Description Logics and Feature Interaction

Carlos Areces[1] and **Wiet Bouma**[2] and **Maarten de Rijke**[1]

[1] ILLC, University of Amsterdam, Plantage Muidergracht 24,
1018 TV Amsterdam, The Netherlands
E-mail: {carlos, mdr}@wins.uva.nl
[2] KPN Research, PO Box 421
2260 AK Leidschendam, The Netherlands
E-mail: L.G.Bouma@research.kpn.com

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. By using standard techniques from description logic reasoning, properties of features can be proved and interactions detected. An EXPTIME upper bound for the complexity of detecting feature interaction as a satisfiability problem is obtained by well known result 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. In [Bowen *et al.*, July 1989], 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, Three-Way Calling, Reverse- and Split Charging, etc., but there are many other feature domains in the telecommunications area like data- and multi-media communication.

Feature *interaction* arises when the behavior of one feature influences the behavior of another, mostly in an unexpected and unwanted way. A first example is the combination of Call Waiting and Call Forwarding on Busy: if the network gives priority to one the other is effectively disabled. In recent years, the problem has 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, the need to rapidly develop new features on top of already installed ones, and the distributed nature of software for telecommunications systems.

There are many ways to approach the problem of detecting feature interaction. One can take the software-engineering point of view, and qualify it as a software maintenance, legacy or re-use problem. Another interesting route is to treat it as a design problem, and devise ways to deal with it during design time. A specific idea here is to do a formal design of features, and use these formal descriptions to detect interaction. This is the approach taken in this paper. There have been many contributions to the literature on this topic [Bouma, 1999, Keck and Kuehn, 1998], but most of them model a network-oriented approach. By contrast the *logic* approach, where features are described in an appropriate formal logic and the definition of interaction is formulated as a reasoning task, are comparatively rare.

Formal Languages for the Description of Features. In [Gammelgaard and Kristensen, 1994] a formal language is introduced to describe features. In their terminology, a *specification* is a set of *axioms*, where an axiom is either a material implication (a *network property*), or a *transition rule* of the form $p - t \rightarrow q$, with the informal interpretation that if in a state, say s, p holds, then there can be a state transition from s, triggered by t to another state, say s', such that q holds in s'. Transitions are labeled by symbols from a designated set L of *trigger symbols*. Some examples of axioms are:

$$\begin{gathered} \mathit{calling}(A,B) \Rightarrow \mathit{ringing}(B) \wedge \mathit{ringback}(A), \\ \mathit{TCS}(A,B) \Rightarrow \neg \mathit{calling}(A,B), \\ \mathit{idle}(A) - \mathit{offhook}(A) \rightarrow \mathit{ready}(A), \end{gathered}$$

where the first and third formula have a clear informal meaning (for example, $\mathit{calling}(A,B)$ is just a propositional symbol and $\Rightarrow$ is material implication), and the second one is an *activation predicate*, specifying that user A has an active Terminating Call Screening feature, and has put user B on her blacklist: user A should never find herself in a state where she is calling user B. These examples are actually axiom schemes, where A, B range over the set SUBS of the *subscribers* to the telephony system.

Semantics is provided by means of labeled transition systems (LTSs) $(St, L, \rightarrow)$, where St is a non-empty set, L is a set of labels and $\rightarrow$ is a set of (deterministic) binary relations on St, one for each label in L; together with a valuation $V : \mathbb{P} \rightarrow Pow(St)$. An LTS *satisfies a network property* ϕ if $V(\phi) = St$, and it *satisfies a transition rule* $p - t \rightarrow q$ if for all $s \in St$: if $s \in V(p)$ then there exists $s' \in St$ such that $(s, s') \in -t\rightarrow$ and $s' \in V(q)$.

After specifying the Basic Call system (BCS), features are introduced at two levels, first by an informal description and then, formally, by activation predicates expressing the operation of a feature on top of BCS. 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. For example, two features *interact with each other* if BCS together with their respective activation predicates are unsatisfiable, while each of them is satisfiable in isolation. Finally, the set-up is illustrated with several examples.

The detection done in the approach sketched above is mostly informal. [Gammelgaard and Kristensen, 1994] does not provide a 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. To address these shortcomings, we proceed as follows in this paper: (1) we will formalize the approach above so that standard reasoning methods can be applied; and (2) we propose a "good" logic to reason about features and the interaction problem, with sufficient expressive power and with sound and complete decision methods.

2 Description Logics for Feature Interaction

We will propose a description logic as a suitable formalism to describe feature interaction.

