

# Defeasible preferences for intelligible pervasive applications to enhance eldercare

Johnson Fong<sup>1,2</sup>, Ho-Pun Lam<sup>1,2</sup>, Ricky Robinson<sup>1,2</sup> and Jadwiga Indulska<sup>1,2</sup>

<sup>1</sup>The University of Queensland

School of Information Technology and Electrical Engineering

<sup>2</sup>National ICT Australia (NICTA)

Queensland Research Laboratory, Australia

{jfong, jaga}@itee.uq.edu.au, {brian.lam, ricky.robinson}@nicta.com.au

*Abstract*— Adaptation decisions made by context-aware applications on behalf of users are based on evaluations of current context and preferences of users. This context information is imperfect by nature and can cause applications to behave in ways that users do not expect. Applications that exhibit unwanted behaviour will negatively impact their usability and violate the trust users have in them. Intelligibility and control in applications can help users to understand why they decided to behave in certain ways, and to forgive the applications by enabling users to override the undesirable adaptation. This paper presents a non-monotonic rule based approach (defeasible logic) for modelling user preferences, which serves as the basis of decision-making of application adaptations. It facilitates automatic generation of explanations regarding reasoning of defeasible preferences. The model also supports creation of feedback mechanisms for non-technical users to formulate their own preferences independently, and modify the adaptation decision process to control application behaviours. Moreover, to demonstrate its applicability we have designed a set of evolvable situations and a context model, which complement the defeasible preferences for building smart home applications to enhance health care of the elderly.

**Keywords** - context-awareness; intelligibility; user control; context modelling; preference modelling

## I. INTRODUCTION

Context-aware applications employ implicit sensing and complex reasoning mechanisms to try to understand the current context and situations of users, and make appropriate adaptations according to users' preferences. Unfortunately, such adaptations do not always result in behaviours that users expect, due to variability in human preferences and imperfect sensing of context information, etc. Applications that exhibit unwanted behaviour will negatively impact their user experience and violate the trust users have in them, leading to rejections. This call for *intelligibility* in context-aware applications, that is, *being able to represent to their users what contexts/situations the applications have inferred, how the reasoning took place, and what adaptation has been performed as a result of the evaluation* [12]. Based on this information, users may also want to be able to *control* the applications, that is, *to modify settings/thresholds to personalise the adaptive behaviours according to their preferences*. Intelligibility helps users to better understand the application by providing them explanations to its decision-making of adaptations. This serves as a foundation for enabling users to control the application and override any unwanted behaviour with appropriate feedback mechanisms. The aim is to maintain

the users' trust and impression of the applications after they behaved erratically.

## A. Contributions

In this paper, we present a non-monotonic rule based approach (defeasible logic [9]) for modelling context-dependant preferences, which serves as the basis of decision-making of application adaptations. The user preference model is an extension to a framework that offers formal context modelling (with CML) and situation abstractions [11]. It supports application intelligibility by facilitating generations of wide range of explanation types [10] derived from reasoning of defeasible preferences. These explanations enable users to understand the implicit linkages between particular contextual situations and various adaptive actions of the applications.

The model also supports user control of application behaviours by assisting developers in the creation of appropriate feedback mechanisms to override any unwanted behaviour. We describe methods for rendering defeasible preferences to users in a readable representation, capturing their feedback/changes, and mapping the modifications back to the preferences, which in turn changes the application behaviours. In addition, to demonstrate the types of applications that can be created with this model, the paper presents a set of evolvable situation abstractions and a context model designed to go together with defeasible preferences for building smart home applications to enhance health care of the elderly.

The remainder of the paper is organized as follows. Section 2 provides backgrounds and examples of defeasible preferences. Section 3 discusses the methods [7] we adapt for explaining the reasoning of defeasible preferences. Section 4 presents situation abstractions and a CML context model required by the preferences for developing a smart home application. Section 5 discusses the related work on pervasive frameworks that support development of intelligible applications. Finally, section 6 summarises the paper with discussion on future work.

## II. PREFERENCE MODEL BACKGROUND

User preferences are indispensable in supporting decision-making processes of context-aware applications, as they capture requirements on how users would prefer the applications to behave in certain situations. This section presents the basics of Defeasible Logic (DL) which we employ for modelling user preferences. DL [9] is a simple

and computationally efficient rule based non-monotonic (sceptical) formalism for reasoning with incomplete and inconsistent information. Its main advantages over other mainstream non-monotonic reasoning formalisms are: low computation complexity (linear time w.r.t. the size of a theory), and its built-in preference handling facilities allowing one to derive plausible conclusions from incomplete and contradictory information in a natural and declarative way [6]. It has also been argued that it is suitable to represent and reasons with norms [13, 23-25].

