Feature Interaction as a Satisfiability Problem

Carlos Areces
ILLC, Univ. of Amsterdam
Pl. Muidergracht 24
1018 TV Amsterdam
The Netherlands
carlos@wins.uva.nl

Wiet Bouma
KPN Research
P.O. Box 421
2260 AK Leidschendam
The Netherlands
L.G.Bouma@research.kpn.com

Maarten de Rijke*
ILLC, Univ. of Amsterdam
Pl. Muidergracht 24
1018 TV Amsterdam
The Netherlands
mdr@wins.uva.nl

Abstract

We present a formal model for the specification of telephone features by means of description logics. Our framework permits the formal definition of the basic telephone system as well as the specification of additional features (Call Waiting, Call Forwarding, etc.). Furthermore, by using standard reasoning tasks from Description Logics, properties of features can be formally proved and interactions detected. An EXPTIME upper bound for the complexity of detecting feature interaction as a satisfiability problem is provided by exploiting well-known results for expressive description languages.

1. Introduction

The term “feature interaction” was originally introduced in the telecommunications domain when new features started being added to the basic telephony service. The phrase seems to have been introduced in 1983, with the publication of a comprehensive set of feature descriptions by Bellcore. In [4], one of the first attempts was made to define the general problem, and to develop a software engineering framework for managing feature interactions. A *feature* in the context of telephone systems is an addition of functionality to provide new behavior to the users or the administration of the telephone system. Classical examples of features are: Call Waiting, Call Forwarding in several flavors, Three-Way Calling, Automatic Call Back (also called Call Completion Busy Subscriber), and reverse- and split-charging. There are many other feature domains in the telecommunications area, for example data- and multi-media communication (like video telephony, video-conferencing, tele-education, video on demand), management of equipment, and billing and administration. However, traditionally, features and their interactions in the end-user telephony domain are the most visible. The list of “classical” features is very long, for instance [1] contains hundreds of feature descriptions.

Feature *interaction* arises when the behavior of one feature influences the behavior of another, mostly in an unexpected and unwanted way. A simple example is the combination of Call Waiting and Call Forwarding on Busy: if the network gives priority to one, the other is effectively disabled. The problem has recently generated a lot of attention in the telecommunications domain. Reasons for this are the ever increasing complexity of the set of features deployed in current telephone networks and the need to rapidly develop and commercially deploy new features on top of already installed ones, feature creation by third parties, and the increasingly distributed nature of software for telecommunications systems: service programs are possibly executed by different network components, or even by components in other domains.

Because the feature interaction problem is so general, there have been many attempts to come up with a classification of interactions. We give a brief overview following [12]. Probably the most accepted is the *causal* view from [6] which classifies interactions according to their reason of occurrence: limitations of network support (like a poor user-to-network signaling interface), intrinsic problems in distributed systems (like timing problems and race conditions), or violation of assumptions (about the environment in which a feature operates). One can also take a *software-engineering* point of view, and subdivide feature interactions according to the phase of the software life-cycle where they occur (requirements specification, software design, implementation, testing, provisioning), see e.g., [7]. Yet another possibility is to classify interactions according to the *number of users and network components* involved. Finally, one can take an *organizational* point of view and focus on who in an organization is responsible for dealing with the occurrence of interactions, see for instance [13].

*Supported by the Spinoza project ‘Logic in Action.’

There are many approaches to finding and dealing with interactions. The accepted categorization from [3] is to use a two-dimensional scheme and split approaches into on-line and off-line on the one hand, and avoidance, detection and resolution on the other. On-line vs. off-line refers to *when* interactions are dealt with: during or before deployment. The split in avoidance, detection and resolution refers to whether features are designed so that interactions simply do not occur (or rather are always completely explicit), feature descriptions are analyzed against each other to find interactions, or different methods are devised to deal with interactions when they occur, respectively.

Traditionally, detection methods have been most popular, and especially analytical methods based on formal verification techniques. The idea is to do a formal design of both system and features, and to use these formal descriptions to detect interaction. In the great majority of these cases, a functional model of the telecommunications network is constructed, on top of which features are added. Correctness criteria for proper functioning of the system and features are then checked using verification techniques, most often model checking. We refer to [12] for a detailed overview. In contrast to the model-oriented approach, only few contributions use a satisfiability approach in which the system is described using a specific logic, mostly predicate, temporal or modal logic. Feature requirements are then checked for consistency against the system description. One example here is the work by Blom [2], which uses specification techniques close to linear temporal logic. Specifications can automatically be mapped onto finite state machines. Interaction is defined as states with transitions that have conflicting enabling conditions, or that have conflicting effects. Gammelgaard and Kristensen [10] model a telephony service, possibly enhanced with features, as a set of *propositional axioms* together with a set of *transition rules*; feature interaction is defined as non-satisfiability of such axiom sets. We will come back to this approach in more detail in Section 5.

The feature detection done in the proposals mentioned above is mostly informal. There is no calculus for the language, and the arguments analyzing feature interaction are a kind of semantic evaluation of models. Given the intricacies of the subject area, this is bound to lead to errors. Moreover, unless we have a proper formalization, we have no way of understanding the complexity of the feature interaction problem, neither a way of using formal tools to help us detect interaction. To address these shortcomings, we provide a completely formal approach using standard reasoning tasks; furthermore, we argue that the language we propose is a "good logic" to reason about features and the interaction problem, with sufficient expressive power and with sound and complete decision methods.