Definition 1 (The Description Logic $\mathcal{FI}$) $\mathcal{FI}$ is a syntactic restriction of $\mathcal{ALC}$ with TBoxes, ABoxes and cyclic terminologies. Fix a signature $\tau = \langle \mathsf{C}, \mathsf{R}, \mathsf{A} \rangle$ of concept names, role names and constants. The sets CON of concepts, TERM of terminological axioms and ASSER of assertional axioms is defined as follows:

$$
\begin{array}{rcl}
\mathsf{BASIC\text{-}CON} & := & C \mid \neg\mathsf{BASIC\text{-}CON} \mid \\
 & & \mathsf{BASIC\text{-}CON} \sqcap \mathsf{BASIC\text{-}CON} \\
\mathsf{CON} & := & \mathsf{BASIC\text{-}CON} \mid \forall R.\mathsf{BASIC\text{-}CON} \mid \\
 & & \exists R.\mathsf{BASIC\text{-}CON} \\
\mathsf{TERM} & := & \mathsf{BASIC\text{-}CON} \sqsubseteq \mathsf{CON} \mid \\
 & & \mathsf{BASIC\text{-}CON} \doteq \mathsf{CON} \\
\mathsf{ASSER} & := & a : \mathsf{CON} \mid (a, b) : R,
\end{array}
$$

where $C \in \mathsf{C}$, $R \in \mathsf{R}$ and $a, b \in \mathsf{A}$. A *knowledge base* K is a tuple $\langle T, A \rangle$ where $T \subseteq \mathsf{TERM}$ and $A \subseteq \mathsf{ASSER}$. We adopt a descriptive semantics — defined by means of an interpretation function — where TBox statements are considered as conditions over models; see [Buchheit *et al.*, 1993].

Once the formal semantics has been defined, it is easy to see that we can capture *network properties* and *transition rules*. The former are TBox axioms of the form $\mathsf{BASIC\text{-}CON} \sqsubseteq \mathsf{BASIC\text{-}CON}$, while the later are of the form $\mathsf{BASIC\text{-}CON} \sqsubseteq \mathsf{CON}$.

2.1 Description of BCS in $\mathcal{FI}$

In this section we model the Basic Call Service (BCS) in $\mathcal{FI}$. We first fix the signature describing atomic concepts and roles and their intended meaning. Assume a finite set SUBS of indexes which represent subscribers to the telephone system. Most cases of feature interaction can be detected already in the presence of a small number of subscribers.

Atomic Concepts. First, concepts expressing that a user $u \in \mathsf{SUBS}$ is in a specific state are needed. This is expressed by the following concepts which we call the set of possible STATES_u of a user u: *idle_u* (the receiver is on hook and silent), *dialtone_u* (the receiver is off hook and emits a dial tone), *busytone_u* (the receiver is off hook and emits a busy tone), *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) and *engaged_u* (there is a speech path with the other party). Additionally, we need concepts expressing a (minimal) presence of the network, because a call proceeds through phases that are not directly perceivable to a user. These phases correspond to internal states of the network. Moreover, we need to be able to express connections between users being active. These, for $u, v \in \mathsf{SUBS}$, form the set $\mathsf{ISTATES}_{uv}$ of internal states: *ready_u* (the phone at u is ready to accept digits), *calling_u_v* (the phone at v is ringing with u waiting for v to accept the call), *rejecting_u* (u's called party is busy or has put the receiver on hook) and *path_u_v* (u and v can communicate). Let C be defined as $\mathsf{C} = \bigcup_{u,v \in \mathsf{SUBS}} (\mathsf{STATES}_u \cup \mathsf{ISTATES}_{uv})$.

Atomic Roles. Roles are the translation of trigger symbols. Here we introduce the set of atomic roles, again with short informal explanations. We put $\mathsf{R} = \bigcup_{u \in \mathsf{SUBS}} \mathsf{ROLES}_u$, where ROLES_u is defined by *offhook_u* (u's action of lifting the receiver), *dial_u_v* (u's action of dialing v's number) and *onhook_u* (u's action of putting down the receiver). Roles are interpreted as partial functions, thus accounting for the determinism in the system being modeled.

