Context-Sensitive Call Control using Constraints and Rules

David Lesaint[1], Deepak Mehta[2], Barry O'Sullivan[2], Luis Quesada[2], and Nic Wilson[2]

[1] BT Research & Technology, BT, UK
dvdlsnt@gmail.com
[2] Cork Constraint Computation Centre, University College Cork, Ireland
{d.mehta,b.osullivan,l.quesada,n.wilson}@4c.ucc.ie

Abstract. Personalisation and context-awareness are fundamental concerns in Telephony. This paper introduces a rule-based system - 4CRULES - which enables context-sensitive call control by the means of feature configuration rules. 4CRULES is interoperable with standard context services and compositional feature architectures. It has been designed to resolve feature interactions, manage conflicting preferences, and mitigate the uncertainty affecting context data. This is achieved through a constraint optimisation model that maximises adherence to user requirements and domain constraints. Experiments on a suite of instances confirm the practicality of the approach and highlight performance- and adherence-critical factors.

1 Introduction

Telecommunications services like instant messaging or internet telephony bring increased flexibility to communicate at home, in the office or on the move. Their pervasiveness is also a source of disruptions and intrusions. Service providers are therefore looking for personalisation solutions allowing users to control the timing and modalities of their communications. In the case of telephony services, personalisation solutions are built around call control features. Technically, a feature is an increment of functionality that modifies the basic system behaviour. Dozens of call control features have been created to address concerns such as mobility, privacy, presentation, or billing. Features are optional and must be configured off-line to fulfil their role. Once configured, they execute by responding to call events (e.g., Call-Divert-On-Busy) and/or user actions (e.g., Call-Transfer) during calls.

Key requirements that drive the design of feature-rich telephony systems are: the ability for users to parametrise features (e.g., "Call-Divert to mobile"), address caller and callee scenarios (e.g., "Do-Not-Disturb and Speed-Dial"), combine or sequence features (e.g., "Call-Screen then Call-Divert"), request context-sensitive feature configurations (e.g., "Mute during seminars"), and express preferences or priorities (e.g., call policies imposed on a workforce). Different approaches ranging from scripting to policy enforcement have been proposed to meet these requirements. None, however, provides a comprehensive personalisation solution to: resolve undesirable feature interactions arising due to compositionality; manage conflicting preferences; and handle the uncertainty inherent to context data. To address these issues this paper presents a system

for Context-sensitive Configuration of Call Control features using Rules (4CRULES). 4CRULES allows a user to describe the behaviour of the communication service through a set of Feature Configuration Rules (FCRs). Conceptually, a FCR is weighted and associates a context condition to a feature subscription, which is defined to be a set of features, and a set of precedence constraints prescribed by a user and the feature catalogue. Each time a user's context is updated, the engine of 4CRULES infers a sequence of features from his/her set of FCRs. This sequence is free of undesirable feature interactions and it is applied to all calls involving the user until his context changes again.

A set of FCRs that are applicable for a given current context of a user could be inconsistent due to a variety of reasons as explained in the later sections. The 4CRULES engine computes a maximal subset of the applicable FCRs that is consistent and optimal in some sense. The optimality is based on a strict weak ordering over FCRs, which is obtained by evaluating a value for each FCR by combining priority of the FCR, concreteness of the context condition of the FCR, and probability of applicability of the FCR. The principle of the presented approach is to view FCRs processing as a constraint optimisation task. Overall, the method ensures maximum adherence to user requirements and interaction constraints. Experiments on a suite of instances confirm its practicality in terms of response time and highlight performance- and adherence-critical factors.

The next section provides background and reviews prior art on the call feature configuration. The architecture of 4CRULES and the principles of its implementation are then introduced. This is followed by a description of the FCRs language and processing method. The paper concludes with experiments using the existing implementation.

2 Related Work

From a user perspective, telephony systems that provide personalisation solutions must provide support for:

- **parameterisation**: some features use operational data supplied by users;
- **role-based selection**: some features apply to outgoing calls, some to incoming calls and others indistinctly (e.g., Call-Waiting);
- **composition**: features provide distinct functionalities and users need the flexibility to combine them and, in some cases, prioritise their execution;
- **contextualisation**: call handling requirements often depend on context, i.e., on extrinsic call characteristics such as user activity or location, and call service behaviour must be adapted accordingly;
- **preferences and priorities**: requirements may be weighted to enforce pre-emptive rights when call control is shared or to help resolve conflicts; and
- **uncertainty management**: information on context may be imprecise (e.g., coarse-grain localisation), incomplete (e.g., unclassified activity in a calendar service) or unavailable (e.g., off-line presence service).

Another critical requirement is the ability to manage feature interactions [1]. A feature interaction is "some way in which a feature modifies or influences the behaviour of another feature in generating the system's overall behaviour" [2]. Some interactions are

desirable and must be enabled to achieve proper system behaviour, e.g., when the modules of a feature must interact. Other interactions are undesirable and must be avoided. For instance, Call-Waiting-On-Busy and Call-Divert-On-Busy are triggered by the same event but take conflicting actions, i.e., put the caller on hold or divert his call.

Modern telephony systems, notably those based on the Session Initiation Protocol (SIP), rest upon compositional application architectures that are opened to personalisation and feature interaction management [3]. SIP is an application layer signalling protocol used to establish, modify and terminate multimedia sessions [4, 5]. Various application programming interfaces (APIs) and domain-specific languages (DSLs) have been proposed to develop SIP services. These capabilities hide away low-level SIP stack operations to facilitate programmatic control over the call logic.

Scripting and policy languages are the main form of DSLs for SIP services. CPL [6] and LESS [7], for instance, provide primitive events and actions to specify fine-grained call control scripts. Scripts are uploaded to devices or application servers and interpreted by scripting engines at runtime. Policy enforcement systems allow for more declarativity and expressiveness. Policies are well suited to capture trigger-response patterns that commonly define feature behaviour. APPEL, for instance, supports an event-condition-action syntax to guard call primitives with call events subject to context conditions [8, 9]. APPEL also supports conflict resolution policies [10]. Conflicts may occur when different policies are triggered and alternative actions may be taken to process a call. It is for the user to define which actions conflict and, if so, which resolution policy to apply (e.g., discard or replace actions).