In Section 2 we introduce description logics and we present the language $\mathcal{FI}$. This specification language is put to use in Section 3 to model the basic call service and additional features. A definition of interaction and an algorithmic method for detecting them, illustrated with an example, forms the content of Section 4. In Section 5 we discuss the relation of our work with that of Gammelgaard and Kristensen mentioned above. The paper concludes with a discussion on the complexity of the approach and directions for future research.

2. Description Logics for Feature Interaction

Description logics deal with the representation of structured concepts, and of objects that may or may not be related to other objects, and that may or may not satisfy some of these concepts. To preserve structural properties of the data being represented, description logics descending from the KL-ONE system [5] split information in two kinds: ABoxes which contain assertional information (facts concerning individuals in the domain), and TBoxes with terminological knowledge (definitions of derived notions).

Example 2.1 Consider the following example:

$$\texttt{TBox:}\ T \;=\; \left\{ \begin{array}{l} \mathit{man} \sqsubseteq \mathit{human} \sqcap \mathit{male} \\ \mathit{father} \doteq \mathit{man} \sqcap \exists \mathit{has{-}child}.\top \end{array} \right\}$$

$$\texttt{ABox:}\ A \;=\; \left\{ \begin{array}{l} m : \mathit{human} \\ m : \mathit{male} \\ (m, e) : \mathit{has{-}child} \end{array} \right\}$$

The formulas in T define concepts (subsets of the domain to which information being represented refers). For example, the first formula defines the concept of being a 'man' as a subset of those humans which are male. It is not a strict definition because further conditions like having to be 'alive' and older than a certain age are not specified. In other words, been *human* and *male* are necessary conditions but they are not sufficient to qualify as *man*. The second formula, on the other hand, strictly defines the concept of being a 'father' as the set of all those men who have a child (*has-child* is a binary relation and $\top$ represents a trivial condition, i.e., a tautology). The formulas in the ABox A assert that a certain m is both human and male, and also that there exists an element e which stands in the 'has-child' relation with m. The available information is not enough to characterize m as a father since we don't know if m satisfies the required further conditions which were left open in the definition of man.

Summing up, concepts and roles are the building blocks of description logics, and each description logic is characterized by the way in which these can be combined in TBoxes and ABoxes. More formally, the *signature* of a description logic is given by specifying three disjoint sets C, R and A of concept and role names, and atomic constants respectively.

Below, we propose a specific description logic as a suitable formalism for describing feature interaction. Consider the following simple example. Suppose we have A and B which are subscribers to the basic telephony system. A communication path established between A and B forces both telephones to be engaged, but *engaged*$_A$ and *engaged*$_B$ are not sufficient conditions for the existence of a communication path between A and B as they can be talking with other subscribers. In symbols, we have $path_{AB} \sqsubseteq engaged_A \sqcap engaged_B$. We will use terminologies and assertions to provide a formal model of the basic telephony system; on top of that, we explicitly define how this basic system is modified by adding features.

The Description Logic $\mathcal{FI}$. Fix a signature $\tau = \langle \mathsf{C}, \mathsf{R}, \mathsf{A} \rangle$ of concept names, role names and constants. The sets Co of concepts, Te of terminologies (formulas allowed in TBoxes) and As of assertions (formulas allowed in ABoxes) are defined as follows:

$$\begin{array}{rcl}
\mathsf{BCo} & := & \top \mid C \mid \neg\mathsf{BCo} \mid \mathsf{BCo} \sqcap \mathsf{BCo} \\
\mathsf{Co} & := & \mathsf{BCo} \mid \forall R.\mathsf{BCo} \mid \exists R.\mathsf{BCo} \\
\mathsf{Te} & := & \mathsf{BCo} \sqsubseteq \mathsf{Co} \mid \mathsf{BCo} \doteq \mathsf{Co} \\
\mathsf{As} & := & a : \mathsf{Co} \mid (a,b) : R
\end{array}$$

Here C is a basic concept in C, R a basic relation in R and a, b constants in A. As in Example 2.1, sets of elements of Te and As are used to specify notions and to assert that certain elements correspond to them. Usually, a pair $\langle T, A \rangle$ where $T \subseteq \mathsf{Te}$ and $A \subseteq \mathsf{As}$ is called a *knowledge base*.

Description logics are interpreted on *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is any non-empty set, and $\cdot^{\mathcal{I}}$ is a function assigning subsets of $\Delta^{\mathcal{I}}$ to concept names, binary relations over $\Delta^{\mathcal{I}}$ to role names, and single elements of $\Delta^{\mathcal{I}}$ to atomic constants. $\mathcal{I}$ can be extended to all formulas in Co as follows $(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall d' \, ((d,d') \in R^{\mathcal{I}} \Rightarrow d' \in C^{\mathcal{I}})\}$, and $(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d' \, ((d,d') \in R^{\mathcal{I}} \ \& \ d' \in C^{\mathcal{I}})\}$. The last two conditions deserve further comments. The intended meaning of $\exists R.C$ is the set of all elements which stand in the relation R with some C-element (see the definition of the concept *father* in Example 2.1). The operator $\forall$ is the dual notion and can be defined in terms of $\exists$: $\forall R.C = \neg\exists R.\neg C$.