A *defeasible theory*  $D$ , i.e., a knowledge base in DL, is a triple  $D = (F, R, >)$  where  $F$  is a finite set of indisputable facts,  $R$  is a finite set of rules, and  $>$  is an acyclic superiority relation on  $R$ . The language of DL consists of a finite set of literals, where a literal is either an atomic proposition or its negation. Given a literal  $L$ ,  $\sim L$  denotes its complement.

*Facts* denote indisputable statements that are deemed to be true. In pervasive computing terminology, facts can be considered as situations that hold true. E.g., an occupant has fallen in a Smart Home. This is expressed as `OccupantHasFallen (Mary)`.

A rule  $r$  in  $R$  is composed of an *antecedent* (body)  $A(r)$  and a *consequent*  $C(r)$ , where  $A(r)$  consists of a finite set of literals and  $C(r)$  contains a single literal.  $A(r)$  can be omitted from the rule if it is empty. There are three types of rules in  $R$ : *strict rules* ( $r: A(r) \rightarrow C(r)$ ), *defeasible rules* ( $r: A(r) \Rightarrow C(r)$ ), and *defeaters* ( $r: A(r) \Leftarrow C(r)$ ).

*Strict rules* are rules in the classical senses, the conclusion follows every time the antecedents hold. E.g., “When an occupant has fallen, contact her daughter”, which is written as:

`OccupantHasFallen (Mary) → Contact (NextOfKin)`

*Defeasible rule* is allowed to assert its conclusion in case there is no contrary evidence to the conclusion. E.g., “When an occupant has fallen, call medical emergency”, which is written as:

`OccupantHasFallen (Mary) ⇒ CallEmergency (medical)`

*Defeaters* cannot support conclusions but they provide contrary evidence to them. E.g., “If the occupant is conscious, it is not necessary to contact emergency”, which is written as:

`Conscious (Mary) ⇐ ¬CallEmergency (medical)`

A *superiority relation* ( $>$ ) describes the relative strength of rules, and it is used to obtain a conclusion when there are applicable conflicting rules. When  $r1 > r2$ , then  $r1$  is called superior to  $r2$ , and  $r2$  inferior to  $r1$ , meaning that the conclusion of  $r1$  may override the conclusion  $r2$  if both of them are applicable. E.g., “If the occupant has fallen, contact emergency, unless she is standing up”, which is written as:

`r1: OccupantHasFallen (Mary) ⇒ CallEmergency (medical)`  
`r2: Standingup (Mary) ⇒ ¬CallEmergency (medical)`  
`r2 > r1`

DL is able to distinguish positive conclusions from negative conclusions, that is, literals that can be proved and literals that are refuted. In addition, it is able to determine the strength of a conclusion, i.e., whether something is concluded using only strict rules and facts or whether we have a defeasible conclusion - a conclusion that can be retracted if more evidence is provided. Accordingly, for a literal  $p$  in  $D$ , we have the following four types of conclusions, called tagged literals:  $+\Delta p$  ( $p$  is definitely provable in  $D$ ),  $-\Delta p$  ( $p$  is definitely rejected in  $D$ ),  $+\hat{c}p$  ( $p$  is defeasibly provable in  $D$ ),  $-\hat{c}p$  ( $p$  is defeasibly rejected in  $D$ ).

At the heart of DL we have its proof theory that tells us how to derive tagged literals. A *proof* is a sequence of tagged literals obeying proof conditions corresponding to inference rules. The inference rules establish when we can add a literal at the end of a sequence of tagged literals based on conditions on the elements of a theory and the previous tagged literals in the sequence. The structure of the proof conditions has an argumentation flavour. For example, to prove  $+\hat{c}p$ :

Phase 1: There is an applicable rule for  $p$  and

Phase 2: For every rule for  $\sim p$  (the complement of  $p$ ) either

Sub-Phase 1: the rule is discarded, or

Sub-Phase 2: the rule is defeated by a (stronger) rule for  $p$

The notion of a rule being applicable means that all the antecedents of the rule are provable (with the appropriate strength); a rule is discarded if at least one of the antecedents is refuted (with the appropriate strength), and finally a rule is defeated, if there is a (stronger) rule for the complement of the conclusion that is applicable (again with the appropriate strength). The above structure enables us to define several variants of DL [8] - such as ambiguity blocking and ambiguity propagation - by giving different parameters (i.e., this is what we mean ‘with the appropriate strength’ in the previous paragraph). Due to the limited space, readers interested in this subject please refer to [17, 25] for details.