These solutions mostly rely on prioritisation to handle conflicts. In addition, they provide limited support to capture constraints relating to application compositionality and they do not handle context uncertainty. Constraint-based systems offer an alternative. Distributed Feature Compositions (DFC), for instance, is an abstract network architecture designed to manage interactions through feature sequencing and exclusion constraints [11–13]. Constraints are elicited by analysing pairs of features and identifying those which are interaction-prone. This is achieved manually, formally or through simulation and testing [14–17].

Features and constraints are recorded in a catalogue and users configure their subscription by selecting and sequencing features accordingly. DFC routers then access user subscriptions to activate features sequentially when calls are set up. [18] propose a constraint optimisation approach to configure (context-agnostic) DFC feature subscriptions based on user preferences. 4CRULES builds upon this method to compute context-sensitive subscriptions.

3 Architecture and Principles of 4CRULES

4CRULES is a prototype designed to compute feature subscriptions of a user based on his/her real-time context information and feature configuration rules. It assumes a DFC-compliant telephony system that exposes a feature catalogue and accepts user subscriptions. It also assumes a context acquisition system that delivers user context information collected from different sources. Its logical architecture comprises a registrar,

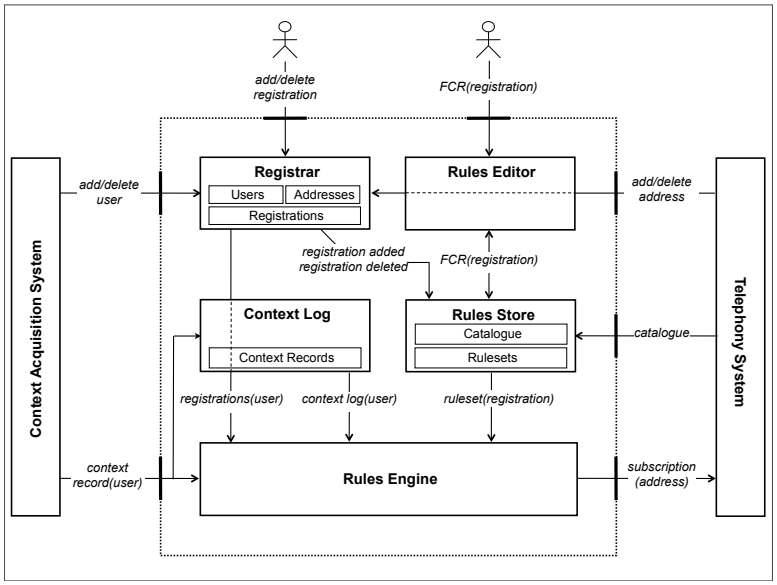

Fig. 1. The architecture of 4CRULES.

a rules-editor, a rules-store, a rules-engine, a context-log, and interfaces to the context acquisition and telephony systems - see Figure 1.

The registrar associates users of the context acquisition system with addresses registered in the telephony system. The mapping between users and addresses is generally one-to-many. For simplicity, we identify users with addresses. 4CRULES associates a set of FCRs with each user. The rules-store maintains the rule-sets and the feature catalogue. The rules editor is the user interface to create, delete and update rule-sets.

The context acquisition system is responsible for tracking the context of users. It notifies the rules-engine when the context of a user changes and passes a record of the context of the user (context-record) which is also stored in the context-log. The engine retrieves the associated set of FCRs from the rules-store and computes a context-sensitive feature subscription. It then communicates the feature subscription to the telephony system. This feature subscription will prevail in all calls involving the user until a new context-record is received.

4CRULES shares a common meta-model with the context acquisition and telephony systems. The context meta-model is simple enough to achieve interoperability with a variety of services (e.g., GPS devices, calendar services, presence servers). It prescribes a finite domain representation for each context dimension (e.g., location) which is assumed to be exhaustive (i.e., no state omitted) and unambiguous (i.e., domain values denote distinct states). The granularity of context domains is unconstrained (e.g., days or quarters). Table 1 provides an example of a context model.

The (concrete) context of a user may thus be described with a single value from each domain of each dimension at any time. However, it is not always feasible to acquire concrete information about context dimensions. For this reason, 4CRULES ac-

Table 1. A context model.

Dimension	DAY	HOURS	ACTIVITY	LOCATION	PRESENCE
Domain	Monday Tuesday Wednesday Thursday Friday Saturday Sunday	AM PM	journey lunch standBy visit anyOther	home office anyOther	appearOffline away beRightBack busy doNotDisturb offline online

cepts abstract contexts as input. Specifically, the context-records communicated by the acquisition system may include alternative values for each dimension to denote multiple states. For instance, the abstract context "Friday, PM, lunch or visit" denoted $\langle$D:{Friday},H:{PM},A:{lunch,visit}$\rangle$[3] is a valid record in reference to the model of Table 1. Notice that no assumption is made about the frequency of communications. It is for the acquisition system to decide which record to communicate and when.

As far as the telephony system is concerned, the meta-models for feature catalogue and feature subscription subsume that of DFC. A catalogue is a set of features and precedence constraints, and the induced relation may be cyclic (i.e., some features may be incompatible) and non-transitive. A feature subscription is a set of features, user-defined precedence constraints, and a set of catalogue precedence constraints defined on the selected features. It is consistent if the induced relation is acyclic. Catalogue and subscription meta-models also accommodate parameter signatures for features.

The feature configuration rule language is built upon the context and the feature catalogue meta-models. The antecedent of a FCR specifies an abstract context which has Cartesian product semantics similarly to that of context-records. The consequent of a FCR is a feature subscription augmented with feature exclusion constraints. Basically, the user can express inclusion and exclusion constraints over individual features and precedence constraints over the included features. The priority of a FCR is selected by the user from a total order (e.g., low $\prec$ medium $\prec$ high). The region of a FCR is target (resp., source) if the user is callee (resp., caller).