We can now define the key notions of *satisfaction* and *consequence*. A satisfaction relation $\models$ is a relation between interpretations and terminologies or assertions. Intuitively, $\models$ relates an interpretation $\mathcal{I}$ with formulas whose intended meaning is supported by $\mathcal{I}$: $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $\mathcal{I} \models C \doteq D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$, $\mathcal{I} \models a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\mathcal{I} \models (a,b) : R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Given a subset K of $\mathsf{Te} \cup \mathsf{As}$, we say that $\mathcal{I} \models K$ if each element of K is supported by $\mathcal{I}$ (i.e., $\mathcal{I} \models \varphi$ for all formulas φ in K). Recall that we will use knowledge bases as in Example 2.1 to provide a formal description of the basic call service. Let BCS be such a knowledge base; we are especially interested in properties that are supported in any interpretation supporting BCS. More generally, given a knowledge base $\langle T, A \rangle$ and $\varphi \in \mathsf{Te} \cup \mathsf{As}$, we say that φ *follows from* $\langle T, A \rangle$ (notation $\langle T, A \rangle \models \varphi$) if for all interpretations $\mathcal{I}$, $\mathcal{I} \models T \cup A$ implies $\mathcal{I} \models \varphi$. Checking whether a formula φ follows from a knowledge base is the main reasoning task for description logics; in Section 3.2 we define detection of feature interaction as a special instance of this task.

3. Modeling

We first model the basic call service as a knowledge base in $\mathcal{FI}$; after that we discuss the addition of features.

3.1 Step 1: Description of *BCS* in $\mathcal{FI}$

We should first fix the set of atomic concepts, relations and constants we will use. We will provide a list, together with a brief description of their intended meanings. Assume there is a fixed (finite) set SUB of indices which represent subscribers to the telephone system. In most cases, feature interaction can already be detected in the presence of a small number of subscribers. We will annotate atomic concepts and roles with indexes from SUB.

Atomic Concepts. We need concepts expressing that a user $u \in \mathsf{SUB}$ is in a specific state. The following concepts form the set ST_u of possible states of a user u.

idle$_u$	the telephone has the receiver on hook and is silent;
dialtone$_u$	the receiver is off hook and emits a dial tone; a number can be dialed;
busytone$_u$	the receiver is off hook and emits a busy tone, indicating a failed call attempt or the party has hung up;
ringing$_u$	the telephone is ringing with the receiver on hook;
ringback$_u$	the receiver is off hook and emits a ring back tone (called party's phone is ringing);
engaged$_u$	there is a connection (also called speech path) with another party.

We also need concepts expressing a (minimal) presence of the network, because a call proceeds through phases (internal states of the network) that are not directly observable by a user. Moreover, we need to be able to express connections between users being active. These, for $u, v \in \mathsf{SUB}$, form the set IST_{uv} of internal states.

calling$_{uv}$	the phone at v is ringing with u waiting for v to accept the call;
path$_{uv}$	u and v can communicate (have a speech path).

Define C to be $\bigcup_{u,v\in\mathsf{SUB}}(\mathsf{ST}_u \cup \mathsf{IST}_{uv})$.

Atomic Roles. Roles represent possible actions of subscribers. As before, define R as the union of all ROLES_u for $u \in \mathsf{SUB}$, where ROLES_u is the following:

- ***offhook***$_u$ representing the action of u lifting the receiver;
- ***dial***$_{uv}$ representing the action of u dialing v's number (for each $v \in \mathsf{SUB}$);
- ***onhook***$_u$ representing the action of u putting down the receiver.

It is standard to assume that the system is deterministic, i.e., any action of a subscriber changes the state of the system to a unique and completely specified new state. This assumption is so common in knowledge representation, that the logic comes "prepared" for handling certain roles as partial functions (i.e., if $(a,b) : R$ and $(a,c) : R$ then $b = c$); see Section 4.2. All roles in R are assumed to be partial functions.

TBox and ABox. Now that concept and role names have been defined, we can specify *BCS*. Again, we use the convention that the statements presented below are schemes where u and v range over the set SUB of subscribers. To begin, there are (TBox) expressions connecting the observable states of a telephone with the ones representing network states:

- $calling_{uv} \sqsubseteq ringing_v \sqcap ringback_u,\ u \neq v$;
- $ringing_u \sqsubseteq \bigsqcup_{v\in\mathsf{SUB},v\neq u} calling_{vu}$;
- $ringback_u \sqsubseteq \bigsqcup_{v\in\mathsf{SUB},v\neq u} calling_{uv}$;
- $path_{uv} \sqsubseteq engaged_u \sqcap engaged_v$;
- $engaged_u \sqsubseteq \bigsqcup_{v\in\mathsf{SUB},v\neq u} path_{uv}$;
- $path_{uv} \doteq path_{vu},\ u \neq v$. (In the absence of billing, a speech path is symmetric.)

Next, there are statements specifying how a user and the network can change state.

- $idle_u \sqsubseteq \exists offhook_u.dialtone_u$ (if u is idle, she can go offhook and accept digits);
- $dialtone_u \sqcap idle_v \sqsubseteq \exists dial_{uv}.calling_{uv}$ (if u has a dialtone and v idle, u can dial v's number and establish a call);
- $dialtone_u \sqsubseteq \exists onhook_u.idle_u$ (if u has a dialtone, she can decide to hang up and go idle);
- $dialtone_u \sqcap \neg idle_v \sqsubseteq \exists dial_{uv}.busytone_u$ (if u's party is busy, u's phone emits a busytone);
- $busytone_u \sqsubseteq \exists onhook_u.idle_u$ (if u has a busytone, she can go onhook to become idle);
- $calling_{uv} \sqsubseteq \exists offhook_v.path_{uv}$ (when u is calling v and v goes offhook, this establishes a speech path);
- $calling_{uv} \sqsubseteq \exists onhook_u.(idle_u \sqcap idle_v)$ (if u goes onhook, she and her party go idle);
- $path_{uv} \sqsubseteq \exists onhook_u.(idle_u \sqcap busytone_v)$ (u can go onhook when talking to v).