#### A. Preference Example

In this section, we illustrate some sample user preferences being applied to a smart home that provides context-aware services designed to assist elderly in maximizing independence and maintaining a high quality of life. A preference `General_Pref` is shown in figure 1 that illustrates the usage of a defeater in a defeasible preference.

```

General_Pref:
r1: MaintenanceRequired() → arrange (Handyman)
r2: SecurityBreached() → soundAlarm ()
r3: MealPrepared() ⇒ initiateDistanceDinning (Relative)
r4: Working (Relative) ⇐
    ¬initiateDistanceDinning (Relative)
r5: GlycosuriaDetected() → remindMedication (Diabetic)
r6: OccupantHasFallen () ⇒ queryStatus (Occupant)

```

Figure 1. Example of a defeasible preference with a defeater.

Overall the rules in the preference are somewhat self-explanatory: Strict rule r1: when there is a maintenance needed to be fixed (e.g., light bulbs burnt out or water leakages somewhere), book a time with a handyman. Strict rule r5: when glucose is detected in occupant’s discharge remind her of diabetic medication. Defeasible rule r3 and defeater r4: initiate a social distance dinner with occupant’s relative when a meal is ready unless the relative is working.

A preference `MobileCall_Pref` is shown in figure 2, which illustrates the usage of a superiority relation in the defeasible preference. The evaluation of this preference can be initiated by users as well as triggers. It can be initiated by users as they wish to make an outgoing call, or initiated by a trigger when there is a change in signal strength of a WiFi network during a VOIP conversation, so as to perform vertical handover decisions over network interfaces. The preference indicates that if a user is currently having a video call over the WiFi network, and the signal strength suddenly turns weak (r2), then the call is switched from video to voice mode. If the WiFi signal is minimal, then the call is handed over to the 3G network with video mode (r3), unless the cost of internet access on the 3G network is high, then use voice only instead (r4 and r5).

```

MobileCall_Pref:
r1: WLAN(Strong) ⇒ UseWLANvoip(Video)
r2: WLAN(Weak) ⇒ UseWLANvoip(Voice)
r3: WLAN(Min) ⇒ UseUMTSVoip(Video)
r4: WLAN(Min), UMTSBandwidthCost(High) ⇒
    -UseUMTSVoip(Video)
r5: WLAN(Min), UMTSBandwidthCost(High) ⇒
    UseUMTSvoip(Voice)
r4>r3
  
```

Figure 2. A defeasible preference with a superiority relation

### III. UNDERSTANDING PREFERENCE EVALUATION

This section presents a method we employ from Bassiliades et al. [5, 7], to extract meaningful explanations from reasoning of defeasible preferences. The aim is to generate some of the explanation types recommended by [10], such as *what*, *why*, *why not* and *How to*, etc, as shown in Figure 3, to explain to users the outcome of a preference evaluation.

Explanation Types	Explanations
What is it doing?	UseUMTSvoip(Voice) - The conversation has handed over from a WiFi Video call to a 3G Voice call.
Why?	UseUMTSvoip(Voice) - Because the <b>WiFi signal is minimal</b> , and the <b>data cost is high for the 3G network</b> .
Why Not perform?	
UseWLANvoip(Video)	Not WiFi Video Call - because WiFi signal is <b>not strong</b> .
UseUMTSvoip(Video)	<i>Not 3G Video call - Although WiFi signal is minimal, the action is blocked when the data cost is high for the 3G network.</i>
How To perform	
UseWLANvoip (Voice)	WiFi Voice Call when WiFi signal is Weak.
Control	User may edit situations and preferences using a User Feedback System

Figure 3. Explanations generated from a reasoning of the preference `MobileCall_Pref`

To illustrate how each explanation type is achieved, we begin with a sample evaluation of the preference `MobileCall_Pref` (figure 2) triggered by a drop of WiFi signal strength. Assuming the resulting adaptation was `UseUMTSvoip(Video)`, which hands over an occupant’s phone conversation from a WiFi video mode to 3G voice mode. After the conversation, the occupant wanted to find out why her call was not switched to 3G video mode instead. And it was simply because the 3G video mode (i.e., `UseUMTSvoip(Video)`) is blocked due to the *high data cost for the 3G network*, although the *WiFi signal was minimal*.