Figure 2 illustrates a set of FCRs. The identifier of the first FCR is 1, its priority is high, its region target, and it prescribes the activation of feature divert with parameter value addr1 if the activity recorded for the user is journey or visit. This means that all incoming calls received during journeys or visits should be forwarded to address addr1. The second rule prescribes the activation of feature tScreen before feature divert as indicated by keyword BEFORE. The third rule excludes feature divert as indicated by keyword DONT. In other words, this rule cannot be composed with a rule that requires divert. The next rule restricts the two dimensions Day and Hour. The last rule handles outgoing calls and is labelled with source instead of target.

Given an abstract context recorded for a user, and a set of FCRs provided by the user the rules-engine proceeds in two steps. It first identifies the set of FCRs that are applicable, i.e., whose antecedents intersect with the recorded abstract context. Notice that an applicable FCR does not necessarily subsume the concrete context of the user. For this

[3] We represent an abstract context by abbreviating dimensions with their initials (e.g., D for DAY) and omitting those that are not restricted (LOCATION and PRESENCE in this example).

```
[1,high,target] A:{journey,visit}              → divert(addr1)
[2,high,target] L:{home}                       → tScreen(list1) BEFORE divert(addr1)
[3,low,target]  A:{lunch}                      → play(away.mp3) DONT(divert)
[4,low,target]  D:{Monday,Friday} AND H:{PM}   → divertNoAnswer(addr2)
[5,low,target]  L:{anyOther}                   → play(welcome.mp3)
[6,high,source] L:{anyOther}                   → oScreen(list2)
```

Fig. 2. A set of feature configuration rules.

reason, the engine computes a probability of applicability for each FCR as described in Section 6. The second step computes an interaction-free sequence of features that is obtained by composing the consequents of applicable FCRs. Since applicable rules may not be consistent as described in Section 5, the engine determines a consistent relaxation, i.e., a subset of rules that are consistent. Since there may be many such relaxations, the engine computes an optimal relaxation using a lexicographic order based on rule priority, concreteness and probability of applicability. The engine solves this combinatorial optimisation problem using a constraint programming model.

The next sections present the notations and definitions related to the context and catalogue meta-models, the rules language, the notions of probability of applicability and relaxation, and the constraint programming formulation of the relaxation problem.

4 Context and Catalogue Meta-models

In this section we describe the context and catalogue meta-models.

4.1 Context Dimensions and Records

A *context model* is a tuple of context dimensions. A *context dimension* is represented by a finite domain of values called *context domain*. Let $\mathcal{M}$ denote a context model, m the number of context dimensions of $\mathcal{M}$, D_i, $1 \leq i \leq m$, the domain associated with the i^{th} dimension of $\mathcal{M}$, and d the size of the largest domain in $\mathcal{M}$. Without loss of generality we assume that each context domain is totally ordered.

An *abstract context* $a = \langle a_1, a_2, \ldots, a_m \rangle$ over $\mathcal{M}$ is a tuple consisting of a (non-empty) set of values per context domain. If all sets in a are singleton, a is said to be *concrete*. For instance, $\langle$`D:{Monday,Tuesday},A:{lunch,visit}`$\rangle$ is an abstract context while $\langle$`D:{Monday},H:{PM},A:{lunch},L:{office},P:{offline}`$\rangle$ is a concrete context wrt. Figure 1. Whenever a dimension is not specified (e.g., `L` in the previous abstract context) the full domain is assumed. A *context-record* is an abstract context over $\mathcal{M}$.

Let $a = \langle a_1, a_2, \ldots, a_m \rangle$ be an abstract context over $\mathcal{M}$. $[\![a]\!]$ denotes the Cartesian product $a_1 \times a_2 \times \cdots \times a_m$. Let $a_i = \langle a_{i_1}, \ldots, a_{i_m} \rangle$ and $a_j = \langle a_{j_1}, \ldots, a_{j_m} \rangle$ be two abstract contexts over $\mathcal{M}$. We say that a_i *subsumes* a_j, denoted by $a_j \subseteq a_i$ if and only if $(a_{j_1} \subseteq a_{i_1}) \wedge \cdots \wedge (a_{j_m} \subseteq a_{i_m})$. The *intersection* of a_i and a_j, denoted by $a_i \cap a_j$, is $\langle (a_{i_1} \cap a_{j_1}), \ldots, (a_{i_m} \cap a_{j_m}) \rangle$. The complexity of context intersection is linear in the number of dimensions and linear in the maximum domain size. The *maximal abstract context* over $\mathcal{M}$ for subsumption is denoted D, i.e., $D = \langle D_1, \ldots, D_m \rangle$.

4.2 Feature Catalogues and Subscriptions

A *feature catalogue* is a pair $\langle \mathcal{F}, \mathcal{H} \rangle$, where $\mathcal{F}$ is a set of features and $\mathcal{H}$ is a set of hard precedence constraints on $\mathcal{F}$. A precedence constraint, $i \prec j \in \mathcal{H}$, means that if the features i and j are part of a subscription then i must precede j in that subscription. Two features i and j are mutually exclusive if they can never appear together in any subscription. This is expressed by a pair of precedence constraints $i \prec j$ and $j \prec i$.

The name of a feature together with the name and type of its parameters define the *signature* of a feature. For instance, feature `play` in Figure 2 is parameterised with the name of a media file whereas `tScreen` is parameterised with a list of addresses. For each feature $f \in \mathcal{F}$, s_f denotes the signature of f. S denotes the set of feature signatures, i.e., $S = \{s_f | f \in \mathcal{F}\}$, and s the largest number of parameters for a feature.

Formally, a feature subscription for a catalogue $\langle \mathcal{F}, \mathcal{H} \rangle$ is a tuple $\langle F, V, H \cup P \rangle$, where $F \subseteq \mathcal{F}$, V is a set of feature parameter assignments complying with signature $s_f \in S$ for each $f \in F$, H is the projection of $\mathcal{H}$ on F, i.e., $\mathcal{H} \downarrow_F = \{(i \prec j) \in \mathcal{H} : i, j \in F\}$, and P is a set of user-defined precedence constraints on F. A feature subscription $\langle F, V, H \cup P \rangle$ is defined to be *consistent* if and only if the directed graph $\langle F, H \cup P \rangle$ is acyclic. For instance, a subscription $\langle \{$`tScreen, divert`$\}$, $\{$`tScreen:list=l1,divert:addr=a1`$\}$, $\{$`tScreen` $\prec$ `divert`$\}\rangle$ is acyclic and hence it is consistent.