Given the functionality of roles, these axioms also produce universal restrictions, i.e., each $\exists$ also acts as a $\forall$.

There is only one ABox statement corresponding to the "initial state" of the system; we require that at state s_0 all users are idle,

$$s_0 : \bigsqcap_{u\in\mathsf{SUB}} idle_u.$$

We should now enforce certain properties which are true in any model of the *BCS*, like the fact that each subscriber is in exactly one state at each moment, and that subscribers can change state only by means of certain actions. The first is expressed as follows by requiring that for all $u \in \mathsf{SUB}$:

$$\top \sqsubseteq \neg \left(\bigsqcup_{s_1,s_2\in\mathsf{ST}_u,s_1\neq s_2}(s_1 \sqcap s_2)\right) \sqcap \bigsqcup_{s\in\mathsf{ST}_u} s.$$

As to the second property, because the set of subscribers and the allowed transition functions are finite, we can explicitly enforce this condition. Define the following notation linking states to the actions they allow. Let D_{uv} be $\{idle_u,\ dialtone_u,\ busytone_u,\ ringing_u \sqcap calling_{vu},\ calling_{uv},\ path_{uv}\}$ for $u, v \in \mathsf{SUB}$. We now define a mapping Act_{uv} from D_{uv} to the subsets of $\mathsf{ROLES}_u \cup \mathsf{ROLES}_v$ as a function specifying the allowed actions:

$$\mathsf{Act}_{uv}(idle_u) = \{offhook_u\} \cup \{dial_{vu} \mid v \in \mathsf{SUB}, v \neq u\}$$
$$\mathsf{Act}_{uv}(dialtone_u) = \{onhook_u\} \cup \{dial_{uv} \mid v \in \mathsf{SUB}, v \neq u\}$$
$$\mathsf{Act}_{uv}(busytone_u) = \{onhook_u\}$$
$$\mathsf{Act}_{uv}(ringing_u \sqcap calling_{vu}) = \{onhook_v\}$$
$$\mathsf{Act}_{uv}(calling_{uv}) = \{onhook_u, offhook_v\}$$
$$\mathsf{Act}_{uv}(path_{uv}) = \{onhook_u, onhook_v\}$$

Then we require that for all $u,\ v \in \mathsf{SUB}$, all $s \in D_{uv}$, and all $R \notin \mathsf{Act}_{uv}(s)$, $s \sqsubseteq \forall R.s$.

We have now completed our formalization of the basic call system in $\mathcal{FI}$. Next we turn to the analysis of feature interaction.

3.2 Step 2: Adding Features

We will now indicate how to extend the *BCS* specification with features. This will be done through refinements. As an example we will add three standard features (*TCS*, *CFU* and *CW*). Furthermore, we formally define feature interaction as a satisfaction problem.

Defining what exactly a feature is, is not a simple task. A very abstract approach is to consider a feature as a pair $\langle\delta, \varphi\rangle$ where δ is a function mapping specifications to specifications (and representing the action of adding the feature to a basic system), and φ is a formal description of the expected behavior of the feature. Then, for example, if *TCS*= $\langle\delta, \varphi\rangle$ is the specification of the Terminating Call Screening feature, it would be expected that $\delta(BCS) \models \varphi$, or in other words, that adding TCS on the Basic Call System achieves the expected behavior. δ can be viewed as the implementation function, while φ represents the feature specification.

Here we will take a more concrete approach. Notice that separating the implementation function from the expected behavior leaves open the possibility of wrong implementations, i.e., given an original specification S, it might turn out that $\delta(S)$ does not satisfy the specification, i.e., $\delta(S) \not\models \varphi$. Because we don't want to consider this possibility, we will drop φ from the definition of a feature and simply view it as a refining operator. Furthermore, and for the sake of simplicity, we will restrict ourself to order independent refinements, e.g., given two refinements δ_1 and δ_2, for any specification S, $\delta_1(\delta_2(S))$ is equivalent to $\delta_2(\delta_1(S))$. This will let us achieve a clear definition of interaction.

Definition 3.1 (Refinement and Interaction) For a given feature F we will define a refinement operation δ_F, and a set of new concepts $\mathsf{A}_F \subseteq \{F_{\bar{u}} \mid \bar{u} \in \mathsf{SUB}^+\}$[1] (the *activation concepts*). Given a knowledge base K, $\delta_F(K)$ is the knowledge base obtained by applying the refinement to K. We say that K and F *interact on activation* N for $N \subseteq \mathsf{A}_F$ if $\delta_F(K) \cup N^c \cup N^d \models \neg\top$, where N^c is the set of *connected features* $N^c = \{\top \sqsubseteq F_{\bar{u}} \mid F_{\bar{u}} \in N\}$ and N^d is the set of *disconnected features* $N^d = \{\top \sqsubseteq \neg F_{\bar{u}} \mid F_{\bar{u}} \in \mathsf{A}_F \backslash N\}$.

Notice that the formalism gives us the flexibility to model different activation possibilities by modifying the activation sets. Trivially, refinements can be iterated. If F_1 and F_2 are features then $\delta_{F_2}(\delta_{F_1}(K))$ denotes the successive refinements through F_1 and F_2, which we denote as $F_1 \circ F_2$. Remember that we will only consider features such that, $F_1 \circ F_2$ is always equivalent to $F_2 \circ F_1$.

