Computing. Springer and Tsinghua University Press, 2007.

[4] K.Arnold, B.O'Sullivan, R.Scheifler, J.Waldo, and A.Wollrath. *The Jini Specification*. Addison-Wesley, 1999.

[5] The OASIS UDDI Specifications, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[6] M. Merabti, P. Fergus, O. Abuelma'atti, H. Yu, and C. Judice, "Managing Distributed Networked Appliances in Home Networks, " in *Proceedings of the IEEE*, Vol.96, No. 1, 2008.

[7] J. P. Sousa, V. Poladian, D. Garlan, B. Schmerl, and M. Shaw, "Task-Based Adaptation for Ubiquitous Computing, " in *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, Vol.36, No.3, 2006.

[8] Y. Zhang, S. S. Zhang, and S. Q. Han, "A New Methodology of QoS Evaluation and Service Selection for Ubiquitous Computing, " in *Proc. International Conference on Wireless Algorithms, Systems, and Applications, LNCS 4138*, Springer-Verlag, 2006.

[9] A. Bottaro and R. S. Hall, "Dynamic Contextual Service Ranking, " in Proc. of International Conference on Software Composition, LNCS 4829, Springer-Verlag, 2007.

[10] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran, *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*, W3C Recommendation, 2004.

[11] G. Lee, P. Faratin, S. Bauer, and J. Wroclawski, "A User-Guided Cognitive Agent for Network Service Selection in Pervasive Computing Environments," in *Proceedings of IEEE International Conference on Pervasive Computing and Communications*, 2004.

[12] M. I. Mandel, G. E. Poliner, and D. P. W. Ellis, "Support Vector Machine Active Learning for Music Retrieval," in *ACM Journal of Multimedia System*, Vol.12, No.1, pp.3-13, 2005.

[13] A.Ranganathan, S.Chetan, J.A.Muhtadi, R.H.Campbell, M.D.Mickunas, "Olympus: A High-Level Programming Model for Pervasive Computing Environments," in *Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications* (PerCom'05), pp.7-16, March 2005.

[14] C.Bettini, and D.Riboni, "Profile Aggregation and Policy Evaluation for Adaptive Internet Service," in *Proceedings of IEEE International Conference on Mobile and Ubiquitous Systems*, Boston, Massachusetts, USA, pp.290-298, 2004.

[15] C.S.Shankar, A.Ranganathan, and R.Campbell, "An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments," in *Proceedings of IEEE International Conference on Mobile and Ubiquitous Systems*, San Diego, CA, USA, pp.33-42, 2005.

[16] C. F. Liao, Y. W. Jong, L. C. Fu, "Toward Reliable Service Management in Message-Oriented Pervasive Systems," in *IEEE Transactions on Service Computing*, to appear. http://doi.ieeecomputersociety.org/10.1109/TSC.2010.59

[17] J. W. Backus, "The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference," In *Proc. of International Conference on Information Processing*, pp.125-132, 1959.

[18] C. D. Manning, P. Raghavan, and H. Schutze, *An Introduction to Information Reterival*, Cambridge University Press, 2009.

[19] E. Silva, L. F. Pires, and M. v. Sinderen, "A framework for the evaluation of semantics-based service composition approaches," In *Proc. of 7th IEEE European Conference on Web Services (ECOWS)*, 2009.

[20] M. Sathya, M. Swarnamugi, P. Dhavachelvan, and G. Sureshkumar, "Evaluation of qos based web-service selection techniques for service composition," in *International Journal of Software Engineering*, Vol.1, No.5, pp.73-90, 2010.

[21] C. J. V. Rijsbergen. *Information Retrieval. Butterworth*, second edition, 1979.