Let $U_1 = \langle F_1, V_1, H_1 \cup P_1 \rangle$ and $U_2 = \langle F_2, V_2, H_2 \cup P_2 \rangle$ be two feature subscriptions. Let $v_{f_i} \in V_i$ denote the parameter assignment for $f \in F_i$ in U_i, $1 \leq i \leq 2$. We say that U_1 and U_2 are *unifiable* if and only if for all $f \in F_1 \cap F_2$, $v_{f_1} = v_{f_2}$. For instance, $\langle \{$`divert`$\}, \{$`divert:addr=a1`$\}, \emptyset \rangle$ and $\langle \{$`tScreen,divert`$\}, \{$`tScreen:list=l1, divert:addr=a1`$\}, \{$`tScreen`$\prec$`divert`$\}\rangle$ are unifiable subscriptions since feature `divert` has the same parameter assignment in each subscription. The *composition* of n subscriptions $\langle F_i, V_i, H_i \cup P_i \rangle$, $1 \leq i \leq n$, is the subscription $\langle F_c, V_c, H_c \cup P_c \rangle$, where $F_c = F_1 \cup \cdots \cup F_n$, $V_c = V_1 \cup \cdots \cup V_n$, $H_c = \mathcal{H} \downarrow_{F_c}$, and $P_c = P_1 \cup \cdots \cup P_n$. For instance, the composition of the two subscriptions $\langle \{$`divert`$\}, \{$`divert:addr=a1`$\}, \emptyset \rangle$ and $\langle \{$`divertNoAnswer`$\}, \{$`divertNoAnswer:addr=a2`$\}, \emptyset \rangle$ is the subscription $\langle \{$`divert,divertNoAnswer`$\}$, $\{$`divert:addr=a1,divertNoAnswer:addr=a2`$\}$,$\{$`divertNoAnswer` $\prec$ `divert`,`divert` $\prec$ `divertNoAnswer`$\}\rangle$ if we assume that `divert` and `divertNoAnswer` are mutually exclusive in the catalogue.

5 Feature Configuration Rules

A feature configuration rule (FCR) associates an abstract context to a consistent feature subscription. Let r be a FCR. $a_r = \langle a_{r_1}, \ldots, a_{r_m} \rangle$ denotes the abstract context of r where $\forall i, 1 \leq i \leq m$, $a_{r_i} \neq \emptyset$. $s_r = \langle F_r, V_r, H_r \cup P_r \rangle$ denotes the consistent feature subscription of r. We say that a FCR r is *applicable* to a given context-record ς if and only if a_r and ς have a common intersection, i.e., $[\![a_r \cap \varsigma]\!] \neq \emptyset$. For instance, FCR 1 in Figure 2 is applicable to $\langle$`A:{journey,standBy}`$\rangle$ but FCR 3 is not. Let R be a set of FCRs. We say that R is applicable to an abstract context ς if and only if ς intersects with each antecedent, i.e., $\forall r \in R$, $[\![a_r \cap \varsigma]\!] \neq \emptyset$. We say that the FCRs in R are *mutually applicable* to ς if and only if their antecedents and ς have

a common intersection, i.e., $[\![\bigcap_{r \in R} a_r \cap \varsigma]\!] \neq \emptyset$. For instance, the FCRs of Figure 2 are individually applicable but they are not mutually applicable to the abstract context $\langle$A:{journey,lunch},L:{home,anyOther}$\rangle$.

We want to allow the possibility of expressing that some feature cannot be part of the final subscription should the FCR be applied and composed with other FCRs. In order to allow that, for each feature $f \in \mathcal{F}$, a dummy feature $\overline{f}$ and two precedence constraints $f \prec \overline{f}$ and $f \succ \overline{f}$ are added to the catalogue to enforce mutual exclusion. The dummy feature $\overline{f}$ can then be included in the subscription to specify the exclusion of f should the rule be applied. This ensures that a FCR excluding f will be incompatible with a rule requesting f. We say that R is *compatible* with the catalogue if and only if the composition of the subscriptions prescribed by the rules of R denoted $\langle F_R, V_R, H_R \cup P_R \rangle$ is consistent. For instance, the rule-set $\{1, 2\}$ is compatible but $\{1, 3\}$ is not due to the mutual exclusion constraint between divert and $\overline{\texttt{divert}}$ in the catalogue. In the following we shall also assume that features divert and divertNoAnswer are mutually exclusive in the catalogue and that no other catalogue constraint applies to the features used in the rule-set in Figure 2. Therefore, the rule-set $\{1, 4\}$ is not compatible with the catalogue.

We say that R is *unifiable* if and only if the feature subscriptions prescribed by FCRs of R are pairwise unifiable. For instance, the rule-set in Figure 2 is not unifiable due to the assignment of feature play in rule 3 and 5. Figure 3 shows the pairs of rules in Figure 2 that are not compatible, not unifiable or not mutually applicable to the abstract context $\varsigma_1 = \langle$D:{Friday},H:{PM},A:{journey,lunch},L:{home, anyOther},P:{office}$\rangle$.

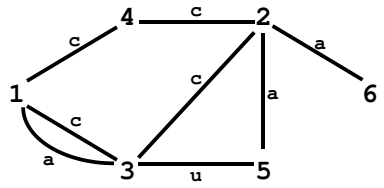

Fig. 3. Graph of inconsistent pairs of rules in Table 2 wrt. abstract context ς_1. Nodes represent rules and edges labelled with c (respectively, u, a) connect rules that are not compatible (resp., unifiable, mutually applicable to ς_1).

Given a set of FCRs R which is applicable to an abstract context ς, we say that R is *consistent* with ς if and only if the FCRs of R are mutually applicable to ς, R is unifiable, and R is compatible with the catalogue. For instance, the set of rules $\{3, 4, 6\}$ in Figure 2 is consistent with the abstract context ς_1 and corresponds to one of the independent sets in the inconsistency graph shown in Figure 3. Note that independence in the inconsistency graph is necessary but insufficient in the general case to ensure consistency since the rules of an inconsistent set of rules may be pairwise consistent. Each FCR r is associated with a weight that includes a user-defined priority, a concreteness measure, and a probability of applicability. The concreteness of a rule is fixed and rep-

resents the cardinality of its abstract context whereas its probability of applicability is relative to the context-record being considered.