When K is the basic call service *BCS*, we say that features F_1 and F_2 *interact* if there exist activation sets $N_{F_1} \subseteq \mathsf{A}_{F_1}$ and $N_{F_2} \subseteq \mathsf{A}_{F_2}$ such that *BCS* and $F_1 \circ F_2$ interact on activation $N_{F_1} \cup N_{F_2}$.

Terminating Call Screening (*TCS*). *TCS* is a feature where a user u can put another user v on a black list: v's phone is not allowed to establish a connection to u's phone. A first attempt to formalize this behavior is by introducing a new concept TCS_{uv}, and simply refining *BCS* by adding $TCS_{uv} \sqsubseteq \neg calling_{vu}$. However, this extension immediately interacts (with itself on activation $\{TCS_{uv}\}$). This is an *expected* interaction, though. In fact, adding a feature *always* modifies, and hence contradicts, the basic system (if the basic system has been completely modeled). To obtain a correct refinement, first define the set of activation concepts for *TCS* to be $\{TCS_{uv} \mid u, v \in \mathsf{SUB}, u \neq v\}$. The refinement δ_{TCS} is defined by replacing, in any knowledge base, terminologies of the form $\mathbf{C}_1[dialtone_u \sqcap idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[calling_{uv}]$ by

$$\begin{array}{l}
\mathbf{C}_1[\neg TCS_{vu} \sqcap dialtone_u \sqcap idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[calling_{uv}] \\
\mathbf{C}_1[TCS_{vu} \sqcap dialtone_u \sqcap idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[busytone_u] \\
TCS_{vu} \sqsubseteq \neg calling_{uv}.
\end{array}$$

Here $\mathbf{C}_1$ and $\mathbf{C}_2$ are contexts. In general $\mathbf{C}[\varphi]$ singles out an occurrence of φ as a subformula of any formula ψ. We will take care that δ_{TCS} is well defined by applying it only to knowledge bases where contexts single out unique occurrences.

Call Forwarding Unconditional (*CFU*). The informal meaning of CFU_{uv} is that whenever w calls u, she will be connected to v instead. For the activation concepts we take the set $\{CFU_{uv} \mid u, v \in \mathsf{SUB}, u \neq v\}$. We formalize δ_{CFU} by replacing axioms of the form $\mathbf{C}_1[dialtone_u \sqcap idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[calling_{uv}]$ and $\mathbf{C}_3[dialtone_u \sqcap \neg idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_4[busytone_u]$ by

$$\begin{array}{l}
\mathbf{C}_1[\neg \sqcup_{w \in \mathsf{SUB}} CFU_{vw} \sqcap dialtone_u \sqcap idle_v] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_2[calling_{uv}], \\
\mathbf{C}_1[CFU_{vw} \sqcap dialtone_u \sqcap idle_w] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_2[calling_{uw}], \\
\mathbf{C}_3[\neg \sqcup_{w \in \mathsf{SUB}} CFU_{vw} \sqcap dialtone_u \sqcap \neg idle_v] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_4[busytone_u], \\
\mathbf{C}_3[CFU_{vw} \sqcap dialtone_u \sqcap \neg idle_w] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_4[busytone_u].
\end{array}$$

CFU affects also the definition of the Act function. For example, if u has CFU_{uv} for some v, then for all w, $dial_{wu}$ will actually not modify her state (as the call will be forwarded to v). Modifying Act appropriately is straightforward.

Some very interesting issues surface when formally defining this feature. Notice that the forwarding event is implemented by changing the state *calling*$_{uv}$ to *calling*$_{uw}$ whenever CFU_{vw} is active (together with some book keeping). But this bypasses the *dial*$_{uv}$ action. If, in addition, CFU_{wx} is active this second forwarding will not be executed. In other words, the specification actually encodes a policy of restricting the number of possible forwarding to only one. To implement multiple forwarding, *network states forwarding*$_{uv}$ should be added and the refinement modified as follows. δ_{TCS} is defined by replacing, in any knowledge base, terminologies of the form $\mathbf{C}_1[dialtone_u \sqcap idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[calling_{uv}]$ and $\mathbf{C}_3[dialtone_u \sqcap \neg idle_v] \sqsubseteq \exists dial_{uv}.\mathbf{C}_4[busytone_u]$ by

$$\begin{array}{l}
\mathbf{C}_1[\neg \sqcup_{w \in \mathsf{SUB}} CFU_{vw} \sqcap dialtone_u \sqcap idle_v] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_2[calling_{uv}], \\
\mathbf{C}_1[CFU_{vw} \sqcap dialtone_u] \sqsubseteq \exists dial_{uv}.\mathbf{C}_2[forwarding_{uw}], \\
\mathbf{C}_3[\neg \sqcup_{w \in \mathsf{SUB}} CFU_{vw} \sqcap dialtone_u \sqcap \neg idle_v] \sqsubseteq \\
\qquad \exists dial_{uv}.\mathbf{C}_4[busytone_u], \\
\neg \sqcup_{x \in \mathsf{SUB}} CFU_{wx} \sqcap forwarding_{uw} \sqcap idle_w \sqsubseteq calling_{uw}, \\
\neg \sqcup_{x \in \mathsf{SUB}} CFU_{wx} \sqcap forwarding_{uw} \sqcap \neg idle_w \sqsubseteq busytone_u, \\
CFU_{wx} \sqcap forwarding_{uw} \sqsubseteq forwarding_{ux}.
\end{array}$$

[1]For any set S, S^+ denotes the set of nonempty strings over S.