In order to generate this *why not* explanation from the evaluation of `MobileCall_Pref`, the model employs the explanation methods discussed in Bassiliades et al. [5] and Antoniou et al. [7], which are capable of generating XML schema for *proof trace* to explain the reasoning of defeasible preferences. The proof trace shown in figure 4 is an extension of RuleML’s 0.91 schema that provides explanation for why the conclusion `UseUMTSVoip(Video)` is *not defeasibly provable* ( $-\partial p$ ). The trace indicates that the conclusion `UseUMTSVoip(Video)` of rule r3 was applicable due to a fact `WLAN(Min)`, however, the conclusion was attacked by a superior rule r4 (which is proven due to facts `UMTSBandwidthCost(High)` and `WLAN(Min)`).

```

<RuleML rulebase="http://.../proof/ex/ex1.ruleml"
xsi:schema="http://www.ruleml.org/0.43/xsd http://...
<Not_Defeasibly_Proved> <oid><Ind
uri="&pr_ex;proof1">proof1</Ind></oid>
<Literal> <RDF_resource uri="http://.../export-
ex1.rdf#UseUMTSvoip(Video)"/>
<Not_Defeasible_Proof>
<supportive_rule> <rule_ref rule="&ex_rb;r3"/>
</supportive_rule>
<defeasible_body_grounds>
<Defeasibly_Proved> <Literal> <Atom> <slot>... ...
<Definite_Proof>
<strict_clause>
<Fact> <RDF_resource uri="http://...
ex1.rdf#WLANMin"/> </Fact>
</strict_clause>
</Definite_Proof>
</defeasible_body_grounds>
</Not_Defeasibly_Proved>
<strongly_attacked>
<rule_ref rule="&ex_rb;r3"/>
<Attacked_by_Superior> <rule_ref rule="&ex_rb;r4"/>
<defeasible_body_grounds>
<Defeasibly_Proved> <Literal> <Atom> <slot>... ...
<Definite_Proof>
<strict_clause>
<Fact> <RDF_resource uri="http://...ex1.
rdf#UMTSBandwidthCostHigh"/> </Fact>
<Fact> <RDF_resource uri="http://...
ex1.rdf#WLANMin"/> </Fact> ...
  
```

Figure 4. Example of a proof trace for explaining why `UseUMTSVoip(Video)` was not defeasibly provable ( $-\partial p$ ).

Given the trace for  $-\partial$  `UseUMTSvoip(Video)` (a not defeasibly provable conclusion), the model is able to generate the readable *why not* explanation phrase as shown in Figure 4, by retrieving descriptions of corresponding situations and concatenating them with appropriate strings using a field replacement approach [15].

Also, to produce the *why* explanations, which inform users why did the application derived a particular conclusion, such conclusion has to be either defeasibly provable ( $+\partial p$ ) or definitely provable ( $+\Delta p$ ). Depending on the tagged literal, a proof trace is generated. Finally, to generate the *How to* explanations, which inform users how can the application produce an alternative conclusion, we simply select the rule that links to the alternative conclusion, and returns the description of each situations that comprises the rule. These descriptions are then concatenated with appropriate keywords (e.g., +because+, +when+, and +Not+, etc.) using the field replacement approach to produce readable explanation phrases.

### A. Modifying Preferences

Given various explanations, users are able to provide feedback (i.e., modifying a particular preference) to control the application’s behaviour via a user feedback interface. Instead of showing users the underlying low-level representation of a preference in defeasible logic, the model maps the defeasible preference to an easily understandable When-Do representation, as shown in figure 5. For each preference, a mapping algorithm combines literals in related defeasible rules with AND and OR operators thereby effectively reducing the amount of rules in a preference, and provides a higher level of abstraction that is easier for user to understand how the preference works in general. The model is also responsible for mapping the user’s modifications from the interface back to the preferences. Should any conflict arises, it is resolved by automatically assigning superiority among conflicting rules. Superiority is assigned by assuming rules that are defined closer to the bottom are more specific, therefore, they have higher priority than rules defined at the top of the preference.