Proposition 1. *Let R be a set of* FCR*s applicable to an abstract context ς. The time complexity of checking the consistency of R with ς is $\mathcal{O}(|R|(|F_R|\, s + m\, d) + |H_R \cup P_R|)$.*

Proof. The time complexity of checking the applicability of R is $\mathcal{O}(|R|\, m\, d)$, where m is the number of context dimensions and d is the maximum domain size of the context dimensions. This is because the time-complexity of computing the intersection of two abstract contexts is $\mathcal{O}(m\, d)$ and $|R|$ such intersections are required. The complexity of checking the compatibility of R is equivalent to checking the consistency of the feature subscription $\langle F_R, V_R, P_R \cup H_R \rangle$, which is $\mathcal{O}(|F_R| + |P_R \cup H_R|)$. The complexity of checking the unifiability of FCRs is $\mathcal{O}(|F_R|\, |R|\, s)$. This is because verifying whether a feature has the set of parameter values in different rules is $\mathcal{O}(|R|\, s)$ and checking this for all features is $\mathcal{O}(|F_R|\, |R|\, s)$. Thus, the overall time complexity is $\mathcal{O}(|R|(|F_R|\, s + m\, d) + |H_R \cup P_R|)$.

6 Probability of Applicability of a FCR

4CRULES assumes that any context-record, ς, sent by the context acquisition system for a user includes his/her current context, i.e., the concrete context denoting his state. Consistently with this assumption, the rules-engine discards rules that are not applicable, i.e, rules whose antecedents do not intersect with ς. Since their antecedents cannot include the current context, it is safe not to apply them. The method, however, considers any other rule as applicable. While this is safe for rules whose antecedents subsume the context-record, and therefore includes the current context, this is not necessarily safe for those rules whose antecedents do not subsume the context-record. For such rules, it cannot be exactly ascertained whether the antecedent includes the current context or not.

For this reason, the rules-engine computes a probability of applicability for each rule with respect to the context-record. This probability is based on a probability distribution over the space of concrete contexts that is specific to the user. Given a user u, the associated probability distribution, a context-record ς, and the antecedent a_r of a rule r associated with u, the probability that r is applicable to ς is the sum of the probabilities of the concrete contexts that are common to a_r and ς. By default, the distribution may be assumed uniform, i.e., each concrete context is equally likely to occur. In this case, the probability of applicability of a rule r to a context-record ς, denoted $Pu(a_r|\varsigma)$, is defined by $Pu(a_r|\varsigma) = |[\![a_r \wedge \varsigma]\!]|/|[\![\varsigma]\!]|$. For example, if $\varsigma = \langle \texttt{L:\{anyOther,office\},A:\{lunch\}} \rangle$, then $Pu(a_6|\varsigma) = 1/2$. Here a_6 denotes the abstract context $\langle \texttt{L:\{anyOther\}} \rangle$ associated with FCR 6 in Figure 2. Notice that $a_6 \wedge \varsigma$ and ς differ only in the dimension Location: the former has only one value for the dimension Location while the latter has two values.

Alternatively, a frequency distribution may be used if the context-records produced by the acquisition system are logged. The context-log of a user is a list of pairs $\langle a, f(a) \rangle$ where a is an abstract context and $f(a)$ its frequency. The log is initialised with the pair $\langle D, 1 \rangle$ corresponding to the maximal abstract context D. When the acquisition system

Algorithm 1 MCS($\mathcal{L}$,ς, R, k)

Require: $\mathcal{L}$: context-log of a given user, ς: context-record of the user, R: set of FCRs of the user applicable to context ς, and k: number of trials.

Ensure: Pf_r is an approximation of $Pf(a_r|\varsigma)$.

1: **for** $1 \leq r \leq |R|$ **do**
2: $s_r \leftarrow 0, t_r \leftarrow 0$
3: **for** $1 \leq q \leq k$ **do**
4: randomly select an abstract context a from $\mathcal{L}$ such that $a \wedge \varsigma \neq \emptyset$ favouring those contexts with higher frequency.
5: **for all** $r = 1 \ldots |R|$ **do**
6: $s_r \leftarrow s_r + |[\![a \wedge \varsigma \wedge a_r]\!]|$
7: $t_r \leftarrow t_r + |[\![a \wedge \varsigma]\!]|$
8: **for** $r = 1 \ldots |R|$ **do**
9: $Pf_r \leftarrow s_r/t_r$

produces an abstract context a for the user, his/her context-log is extended with a new pair $\langle a, 1\rangle$ if there is no pair whose first element is a or else the frequency of a is incremented by 1 in the existing pair $\langle a, f(a)\rangle$. Given a log of context-records $\mathcal{L}$, the probability of applicability of a rule r to a context-record ς, denoted $Pf(a_r|\varsigma)$, is

$$Pf(a_r|\varsigma) = \frac{\sum_{\langle a,f(a)\rangle \in \mathcal{L}} |[\![a \wedge \varsigma \wedge a_r]\!]| \times f(a)}{\sum_{\langle a,f(a)\rangle \in \mathcal{L}} |[\![a \wedge \varsigma]\!]| \times f(a)} \quad (1)$$

The denominator is the total count of the concrete contexts covered by the abstract contexts subsumed by ς according to $\mathcal{L}$, and the numerator is the number of times a user has been in one of the concrete contexts subsumed both by ς and a_r. Let $size(\mathcal{L}) = \sum_{a\in\mathcal{L}} f(a)$ be the *size* of a log $\mathcal{L}$. The exact computation of $Pf(a_r|\varsigma)$ is linear with respect to $size(\mathcal{L})$. If $size(\mathcal{L})$ is large, a Monte-Carlo method such as Algorithm 1 can be used for generating a close approximation. Given a context-log $\mathcal{L}$ for a user u, a context-record ς, a set of rules R applicable to ς, and an integer k, Algorithm 1 randomly selects k abstract contexts from the log favouring those contexts with higher frequency and it updates two counters s_r and t_r for each rule r. Here t_r denotes the total count of the concrete contexts covered by k abstract contexts selected from the log that are subsumed by ς, and s_r denotes the portion of t_r that is consistent with a_r. The algorithm returns the ratio of s_r and t_r which approximates $Pf(a_r|\varsigma)$. When $size(\mathcal{L})$ is small, it is reasonable to use $Pu(a_r|\varsigma)$ for computing the probability of applicability. As $size(\mathcal{L})$ increases, it is desirable to gradually switch to $pf(a_r|\varsigma)$. In order to achieve that, a weighted average of $Pu(a_r|\varsigma)$ and $Pf(a_r|\varsigma)$, denoted $Pa(a_r|\varsigma)$, may be used.