Notice that the new network states are not *user states* (i.e., they are not members of ST). This is crucial, as in our model the network will be going through multiple *forwarding* states till it reaches a user w which does not have the call forwarding feature activated.

In the new specification there is no limit on the number of times a call can be forwarded. This, of course, might lead to infinite loops. Properties like this can be detected by the calculus introduced in Section 4.

Call Waiting (*CW*). The specification of the *CW* feature is much more involved than the ones we have discussed so far. The intended behavior of *CW* is as follows. Suppose subscriber A has *CW* activated, and A is speaking with B. If C calls A, A should receive a notification of a waiting second call (A will hear a signal while speaking with B). He can then decide to put B on hold by doing a flash hook and attend to C's call. He can continue to swap between the parties by doing further flash hooks. In the end, if the subscriber does an on hook while a party is still on hold, he will be rung back from the held party.

It is immediate from the informal description that CW adds both new states (e.g., *onhold*) and new actions (e.g., *flashhook*), and hence its specification asks for a more elaborated knowledge base. It should be clear however, that once we have specified the different states the subscribers and the system will pass through, together with the allowed actions, the model can be encoded in $\mathcal{FI}$ without further inconveniences. We don't provide full details but, for example, by adding the following formulas to the TBox we can model some of the actions which are possible for a user which has the *CW* feature enabled.

$$\begin{array}{l}
CW_x \sqcap path_{xy} \sqsubseteq \exists dial_{zx}.cwalerting_{xyz} \\
CW_x \sqcap cwalerting_{xyz} \sqsubseteq \exists flashhook_x.onhold_{xzy} \\
CW_x \sqcap onhold_{xzy} \sqsubseteq \exists flashhook_x.onhold_{xyz} \\
CW_x \sqcap onhold_{xyz} \sqsubseteq \exists onhook_x.calling_{zx} \sqcap busytone_y
\end{array}$$

4. Using $\mathcal{FI}$ for Detecting Feature Interaction.

Now let us take stock. First, it is important to remark that many of the properties spelled out in Section 3 were enforced only informally (or not at all) in previous proposals, like [10]. Our modeling effort revealed a number of hidden assumptions. Second, we can now use formal results available about description logics to obtain information about the properties of $\mathcal{FI}$ and, in particular, about feature interaction.

How? For a start, $\mathcal{FI}$ is just a subset of a well known description logic which is called $\mathcal{ALC}$ with assertions and cyclic definitions. In [8], De Giacomo et al. define a tableau method to prove consistency of knowledge bases in $\mathcal{ALC}$ with assertions and cyclic definitions, which is EXPTIME complete (even though the paper only mentions an EXPTIME upper bound we can easily encode global satisfiability of the basic modal logic **K** in this logic). What does this imply for the complexity of determining feature interaction? Because we have defined this task as a special satisfiability problem, it follows that determining feature interaction (as it is modeled here) is decidable in EXPTIME. Furthermore, even though $\mathcal{FI}$ is a proper fragment of full $\mathcal{ALC}$ with cyclic TBoxes and assertions, it actually has full $\mathcal{ALC}$ expressivity: the full fragment can be encoded in $\mathcal{FI}$ by means of definitions. Hence, EXPTIME is also a lower bound for deciding satisfiability in $\mathcal{FI}$.

4.1 Decision Methods for $\mathcal{FI}$

In this section we provide details on the tableau method which can be used to detect feature interaction. The following tableaux method is adapted to $\mathcal{FI}$ from [8].

Definition 4.1 (Tableaux Method) A tableau for $\mathcal{FI}$ can be represented as a set of formulas with added prefixes of the form $\langle b \mid p : C\rangle$ where the *segment* b is a binary string representing choices in a binary branching tree; the *element* p is a string alternating integers (names for individuals) and role names; and C is an $\mathcal{FI}$ concept.

Define the following notation: given two strings σ_1 and σ_2, $\sigma_1 \preceq \sigma_2$ ($\sigma_1 \prec \sigma_2$) means that σ_1 is a prefix (strict prefix) of σ_2. A segment b_M is maximal for another segment b in a tableau T if both b_M and b are present in T and b_M is the longest segment in T of which b is a prefix.

Let $\langle T, A\rangle$ be the knowledge base obtained by considering a given refinement of *BCS* together with a particular activation set N. Suppose we want to check whether $\langle T, A\rangle$ interact on activation N (e.g., $\langle T, A\rangle \models \neg\top$). Initialize the tableau with $\langle \varepsilon \mid s_0 : \bigsqcap_{u\in\mathsf{SUB}} idle_u\rangle$ (where ε is the empty string) and apply the following rules.

AND:	$\dfrac{\langle b \mid p : C \sqcap D\rangle}{\langle b \mid p : C\rangle \quad \langle b \mid p : D\rangle}$	
OR:	$\dfrac{\langle b \mid p : C \sqcup D\rangle}{\langle b_M 0 \mid p : C\rangle \quad \langle b_M 1 \mid p : D\rangle}$	with b_M maximal for b
SOME:	$\dfrac{\langle b \mid p : \exists R.C\rangle}{\langle b \mid pRn : C\rangle}$	with pRn new
ALL:	$\dfrac{\langle b \mid p : \forall R.C\rangle}{\langle b \mid pRn : C\rangle}$	with pRn present in b
KB:	$\dfrac{\vdots}{\langle b \mid p : \neg C \sqcup D\rangle}$	with p present in b and $C \sqsubseteq D \in T$

To handle functional rules, the application of the SOME rule is restricted by the side condition requiring that no R successor of p already exists in the branch.