## IV. CONTEXT MODEL AND SITUATION

This section illustrates a context model and some of the situations required by the defeasible preferences for developing smart home services to support health monitoring of elderly. The context model, as presented in figure 7, captures low-level context information gathered from sensors and abstracts it into context facts that are required by the smart home application. It is modelled using CML [11], an object-role based graphical notation that assists designers with the tasks of exploring and specifying the context requirements of the application. It supports the application to reason about the current context and to derive higher level context information (i.e., situations).

Situations are abstractions of the events occurring in the real world derived from context facts. They are specified using first order logic with a combination of facts supported by universal quantifiers and/or existential quantifiers as shown in figure 6. For example, the situation definition `Sharpfall (Occupant)` indicates the situation will return true when an accelerometer worn by the elderly has a downward acceleration greater than  $5.8\text{m/s}^2$ . This situation

can be combined with another situation `PersonIsHorizontal (person)` to detect if an occupant has fallen as in `OccupantHasFallen(person)`, which will hold true when the occupant experienced a sudden downward acceleration, and was lying down not on her bed. These situations are then used by the defeasible preferences (e.g., figure 1) to provide linkages between various contextual situations and adaptive actions of the application.

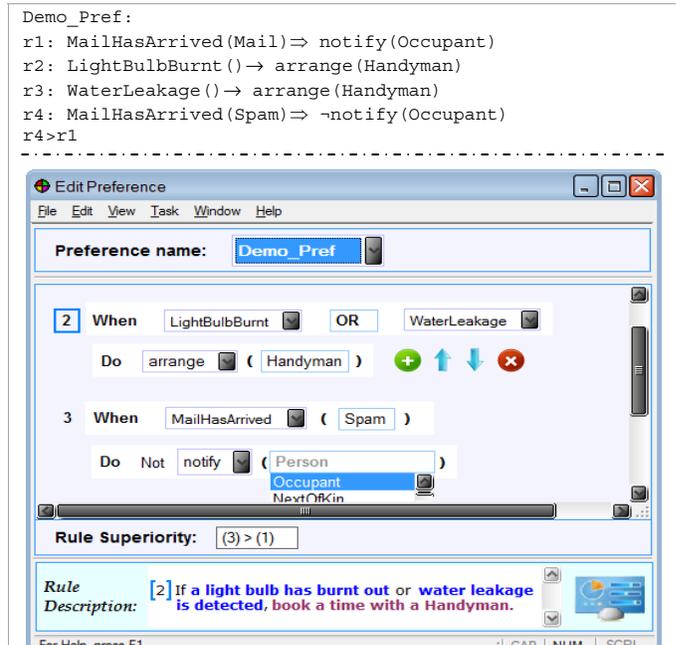


Figure 5. A mapping of a defeasible preference to a user feedback interface.

```

GlycosuriaDetected (person) :
∃ SpaceId • hasLocation [person, SpaceId]
  SpaceId=Lavatory AND
  ∃ unusualSubstance • detectsSubstance [SpaceId,
  unusualSubstance]
  SpaceId=Lavatory
  unusualSubstance = glucose

OccupantHasFallen (person) :
SharpFall(person) AND PersonIsHorizontal(person) AND
  ∃ room • hasLocation [person, spaceId]
  SpaceId != bedroom

Sharpfall (person) :
  ∃ Movement • HasAcceleration [movement, acceleration]
  acceleration > 5.8 m/s^2
  • HasMovement [person, movement, t1, t2]
  (t1 ≥ (timenow() - 5 sec) )
  • HasVector[movement, vector]
  vector = Downward

workingHrs (person) :
  ∃ day,t1, t2 • hasWorkingDays [person, day, t1, t2]
  • t1 < timenow() < t2
  • day = timenow().day

```

Figure 6. Situations required by the preferences to support eldercare.

## V. RELATED WORK

This section briefly reviews relevant preference modelling approaches adapt by pervasive frameworks to support intelligibility of context-aware applications. There has been