TBox and ABox. Now that concept and role names have been introduced, we can define a knowledge base describing *BCS*. Again, we use the convention that the statements presented below are schemes where u and v range over a finite set SUBS of subscribers. First, there are several (TBox) expressions connecting the observable states of a telephone with the ones representing network states:

$$\begin{array}{rcll}
\mathit{dialtone_u} & \sqsubseteq & \mathit{ready_u} & \\
\mathit{calling_u_v} & \sqsubseteq & \mathit{ringing_v} \sqcap \mathit{ringback_u}, & u \neq v \\
\mathit{ringing_u} & \sqsubseteq & \sqcup_{v \in \mathsf{SUBS}, v \neq u} \mathit{calling_v_u} & \\
\mathit{ringback_u} & \sqsubseteq & \sqcup_{v \in \mathsf{SUBS}, v \neq u} \mathit{calling_u_v} & \\
\mathit{rejecting_u} & \doteq & \mathit{busytone_u} & \\
\mathit{path_u_v} & \sqsubseteq & \mathit{engaged_u} \sqcap \mathit{engaged_v}, & u \neq v \\
\mathit{engaged_u} & \sqsubseteq & \sqcup_{v \in \mathsf{SUBS}, v \neq u} \mathit{path_u_v} & \\
\mathit{path_u_v} & \doteq & \mathit{path_v_u}. &
\end{array}$$

Second, there are statements specifying how a user and the network can change state:

$$\begin{array}{rcl}
\mathit{idle_u} & \sqsubseteq & \exists \mathit{offhook_u}.\mathit{ready_u} \\
\mathit{ready_u} \sqcap \mathit{idle_v} & \sqsubseteq & \exists \mathit{dial_u_v}.\mathit{calling_u_v} \\
\mathit{ready_u} & \sqsubseteq & \exists \mathit{onhook_u}.\mathit{idle_u} \\
\mathit{ready_u} \sqcap \neg \mathit{idle_v} & \sqsubseteq & \exists \mathit{dial_u_v}.\mathit{rejecting_u} \\
\mathit{rejecting_u} & \sqsubseteq & \exists \mathit{onhook_u}.\mathit{idle_u} \\
\mathit{calling_u_v} & \sqsubseteq & \exists \mathit{offhook_v}.\mathit{path_u_v}, \quad u \neq v \\
\mathit{calling_u_v} & \sqsubseteq & \exists \mathit{onhook_u}.(\mathit{idle_u} \sqcap \mathit{idle_v}) \\
\mathit{path_u_v} & \sqsubseteq & \exists \mathit{onhook_u}.(\mathit{idle_u} \sqcap \mathit{rejecting_v}), \\
 & & u \neq v.
\end{array}$$

Given the functionality of roles, these axioms also produce universal restrictions.

There is only one A-box statement corresponding to the "initial state" of the system: we require that at state s_0 all users are idle,

$$s_0 : \bigsqcap_{u \in \mathsf{SUBS}} \mathit{idle\_u}.$$

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 they can change state only by means of certain actions. The first is expressed as follows: For all $u \in \mathsf{SUBS}$,

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

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. If D is the set of conditions (boolean concepts over $\mathsf{STATES} \cup \mathsf{ISTATES}$), we can define Actions : $D \to \mathit{Pow}(\mathsf{R})$ as a function specifying the allowed actions. Then we require: For all $s \in D$ and all $R \notin \text{Actions}(s)$, we set $s \sqsubseteq \forall R.s$.

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*, *ringing_u* $\sqcap$ *calling_v_u*, *ringback_u*, *ringback_u* $\sqcap$ *calling_u_v*, *engaged_u*, *engaged_u* $\sqcap$ *path_u_v*, *ready_u*, *calling_u_v*, *rejecting_u*, *path_u_v*$\}$ for $u, v \in \mathsf{SUBS}$. Define the mapping $\text{Actions}_{uv} : D_{uv} \to \mathit{Pow}(\mathsf{R})$ — the function specifying the allowed actions — as in Figure 1.

Then we require:

$$\begin{array}{ll}
\text{For all } u, v \in \mathsf{SUBS}, & \\
\text{for all } s \in D_{uv}, & s \sqsubseteq \forall R.s. \\
\text{for all } R \notin \text{Actions}_{uv}(s), &
\end{array}$$

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

2.2 Adding Features

Now that we have completed our formalization of *BCS*, let us take stock. First, many of the properties spelled out above were enforced only informally (or not at all) in the models of [Gammelgaard and Kristensen, 1994]; our modeling effort revealed a number of hidden assumptions. Second, we can use formal results about $\mathcal{ALC}$ to get results about $\mathcal{FI}$ and about feature interaction. How? For a start, the tableau system introduced in [Buchheit *et al.*, 1993] for $\mathcal{ALC}$ provides a decision method to prove consistency of knowledge bases in $\mathcal{FI}$; properties from the basic system (e.g., no user can establish a path with herself) can be formally written in $\mathcal{FI}$ and automatically checked.