7 Optimal Relaxation of a Set of FCRs

Given a context-record ς and a set of FCRs $\mathcal{R}$, the engine searches for a consistent set of rules amongst $\mathcal{R}$. Let $R \subseteq \mathcal{R}$ be the set of rules applicable to ς. If the rules in R are mutually not applicable to ς, or R is non-unifiable, or it is incompatible with the catalogue then R is inconsistent, in which case the task is to find a relaxation of $\mathcal{R}$ that

Algorithm 2 $\lambda(r_i, r_j)$: `Boolean`

1: **If** $Pa(a_{r_i}/\varsigma) > Pa(a_{r_j}/\varsigma)$ **then return** `true`
2: **else if** $Pa(a_{r_i}/\varsigma) < Pa(a_{r_j}/\varsigma)$ **then return** `false`
3: **else if** $\kappa(r_i) > \kappa(r_j)$ **then return** `true`
4: **else if** $\kappa(r_i) < \kappa(r_j)$ **then return** `false`
5: **else if** $\gamma(r_i) < \gamma(r_j)$ **then return** `true`
6: **else return** `false`

is consistent. From now on, by relaxation we mean consistent relaxation. A consistent relaxation R' of R is *maximal* if there does not exist any relaxation R'' verifying $R' \subset R''$. Since there may be many maximal relaxations, the engine searches for a maximal relaxation that is optimal in some sense. The notion of optimality is defined using a lexicographic order over sets of FCRs. A lexicographic order over maximal relaxations can be derived from a strict weak ordering over rules.

A strict weak ordering is a binary relation on a set that is a strict partial order in which the relation "neither a is related to b nor b is related to a" is transitive. Given a rule $r_i \in R$, $Pa(a_i|\varsigma)$ denotes the probability that the rule is applicable to ς, $\kappa(r_i)$ denotes the priority of the rule, and $\gamma(r_i)$ denotes the concreteness of the rule. Algorithm 2 introduces a strict weak ordering λ which compares rules based on their probability of applicability, priority, and concreteness. Given two rules, λ first compares their probabilities of applicability. If the probabilities are equal then λ compares their priorities. If the priorities are equal then λ compares their concreteness. If their concreteness are also identical then the two rules are λ-incomparable, i.e., equally important with respect to λ. Another variant of the described ordering could be obtained by defining equivalence classes of the probabitlies and using them to compare the rules instead of their precise probabilities. An alternative strict weak ordering can be obtained by combining priority and probability of applicability, i.e., r_i is preferred to r_j if and only if $Pa(a_{r_i}/\varsigma) \times \kappa(r_i) > Pa(a_{r_j}/\varsigma) \times \kappa(r_j)$.

Given a strict weak ordering λ, $\Theta_\lambda(R)$ denotes the total ordering of R obtained by ordering its elements using λ and breaking the ties between λ-incomparable rules using their identifiers. Let R' and R'' be two consistent sets of rules such that $\Theta_\lambda(R') = \langle r'_1, \ldots, r'_p \rangle$ and $\Theta_\lambda(R'') = \langle r''_1, \ldots, r''_q \rangle$. R' is *lexicographically better* than R'' for λ, denoted $\Theta_\lambda(R') \prec_{lex} \Theta_\lambda(R'')$ if and only if

- $p > q$ and for all i, $i \leq q$, neither $\lambda(r'_i, r''_i)$ nor $\lambda(r''_i, r'_i)$ holds, or
- there exists $i \leq \min(p, q)$ such that $\lambda(r'_i, r''_i)$ holds and for all $l < i$, neither $\lambda(r'_l, r''_l)$ nor $\lambda(r''_l, r'_l)$ holds.

In words, R' is better than R'' if there exists $r'_i \in \Theta_\lambda(R')$ and $r''_i \in \Theta_\lambda(R'')$ such that r'_i is more important than r''_i with respect to λ and for all r'_l and r''_l occurring before r'_i and r''_i, r'_l and r''_l are equally important. This assumes that R' and R'' have the same number of rules. If not, the smaller one is padded with "blank rules", where a blank rule is treated as a least important rule with respect to λ.

A relaxation R' of R is *optimal* if and only if there does not exist any relaxation R'' of R such that $\Theta_\lambda(R'') \prec_{lex} \Theta_\lambda(R')$. By definition of $\prec_{lex}$, an optimal relaxation is necessarily maximal with respect to set inclusion. If λ generates a total order on R

then $\prec_{lex}$ is itself a total order and finding an optimal relaxation of R is polynomial. However, if λ is a strict weak order on R, then $\prec_{lex}$ is also a strict weak ordering and finding an optimal relaxation of R is NP-hard. This can be proven by reducing the maximum independent set problem to the problem of finding a lexicographically optimal relaxation of a set of rules. Given a graph, a node can be associated with a rule and an edge between two nodes can be associated with an incompatibility between two rules. As finding a maximum independent set is NP-hard, finding a lexicographically optimal relaxation is also NP-hard. Notice that two rules can be incompatible: (1) when they are mutually not applicable, i.e., when their abstract contexts do not intersect, (2) when they are non-unifiable, i.e., when a common feature is assigned different parameter values, or (3) when the composition of their configurations is inconsistent.