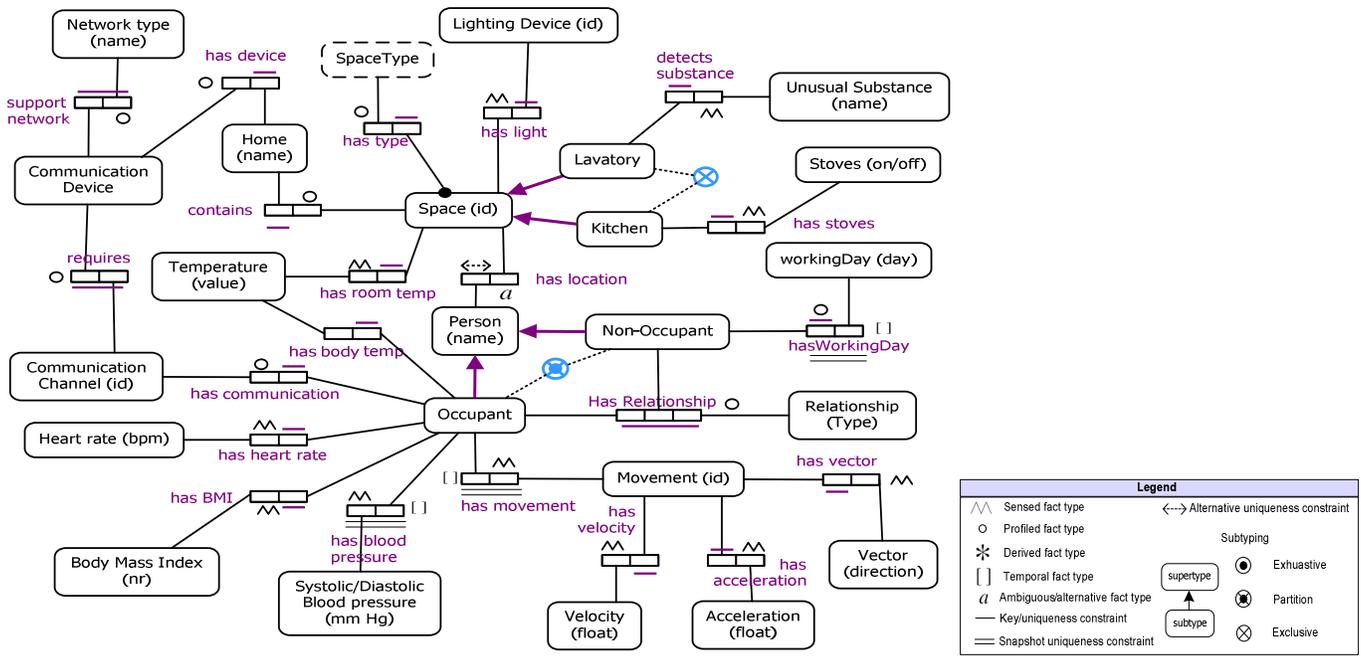


Figure 7. A CML context model for capturing context information to support independent living of elderly

considerable amount of research into pervasive middleware frameworks. Much of it has been driven by the technical aspects, such as aggregating and managing context information, focusing on specific problems e.g., fault-tolerance, resource discovery, user privacy and security, etc. While some of frameworks do provide mechanisms for modelling user preferences and evaluating user policy, for example: CARE [1] uses policy rules composed by a set of conditions on profile data represented using CC/PP attributes [20]; and Costa et al. [18] adapts a domain specific language ECA-DL for the purpose of specifying context-aware reactive rules. However, only a few has explored ways to help users to understand and control the operation of such frameworks: an extension to the context toolkit, known as Intelligibility Toolkit [4], can explain key forms of reasoning within the framework into various explanation types that are of interest to users; PersonisAD [19] supports *What* explanations by resolving identities and associations of devices, locations, people, etc; Hardian et al. [3] has developed application interfaces for the PACE framework[11] to expose the internal workings of applications, but users do not necessary understand what is being revealed to them as the original preference model was designed for developers. Hence, we are extending PACE with defeasible preferences aimed to provide a more user-centred version for supporting explanations generation and user customisation of adaptation decision processes. There have been some works on modelling user preferences using defeasible logic in pervasive computing [16]. Table 1 presents a brief comparison in various dimensions (time complexity, maturity, features, conflicts resolution and usability) between defeasible preferences and the original PACE preferences.