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 $\mathcal{FI}$, since the syntactic restrictions imposed can be overcome by means of definitions. The following complexity result for $\mathcal{ALCNR}$ (the extension of $\mathcal{ALC}$ with number restrictions and role conjunctions) with cyclic terminologies and assertions is provided in [De Giacomo *et al.*, 1996].

Theorem 2 *Given an $\mathcal{ALCNR}$ knowledge base K, checking whether K is satisfiable can be done in EXPTIME.*

$$
\begin{array}{rcl}
\text{Actions}_{uv}(\mathit{idle\_u}) & = & \{\mathit{offhook\_u}\} \cup \{\mathit{dial\_v\_u} \mid v \in \mathsf{SUBS}, v \neq u\} \\
\text{Actions}_{uv}(\mathit{dialtone\_u}) & = & \{\mathit{onhook\_u}\} \cup \{\mathit{dial\_u\_v} \mid v \in \mathsf{SUBS}, v \neq u\} \\
\text{Actions}_{uv}(\mathit{busytone\_u}) & = & \{\mathit{onhook\_u}\} \\
\text{Actions}_{uv}(\mathit{ringing\_u}) & = & \{\mathit{offhook\_u}\} \\
\text{Actions}_{uv}(\mathit{ringing\_u} \sqcap \mathit{calling\_v\_u}) & = & \{\mathit{onhook\_v}\} \\
\text{Actions}_{uv}(\mathit{ringback\_u}) & = & \{\mathit{onhook\_u}\} \\
\text{Actions}_{uv}(\mathit{ringback\_u} \sqcap \mathit{calling\_u\_v}) & = & \{\mathit{offhook\_v}\} \\
\text{Actions}_{uv}(\mathit{engaged\_u}) & = & \{\mathit{onhook\_u}\} \\
\text{Actions}_{uv}(\mathit{engaged\_u} \sqcap \mathit{path\_v\_u}) & = & \{\mathit{onhook\_v}\} \\
\text{Actions}_{uv}(\mathit{ready\_u}) & = & \{\mathit{onhook\_u}\} \cup \{\mathit{dial\_u\_v} \mid v \in \mathsf{SUBS}, v \neq u\} \\
\text{Actions}_{uv}(\mathit{calling\_u\_v}) & = & \{\mathit{onhook\_u}, \mathit{offhook\_v}\} \\
\text{Actions}_{uv}(\mathit{rejecting\_u}) & = & \{\mathit{onhook\_u}\} \\
\text{Actions}_{uv}(\mathit{path\_u\_v}) & = & \{\mathit{onhook\_u}, \mathit{onhook\_v}\}
\end{array}
$$

Figure 1: Allowed actions in a given state.

What does this imply for the complexity of determining feature interaction? Below, we will define this task as a special satisfiability problem, and, hence, from the theorem it follows that determining feature interaction (as it is modeled here) is decidable in EXPTIME.

Features. 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, $\delta(S) \not\models \varphi$. 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 (Refinement and Interaction) For a given feature F we will define a refinement operation δ_F, and a set of new concepts $\mathsf{ACT}_F \subseteq \{F_{\bar{u}} \mid \bar{u} \in \mathsf{SUBS}^+\}$ (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 $N \subseteq \mathsf{ACT}_F$ if $\delta_F(K) \cup N^c \cup N^d \models \bot$, 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* $N^d = \{\top \sqsubseteq \neg F_{\bar{u}} \mid F_{\bar{u}} \in \mathsf{ACT}_F \backslash N\}$.

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 $N_{F_1} \subseteq \mathsf{ACT}_{F_1}$ and $N_{F_2} \subseteq \mathsf{ACT}_{F_2}$ such that BCS and $F_1 \circ F_2$ interact on activation $N_{F_1} \cup N_{F_2}$.

We will now indicate how to extend the BCS with features via refinements. As an example we will add two standard features (TCS and CFU) and indicate how their interaction can be detected.