Let us consider an example to demonstrate the concepts of maximal relaxation and optimal relaxation. Let $\varsigma_1 = \langle$`D:{Friday},H:{PM},A:{journey,lunch},L:{home, anyOther},P:{office}`$\rangle$ be an abstract context. All the rules in Figure 2 are applicable, since the intersection of ς_1 with each of them is non-empty. The probability of applicability of FCR 4 is 100%, while for others is 50%. The concreteness of FCR 4 is 210, that of FCR 3 is 294, that of FCRs 2, 5 and 6 is 490 and that of FCR 1 is 588. The strict weak ordering on the rule-set with respect to λ is $4 \prec \{2, 6\} \prec 1 \prec 3 \prec 5$. This rule-set is not consistent with ς_1 as shown in Figure 3. It has 4 maximal relaxations which are $A = \{2, 1\}$, $B = \{1, 6, 5\}$, $C = \{4, 6, 3\}$ and $D = \{4, 6, 5\}$. The lexicographic ordering over maximal relaxations with respect to $\prec_{lex}$ is a strict total order equal to $C \prec D \prec B \prec A$. C is therefore the optimal relaxation.

8 A Constraint Optimisation Formulation

This section formulates the problem of finding a lexicographically optimal relaxation of an inconsistent set of FCRs as a constraint optimisation problem (COP). The COP is defined in terms of finite domain variables, constraints restricting the assignments of values to the variables, and an objective function. The variables model the inclusion of rules in the computed relaxation, the inclusion and positioning of features in the computed subscription, and the satisfaction of user precedences. The constraints model all the requirements —mutual applicability, unifiability and compatibility of the rule-set— whereas the objective function models the lexicographic order over rule-sets.

Variables and Domains. A Boolean variable br_i is associated with each rule $r_i \in R$, which is instantiated to 1 or 0 depending on whether r_i is included in the computed relaxation or not, respectively. Each $f_i \in F_R$ is associated with two variables: a *Boolean variable* bf_i and an *integer variable* pf_i. A variable bf_i is instantiated to 1 or 0 depending on whether f_i is included in the computed subscription or not, respectively. The domain of each variable pf_i is $\{1, \ldots, |F_R|\}$. If f_i is included then pf_i denotes the position of f_i in a sequence. Each user precedence constraint $p_{ij} \equiv (f_i \prec f_j) \in P_R$ is associated with a *Boolean variable* bp_{ij}, which is instantiated to 1 or 0 depending on whether p_{ij} is respected in the computed subscription or not, respectively.

Constraints. A catalogue precedence constraint $(i \prec j) \in H_R$ can be expressed as $bf_i \wedge bf_j \;\Rightarrow\; (pf_i < pf_j)$, which is trivially satisfied when either bf_i or bf_j is

instantiated to 0. A user precedence constraint $(i \prec j) \in P_R$ can be expressed as $bp_{ij} \Rightarrow (bf_i \wedge bf_j \wedge (pf_i < pf_j))$. If it holds then f_i and f_j are included in the subscription and f_i is placed before f_j in the sequence. For each $f \in F_R$, if $f \in (F_{r_i} \cap F_{r_j})$ and $v_{f_i} \neq v_{f_j}$, then the rules r_i and r_j are non-unifiable. In order to ensure unifiability the constraint $\neg br_i \vee \neg br_j$ is added. The computed relaxation is a mutually applicable set of rules which is expressed by $[\![\bigwedge_{r_i \in R \wedge br_i = 1} a_i \wedge \varsigma]\!] \neq \emptyset$. If r_i is included in the computed relaxation then its subscription is included in the computed subscription (i.e., the subscription induced by the computed relaxation). This is expressed by the constraint $br_i \Rightarrow \bigwedge_{f_j \in F_i} bf_j \wedge \bigwedge_{p_{kl} \in P_i} bp_{kl}$.

A feature f_j (respectively, a user precedence p_{kl}) can only be included in the computed subscription if it belongs to the subscription of a rule that is included in the computed subscription. For each feature $f_j \in F_R$ and each user precedence $p_{kl} \in P_R$, we add the constraints $bf_j \Rightarrow \bigvee_{(f_j \in F_i) \wedge (r_i \in R)} br_i$ and $bp_{kl} \Rightarrow \bigvee_{(p_{kl} \in P_i) \wedge (r_i \in R)} br_i$.

Objective Function. A solution of the COP model is a subscription induced from the set of the rules that are included. Let I be a subset of rules of R that are included, i.e., $\{r_i | r_i \in R \wedge br_i = 1\}$. The value of the solution is $\Theta_\lambda(I)$, which is an ordered set of rules based on the comparator λ. The objective is to find an ordered set of rules, $\Theta_\lambda(I)$, that is consistent and lexicographically optimal.

9 Empirical Evaluation

The purpose of our experiments was to get an insight into the behaviour of 4CRULES and assess the feasibility of the optimisation approach for rules processing. Experiments were carried out using an implementation of 4CRULES based on Choco, version 2.1, a Java library for constraint programming systems (`http://choco.sourceforge.net/`).

We devised our experimental model based on practical knowledge of feature catalogues and context models to assess the impact of input context-records on response time and rules applicability. To do so, we created a context model, a catalogue, a rule-set and context records. The context model has 4 context dimensions of maximum domain size 12, which is realistic for consumer applications. The catalogue includes 25 features and 125 precedence constraints and was randomly generated as specified in [18]. The generated catalogue is similar in size to those found in academic literature [17] or used commercially [13]. The rule-set includes 50 FCR

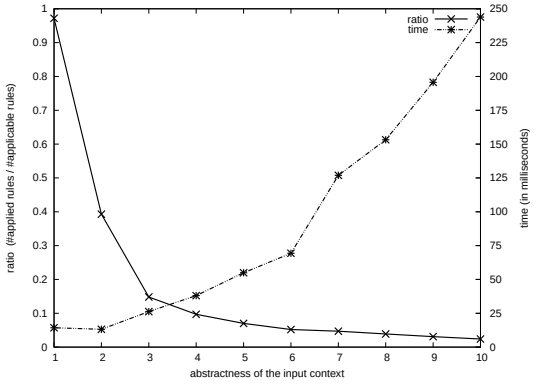

Fig. 4. Behaviour of 4CRULES wrt. input context-records.

which probably exceeds the number of situations a user might really want to "control". The rules were generated as follows: each rule antecedent was generated by randomly selecting 2 concrete values per dimension, and each consequent was generated by randomly selecting 2 features. The same priority was used for all rules. For i ranging from 1 to 10, we generated a context-record containing i concrete values per dimension.