	Defeasible Preferences	PACE Preferences
Reasoning Complexity	Both have Linear Time $O(n)$ [6] assuming the complexity of the function employed in PACE for aggregating choice	

Maturity	Proposed by Nute in 1993 [9]. Since then, efficient implementation exists [21], and the most recent implementation fully support Semantic Web standards (RDF, RuleML) [14] and extensions to deal with normative condition have been proposed [2].	Proposed by Agrawal and Wimmers in 2000 [22] as quantitative preference model. Later extended by Henricksen for combining adaptation choices using utility functions.
Features	<ol style="list-style-type: none"> <li>1Forbid can be emulated by using defeaters to establish a negation of a conclusion.</li> <li>2Oblige is supported by default when the conclusion is not blocked by another rule (using a defeater or a superiority relation).</li> </ol>	<ol style="list-style-type: none"> <li>1Forbid is supported by assigning "Forbid" as the rating of an adaptation choice.</li> <li>2Oblige is supported by assigning "Oblige" as the rating of an adaptation choice.</li> </ol>
Resolving Inconsistent information	Superiority relations are used to resolve conflicts among competing rules. However, a literal will become ambiguous if there is a chain of reasoning that supports a conclusions that p is true, and another the support $\sim p$ is true, and superiority relation does not resolve the conflict. Then under this situation, DL will falsify both of them.	When a choice is assigned ratings of both Forbid and Oblige, the combined rating will be Oblige.[11]
Resolving Incomplete information	A literal is considered to be not provable if it does not appear as a head of any rules inside a theory	When a situation in a preference is undefined /incomplete, it is deemed false. Thus, the preference is not applicable.
Usability	Please refer to note <sup>3</sup> (3)	Please refer to note <sup>4</sup> (4)

Table 1. Comparison between defeasible preferences and PACE preferences

<sup>1</sup> When an adaptation is Forbade, that means, the adaptation should not be performed when the corresponding situations hold.

<sup>2</sup> When an adaptation is Obliged, that means, the adaptation must be performed when the corresponding situations hold.

<sup>3</sup>Certain user awareness is required when inserting/modifying a rule. Users need to be aware of how they want the application to behave (e.g., what application should do under certain conditions), and also manually assign superiorities to resolve conflicts at the time of preference creation. Therefore, the reasoning outcome is usually user expected. Given some facts, it is not hard for users to figure out what the conclusions are, or why a particular conclusion is provable or not. Also, there exist explanation mechanisms for explaining the proof theory of a conclusion.

<sup>4</sup>When inserting/modifying a preference, users are only required to come up with a rating - an approximation of desirability of an adaptation choice. Then the system automatically combines all ratings in the preference set with a utility function to resolve any conflicts and derive an outcome. This place a lower cognitive load on users, but the trade off is that users cannot be certain as to what the final adaptation is. I.e., given evaluations of situations, users would have an idea of what the final adaptation is, but not with absolute confidence.

Notes for Table 1

## VI. CONCLUSION AND FUTURE WORK

Context-aware applications do not always adapt their behaviours in ways that users expect for a variety of reasons. This can cause annoyance and hinder user objectives. This paper presents our effort to address the issue of intelligibility and user control in applications behaviours. Contributions include (1) works on using defeasible logic to model user preferences. We show (2) how explanations can be generated from reasoning of defeasible preferences, and briefly discuss (3) how preferences can be mapped to/from a user feedback interface. In addition, to demonstrate the types of applications that can be created with this model, we design (4) sample preferences, situations and a context model for developing smart homes to enhance health care of elderly. Moreover, we provide a (5) comparison in various dimensions between defeasible preferences and the original preferences model in the PACE framework that we are extending to support development of intelligible context-aware applications.

Future work involves developing a probabilistic approach for learning preferences based on users' past behaviour (observations), so applications can adapt to changes in user behaviour while minimising user interactions. The approach would enable applications to revise/update their knowledgebase with rules that have the highest probability of describing the observations. Unfortunately, the fundamental intelligibility issues still remain even when preferences are learnt dynamically. Due to variability of human preferences and imperfect nature of sensing technology, applications will still behave unexpectedly from time to time. And users will still have to intervene and interact with the system at some point (via explanations and feedback mechanisms) to correct any undesirable adaptation. Therefore, the works presented in this paper are always relevant and valuable in pervasive computing.

## ACKNOWLEDGMENT

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program; and the Queensland Government.