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_u_v, and simply refining BCS by adding $TCS\_u\_v \sqsubseteq \neg \mathit{calling\_v\_u}$. However, this extension immediately interacts (with itself on activation $\{TCS\_u\_v\}$). 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 $\{TCS\_u\_v \mid u, v \in \mathsf{SUBS}, u \neq v\}$. The refinement δ_{TCS} is defined by replacing, in any knowledge base, terminologies of the form $\mathbf{C}_1[\mathit{ready\_u} \sqcap \mathit{idle\_v}] \sqsubseteq \exists \mathit{dial\_u\_v}.\mathbf{C}_2[\mathit{calling\_u\_v}]$ by

$$
\begin{aligned}
\mathbf{C}_1[\neg\, TCS\_v\_u \sqcap ready\_u \sqcap idle\_v] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_2[calling\_u\_v], \\
\mathbf{C}_1[TCS\_v\_u \sqcap ready\_u \sqcap idle\_v] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_2[rejecting\_u], \\
TCS\_v\_u &\sqsubseteq \neg calling\_u\_v.
\end{aligned}
$$

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_u_v is that whenever w calls u, she will be connected to v instead. Take the set $\{CFU\_u\_v \mid u, v \in \mathsf{SUBS}, u \neq v\}$ as activation concepts. Define δ_{CFU} as replacing axioms of the form $\mathbf{C}_1[ready\_u \sqcap idle\_v] \sqsubseteq \exists dial\_u\_v.\mathbf{C}_2[calling\_u\_v]$ and $\mathbf{C}_3[ready\_u \sqcap \neg idle\_v] \sqsubseteq \exists dial\_u\_v.\mathbf{C}_4[rejecting\_u]$ by

$$
\begin{aligned}
\mathbf{C}_1[\neg \textstyle\bigsqcup_{w\in\mathsf{SUBS}} CFU\_v\_w \sqcap ready\_u \sqcap idle\_v] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_2[calling\_u\_v], \\
\mathbf{C}_1[CFU\_v\_w \sqcap ready\_u \sqcap idle\_w] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_2[calling\_u\_w], \\
\mathbf{C}_3[\neg \textstyle\bigsqcup_{w\in\mathsf{SUBS}} CFU\_v\_w \sqcap ready\_u \sqcap \neg idle\_v] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_4[rejecting\_u], \\
\mathbf{C}_3[CFU\_v\_w \sqcap ready\_u \sqcap \neg idle\_w] &\sqsubseteq \\
&\exists dial\_u\_v.\mathbf{C}_4[rejecting\_u].
\end{aligned}
$$

Some interesting issues surfaced when formally defining this feature. Notice that the forwarding event is implemented by changing the state *calling_u_v* to *calling_u_w* whenever CFU_v_w is active (together with some book keeping). But this bypasses the *dial_u_v* action. If, in addition, CFU_w_x 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.

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 $\{TCS\_C\_A\} \cup \{CFU\_B\_C\}$.

In similar ways, other reasoning tasks performed in [Gammelgaard and Kristensen, 1994] can be accommodated.

3 Conclusion

In this paper we have provided a formal analysis of the method for feature interaction detection first introduced in [Gammelgaard and Kristensen, 1994]. We have proposed a description logic which tightly fits the informal specification and arguments used in that paper to study the feature interaction problem. The modeling enterprise undertaken in this paper has displayed various inaccuracies and ambiguities in [Gammelgaard and Kristensen, 1994]. 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. For example, it seems interesting to further investigate the nature of the refinement operator, and to explore links with the general literature on refinements. As to interaction issues, we have yet to devise ways of modeling features that affect the network (such as Call Waiting), and the issue of good vs. bad interaction deserves more attention.

Acknowledgments: Maarten de Rijke is supported by the Spinoza Project 'Logic in Action.'

References

[Bouma, 1999] W. Bouma. Feature interactions. In *Encyclopedia of Computer Science and Technology*. Marcel Dekker, Inc, 1999.

[Bowen *et al.*, July 1989] T. Bowen, F. Dworack, C. Chow, N. Griffeth, and Y. Lin. The feature interaction problem in telecommunication systems. In *Proc. 7th. Int. Conf. on Soft. Eng. for Telecommunication Switching Systems*, pages 59–62, 1989.

[Buchheit *et al.*, 1993] M. Buchheit, F. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[De Giacomo *et al.*, 1996] G. De Giacomo, F. Donini, and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. In *Proc. of DL'96*, pages 107–110, 1996.

[Gammelgaard and Kristensen, 1994] A. Gammelgaard and J. Kristensen. Interaction detection, a logical approach. In L. Bouma and H. Velthuijsen, editors, *Feature Interactions in Telecommunication Systems*, pages 178–196. IOS Press, 1994.

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