We ran the rules engine with each context-record using the rule-set and the catalogue. The results are shown in Figure 4. The x-axis represents the abstractness of the context-record, that is, the parameter i. The y-axis depicts as the ratio of the number of applied rules to the number of applicable rules (left axis) as well as the response time (in milliseconds) for finding an optimal set of rules (right axis). Each point in the plot is an average of 25 instances. The graph shows that as the abstractness of the context-record increases the response time increases while the ratio of applicability decreases. These results confirm the practicality of the proposed model in terms of response time.

10 Conclusion

We have introduced 4CRULES, a rule-based system that enables context-sensitive call control using feature configuration rules. 4CRULES has been designed to be interoperable with standard context services and compositional feature architectures. 4CRULES handles conflicting preferences and mitigates the uncertainty affecting context data. A constraint optimization approach is used to compute configurations that meet the user requirements to an optimum degree.

The approach adopted was to compute optimal consistent rule-sets by determining the mutual applicability of rules to the input context, the compatibility of their configurations with feature interaction constraints, and their aggregate value using a lexicographic ordering combining rule priority, concreteness, and probability of applicability. Experiments on random test instances confirmed the practicality of the approach and highlighted performance critical factors.

Future work will involve the investigation of the rules edition functionalities (e.g., validation, refactoring, compilation), richer ontological models for context domains and catalogues (e.g., feature dependencies), and new application domains (e.g., smart RSS feeds, software plug-in configurations).

Acknowledgments

This material is based upon work supported by the Science Foundation Ireland under Grant numbers 05/IN/I886, 08/PI/I1912, and Embark Post Doctoral Fellowships numbers CT1080049908 and CT1080049909.

References

1. Calder, M., Kolberg, M., Magill, E.H., Reiff-Marganiec, S.: Feature Interaction: A Critical Review and Considered Forecast. Computer Networks **41**(1) (January 2003) 115–141

2. Bond, G.W., Cheung, E., Purdy, H., Zave, P., Ramming, C.: An Open Architecture for Next-Generation Telecommunication Services. ACM Transactions on Internet Technology **4**(1) (2004) 83–123
3. Lesaint, D., Papamargaritis, G.: Personalised Communications. In Voudouris, C., Owusu, G., Dorne, R., Lesaint, D., eds.: Service Chain Management - Technology Innovation for the Service Business. Springer (2008) 187–203
4. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A.B., Peterson, J., Sparks, R., Handley, M., Schooler, E.M.: SIP: Session Initiation Protocol. RFC 3261, IETF (June 2002)
5. Sparks, R.: SIP: Basics and Beyond. ACM Queue **5**(2) (March 2007) 22–33
6. Lennox, J., Wu, X., Schulzrinne, H.: Call Processing Language (CPL): A Language for User Control of Internet Telephony Services. RFC 3880, IETF (October 2004)
7. Wu, X., Schulzrinne, H.: Handling Feature Interactions in the Language for End System Services. In: Feature Interactions in Telecommunications and Software Systems VIII (ICFI'05), Leicester, UK, IOS Press (June 2005) 28–30
8. Turner, K.J., Reiff-Marganiec, S., Blair, L., Pang, J., Gray, T., Perry, P., Ireland, J.: Policy Support for Call Control. Computer Standards & Interfaces **28**(6) (2006) 635–649
9. Reiff-Marganiec, S., Turner, K.J., Blair, L.: APPEL: The ACCENT Project Policy Environment/Language. Technical report, University of Stirling, Scotland (December 2005)
10. Blair, L., Turner, K.J.: Handling Policy Conflicts in Call Control. In Reiff-Marganiec, S., Ryan, M., eds.: Feature Interactions in Telecommunications and Software Systems VIII, ICFI'05, Leicester, UK, IOS Press (June 2005) 39–57
11. Jackson, M., Zave, P.: Distributed Feature Composition: a Virtual Architecture for Telecommunications Services. IEEE Transactions on Software Engineering **24**(10) (October 1998) 831–847
12. Jackson, M., Zave, P.: The DFC Manual. AT&T. (November 2003)
13. Bond, G.W., Cheung, E., Goguen, H., Hanson, K.J., Henderson, D., Karam, G.M., Purdy, K.H., Smith, T.M., Zave, P.: Experience with Component-Based Development of a Telecommunication Service. In: Proc. of the 8th Int. Symposium on Component-Based Software Engineering (CBSE 2005). Volume 3489 of Lecture Notes in Computer Science., St. Louis, MO, Springer (2005) 298–305
14. Zave, P.: An Experiment in Feature Engineering. In McIver, A., Morgan, C., eds.: Programming Methodology. Springer-Verlag (2003) 353–377
15. Zave, P., Cheung, E.: Compositional Control of IP Media. In Diot, C., Ammar, M., da Costa, C.S., Lopez, R., Leitao, A.R., Feamster, N., Teixtera, R., eds.: Proc. of the 2nd Conf. on Future Networking Technologies (CoNext 06), Lisboa, Portugal, SIGCOMM (December 2006) 67–78
16. Zave, P.: Audio Feature Interactions in Voice-over-IP. In Bond, G.W., Schulzrinne, H., Sisalem, D., eds.: Proc. of the 1st Int. Conf. on Principles, Systems and Applications of IP Telecommunications (IPTComm), New York, NY, IPTComm (July 2007) 67–78
17. Zimmer, A.P.: Prioritizing Features Through Categorization: An Approach to Resolving Feature Interactions. PhD thesis, University of Waterloo, Canada (September 2007)
18. Lesaint, D., Mehta, D., O'Sullivan, B., Quesada, L., Wilson, N.: Personalisation of Telecommunications Services as Combinatorial Optimisation. In: IAAI-08, AAAI Press (2008) 1693–1698