## REFERENCES

- [1] A. Agostini, C. Bettini, and D. Riboni, "Hybrid reasoning in the CARE middleware for context awareness," *Int J Web Eng Technol*, 5, 3–23, 2009.
- [2] H. P. Lam, G. Governatori, "The making of SPINdle", *RuleML 2009*, US.
- [3] Hardian, Indulka, Henricksen, "Exposing Contextual Information for Balancing Software Autonomy & User Control in Context-Aware Systems", *Pervasive 2008 WS*
- [4] B.Lim, A.Dey, "Toolkit to Support Intelligibility in Context-Aware Applications", *Proceedings of the 12th ACM Ubicomp '10*, pp. 13-22, Copenhagen, Denmark, 2010
- [5] N.Bassiliades, G. Antoniou, G. Governatori, "Proof Explanation in the DR-DEVICE System", *Web Reasoning and Rule Systems*, 249-258, 2007
- [6] H. P. Lam, G. Governatori, "What are the Necessity Rules in Defeasible Reasoning", *LPNMR-2011*, Vancouver, BC, Canada, May, 2011.
- [7] G. Antoniou, A. Bikakis, N. Dimareisis, M. Genetzakis, G. Georgalis, G. Governatori, E. Karouzaki, N. Kazepis, D. Kosmadakis, M. Kritsotakis, G. Lilis, A. Papadogiannakis, P. Pedititis, R. Theodosaki, D. Zeginis, "Proof explanation for a nonmonotonic Semantic Web rules language", *Data and Knowledge Engineering*, vol. 64, no. 3, pp. 662-687, 2008.
- [8] G. Antoniou, D. Billington, G. Governatori, M. Maher, and A. Rock, "A Family of Defeasible Reasoning Logics and its Implementation," in *ECAI 2000*, pp. 459–463.
- [9] D. Nute, "Defeasible logic: theory, implementation, and applications", *Proceedings of INAP 2001*, IF Computer Japan, pp. 87-114, 2001
- [10] B. Lim, and A. Dey, "Assessing Demand for Intelligibility in Context-Aware Applications", *UbiComp'09*. pp. 195-204, New York, 2009
- [11] Henricksen. K, J. Indulka, "Developing Context-Aware Pervasive Computing Applications: Models and Approach", *PMC*, 2, pp.37-64, 2006.
- [12] Bellotti.V, Edwards. K, "Intelligibility and accountability: Human considerations in context-aware systems", *HCI*, 16 (2-4), 2001, 193-212.
- [13] G. Governatori, A. Rotolo, and G. Sartor, "Temporalised Normative Positions in Defeasible Logic," in *ICAAIL '05*. New York, NY, USA: ACM, 2005, pp. 25–34.
- [14] N. Bassiliades, G. Antoniou and I. Vlahavas, "A Defeasible Logic Reasoner for the Semantic Web," *IJISWIS*, vol. 2, no. 1, pp. 1-41, 2006.
- [15] T. Halpin and M. Curland, "Automated Verbalization for ORM 2", in *OTM 2006 Workshops*, LNCS Volume 4278, pp. 1181-1190, 2006
- [16] Bikakis.A and Antoniou.G, "Distributed Defeasible Contextual Reasoning in Ambient Computing", *Ambient Intelligence LNCS*, 308-325, 2008
- [17] G. Antoniou, D. Billington, G. Governatori, M. J. Maher, "Representation Results for Defeasible Logic," *ACM Trans on Computational Logic*, 2001
- [18] P. D. Costa, J. P. Almeida, L. Pires, and J. Sinderen, "Situation Specification and Realization in Rule-Based Context-Aware Applications", *DAIS*, 2007, Cyprus.
- [19] M. Assad, D. J. Carmichael, J. Kay, B. Kummerfeld. "PersonisAD: Distributed, active, scrutable model framework for context-aware services". *Pervasive 2007*.
- [20] C. Bettini, L. Pareschi, and D. Riboni, "Efficient profile aggregation and policy evaluation in a middleware for adaptive mobile applications," *Perv. Mob. Comput.*, vol. 4, no. 5, pp. 697–718, 2008.
- [21] D. Bryant and P. Krause, "A review of current defeasible reasoning implementations," *The Knowledge Engineering Review*, Vol 23, 2008
- [22] R. Agrawal, E. Wimmers, "A framework for expressing and combining preferences", *COMAD 2000*, NY, USA
- [23] G. Antoniou, D. Billington, G. Governatori, and M.J. Maher, "On the Modeling and analysis of regulations," in *Proceedings of Australian Conference Information Systems*, 1999, pp. 20-29
- [24] G. Governatori, "Representing Business Contracts in RuleML," *Intl Journal of Cooperative Info Sys*, vol 14, no. 2-3, pp. 181-216, 2005.
- [25] D. Billington, G. Antoniou, G. Governatori and M. J. Maher, "An Inclusion Theorem for Defeasible Logic," *ACM Transactions in Computational Logic*, vol. 12, no. 1, 2010.