If we also provide the correct definition of when a branch in the tableau is closed together with a heuristic of how the rules should be applied, termination is guaranteed (see [8] for details). In addition, the decision method is sound and complete, i.e., all the branches in the tableau are closed if and only if the knowledge base is inconsistent.

4.2 Example: Interaction between *TCS* and *CFU*.

Intuitively, interaction between *TCS* and *CFU* might occur as follows. Suppose B forwards his phone to C and C puts A on his screening list. Now A calls B, and because B has forwarded his calls to C and he is not on C's screening list, a connection is established from A to C! This is a bad interaction which can be formally detected: $\delta_{CFU}(\delta_{TCS}(BCS))$ interacts on activation $\{TCS_{CA}\} \cup \{CFU_{BC}\}$. We formally prove our claim that the interaction sketched is detected by the tableaux calculus. The derivation will produce a closed tableau; see below. Note that only the application of the OR rule in the calculus gives rise to new branches.

1. Applying AND to the start formula $\langle \varepsilon \mid s_0 : \bigsqcap_{u\in\mathsf{SUB}} idle_u\rangle$ gives $\langle \varepsilon \mid s_0 : idle_A\rangle$.
2. An intermediate derivation produces:
   (a) applying KB to rule $idle_A \sqsubseteq \exists offhook_A.dialtone_A$ of $\mathcal{T}$ produces $\langle \varepsilon \mid s_0 : \neg idle_A \sqcup \exists offhook_A.dialtone_A\rangle$
   (b) Applying OR gives two branches:
      i. $\langle 0 \mid s_0 : \neg idle_A\rangle$. This one is already closed because of the appearance of $\langle \varepsilon \mid s_0 : idle_A\rangle$ earlier in this segment,
      ii. and $\langle 1 \mid s_0 :\mid \exists offhook_A.dialtone_A\rangle$,
   (c) and finally by applying SOME: $\langle 1 \mid s_0\ offhook_A\ s_1 : dialtone_A\rangle$, with s_1 a new state.
3. We derive $\langle \varepsilon \mid s_0 : \neg idle_B \sqcup \forall offhook_A.idle_B\rangle$ in an intermediate step, by applying KB to the frame axioms. Then, applying OR:
   (a) $\langle 10 \mid s_0 : \neg idle_B\rangle$ which is again already closed, and
   (b) $\langle 11 \mid s_0 : \forall offhook_A.idle_B\rangle$ to which we apply rule ALL, resulting in
   (c) $\langle 11 \mid s_0\ offhook_A\ s_1 : idle_B\rangle$
4. We need a further intermediate derivation. Applying KB to rule $dialtone_A \sqcap idle_B \sqsubseteq \exists dial_{AB}.calling_{AC}$ of the (CFU-modified) TBox gives: $\langle 11 \mid s_0\ offhook_A\ s_1 : \neg dialtone_A \sqcup \neg idle_B \sqcup \exists dial_{AB}.calling_{AC}\rangle$. Apply OR:
   (a) $\langle 110 \mid s_0\ offhook_A\ s_1 : \neg dialtone_A\rangle$ which is already closed, and
   (b) $\langle 111 \mid s_0\ offhook_A\ s_1 : \neg idle_B \sqcup \exists dial_{AB}.calling_{AC}\rangle$

   Apply OR again:
   (a) $\langle 1110 \mid s_0\ offhook_A\ s_1 : \neg idle_B\rangle$ (closed), and
   (b) $\langle 1111 \mid s_0\ offhook_A\ s_1 : \exists dial_{AB}.calling_{AC}\rangle$, and finally SOME again, producing
   (c) $\langle 1111 \mid s_0\ offhook_A\ s_1\ dial_{AB}\ s_2 : calling_{AC}\rangle$ which closes the last open segment of the tableau. This is proved by applying KB to rule $TCS_{CA} \sqsubseteq \neg calling_{AC}$ of the CFU-modified TBox, giving $\langle 1111 \mid s_0\ offhook_A\ s_1\ dial_{AB}\ s_2 : \neg TCS_{CA} \sqcup \neg calling_{AC}\rangle$, and finally by OR:
      i. $\langle 11110 \mid s_0\ offhook_A\ s_1\ dial_{AB}\ s_2 : \neg TCS_{CA}\rangle$ which is closed by TCS_{CA} itself, and
      ii. $\langle 11111 \mid s_0\ offhook_A\ s_1\ dial_{AB}\ s_2 : \neg calling_{AC}\rangle$ which finishes the closure.

As we obtain a closed tableau, $\delta_{CFU}(\delta_{TCS}(BCS))$ interacts on activation $\{TCS_{CA}\} \cup \{CFU_{BC}\}$. It is important to realize that the closed tableau can be obtained mechanically, and hence feature interaction automatically detected.

5. Related work

We briefly discuss previous work that is connected with our approach to understanding feature interaction.

Gammelgaard and Kristensen's paper [10] has been the starting point for the current contribution. The authors introduce a formal language for specifying theories consisting of global propositional formulas (*network properties*) and (propositional) transition rules of the form $p - t \rightarrow q$, where transitions are labeled by symbols from a designated set L of *trigger symbols*. Some examples are:

$$
\begin{array}{l}
calling(A,B) \Rightarrow ringing(B) \wedge ring\_back(A) \\
path(A,B) \Rightarrow engaged(A) \wedge engaged(B) \\
idle(A) - offhook(A) \rightarrow dialtone(A) \\
TCS(A,B) \Rightarrow \neg calling(B,A)
\end{array}
$$

Here, the first three formulas are more or less self-explanatory (for example, $calling(A,B)$ is just a propositional symbol and $\Rightarrow$ is material implication), and the last one is a so-called *activation predicate*, specifying that user A has an active Terminating Call Screening feature, and has put user B on her black list.

A semantics for such specifications is provided by introducing deterministic labeled transition systems (LTSs) equipped with a propositional valuation on states, as candidate models. Network properties are required to hold globally (in all states), and a transition rule $p - t \rightarrow q$ holds iff for all states s we have that if p holds in s, then there exists a transition labeled with t leading to a state where q holds. An LTS *M* is a *model* for a specification *F* iff it satisfies all axioms of *F* and the following *initial state* condition holds: there is a state in which $idle(A)$ holds for each subscriber A. A specification is *satisfiable* iff it has a model.

Features are introduced at two levels, by an informal description, and formally by activation predicates as discussed above. An activation predicate is an instance of a network property scheme; it represents a feature that is active in the network, which means that it is active in every state. Features that are not explicitly active are supposed to be switched off. Using these activation predicates, Gammelgaard and Kristensen give several definitions: features can interact with the basic call service, with themselves, or with other features. The set-up is illustrated with examples.

The detection done in the approach sketched above is informal. There is no calculus for the language, and the arguments analyzing feature interaction are a kind of semantic evaluation of models. Given the intricacies of the subject area, this is bound to lead to errors. Moreover, unless we have a proper formalization, we have no way of understanding the complexity of the feature interaction problem, and we have no way of using formal tools to help us detect interaction. Our contribution is that we address these shortcomings by representing network properties and transition rules in an appropriate description logic and by expressing feature interaction as a satisfiability problem for which decision methods and complexity results are available.

6. Conclusion

We have provided a formal method for feature interaction detection. Our method formalizes some of the ideas first introduced in [10]. Our modeling enterprise has revealed various inaccuracies and ambiguities in previous approaches as described in Section 5. In addition, we have established an EXPTIME upper bound for the problem of detecting feature interaction, indicating that logical approaches to feature interaction are probably expensive.

Where to go from here? There is a clear line of logic-related questions, and a line of questions related to feature interaction. Since our formalization of the interaction problem does not need the full expressivity of $\mathcal{FI}$ (disjunctions of concepts were never used), there is some hope that the problem falls inside a lower complexity class than EXPTIME. Another option is to adapt the general tableaux calculus to the specific problem of detecting feature interaction in order to improve its complexity. Also, it seems interesting to further investigate the nature of our refinement operator, and to explore notions of refinement in the literature. As to interaction issues, a classification in terms of how they should be specified seems interesting. For example, *TCS* does not introduce new states while new network states are needed to model *CFU*. The more complex Call Waiting feature needs both new actions and new user states. Implementing the scheme we laid out in this paper in a theorem prover for description logics is a top priority which will certainly provide new insights into modeling the feature interaction problem. Preliminary experiments with the RACE system developed by Haarslev and Möller [11] show that each subscriber adds an exponential number of new interactions; that is, the time needed to check the consistency of the basic call service increases exponentially with the number of subscribers — as is to be expected. Finally, there is room for improving the knowledge bases that describe the basic call service (possibly modified by features); we are currently duplicating the descriptions of subscribers without even trying to exploit the fact that these descriptions are virtually identical. Given the exponential behavior noted above, it may be worthwhile to develop more sophisticated representations that exploit the similarities.

References

[1] Bellcore. LATA switching systems generic requirements (LSSGR). Tech. Reference TR-TSY-000064, Bellcore, Piscataway, N.J., 1992.

[2] J. Blom. Formalisation of requirements with emphasis on feature interaction detection. In Dini et al. [9], pages 61–77.

[3] L. Bouma and H. Velthuijsen, editors. *Feature Interactions in Telecommunication Systems*, Amsterdam, Oxford, Washington DC, Tokyo, 1994. IOS Press.

[4] T. Bowen, F. Dworack, C. Chow, N. Griffeth, and Y. Lin. The feature interaction problem in telecommunication systems. In *Proc. 7th Int. Conf. Software Engineering for Telecommunication Switching Systems*, pages 59–62, Bournemouth, UK, 1989.

[5] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[6] E. Cameron, N. Griffeth, Y. Lin, M. Nilson, W. Schnure, and H. Velthuijsen. A feature interaction benchmark for IN and beyond. In Bouma and Velthuijsen [3], pages 1–23.

[7] J. Cameron and H. Velthuijsen. Feature interactions in telecommunications systems. *IEEE Communications Magazine*, 31(8):18–23, August 1993.

[8] G. De Giacomo, F. Donini, and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. In *Proceedings International Workshop on Description Logics (DL'96)*, pages 107–110, Cambridge, MA, USA., November 1996.

[9] P. Dini, R. Boutaba, and L. Logrippo, editors. *Feature Interactions in Telecommunication Systems, III*, Amsterdam, Oxford, Washington, Tokyo, 1997. IOS Press.

[10] A. Gammelgaard and J. Kristensen. Interaction detection, a logical approach. In Bouma and Velthuijsen [3], pages 178–196.

[11] V. Haarslev and R. Möller. RACE system description. In *Proceedings International Workshop on Description Logics (DL'99)*, 1999.

[12] D. Keck and P. Kuehn. The feature and service interaction problem in telecommunications systems: A survey. *IEEE Trans. Software Engineering*, 24(10), 1998.

[13] K. Kimbler. Addressing the interaction problem at the enterprise level. In Dini et al. [9], pages 13–22.