SECRETS OF CALL FORWARDING: A SPECIFICATION CASE STUDY

Pamela Zave

AT&T Bell Laboratories
Room 2B-413
Murray Hill, New Jersey 07974, USA
+1 908 582 3080
`pamela@research.att.com`

18 April 1995

Abstract: Call forwarding and features related to it have remained mysterious despite the efforts of many telecommunications experts to understand them. This paper shows how an analysis of the human situations supported by telecommunications features can clarify what feature behavior is desirable, even in subtle and apparently ambiguous situations. The paper also shows how a formal specification of the routing view of a switching system, written in a tabular form of Z, can provide a concise yet comprehensive description of forwarding features—including their interactions with each other and with blocking features. This view-oriented approach to formal specification of telecommunications systems has many practical advantages.

Keywords: telecommunications, feature interactions, formal specifications and methods, views, the Z specification language.

Secrets of call forwarding: A specification case study

Pamela Zave
AT&T Bell Laboratories
Murray Hill, New Jersey 07974, USA, pamela@research.att.com

Abstract

Call forwarding and features related to it have remained mysterious despite the efforts of many telecommunications experts to understand them. This paper shows how an analysis of the human situations supported by telecommunications features can clarify what feature behavior is desirable, even in subtle and apparently ambiguous situations. The paper also shows how a formal specification of the routing view of a switching system, written in a tabular form of Z, can provide a concise yet comprehensive description of forwarding features—including their interactions with each other and with blocking features. This view-oriented approach to formal specification of telecommunications systems has many practical advantages.

Keywords

Telecommunications, feature interactions, formal specifications and methods, views, tables, the Z specification language

1 THE CHALLENGE OF CALL FORWARDING

Feature interactions are a severe problem in telecommunications systems, and are now receiving a great deal of attention from researchers [IWFI 92, Bouma and Velthuijsen 94]. Despite progress in applying formal methods to the specification of features and to the detection, resolution, and avoidance of feature interactions, many difficulties remain. Two of these difficulties are illustrated by call forwarding.

The first difficulty is deciding how features *should* behave in the presence of others. Let's look at some simple examples concerning call forwarding:

Example 1: Directory number (DN) d_1 is forwarded to DN d_2, and d_2 is forwarded to d_3. Should a call to d_1 be routed to d_2 or d_3?

Example 2: d_1 is forwarded to d_2. What happens to calls to d_1 when Do Not Disturb (DND) applies to d_1 and not to d_2? When DND applies to d_2 and not to d_1?

Example 3: The subscriber of d_1 subscribes to Originating Call Screening (OCS, also known as Outgoing Call Barring and Call Restriction) and has used it to block calls to d_2. His child's friend at d_3 sets up forwarding from d_3 to d_2, and his child calls d_3. Should the call go through?

Example 4: The subscriber of d_1 subscribes to OCS and has used it to block calls to d_2. Calls from d_2 are forwarded to d_3. Should a call from d_1 to d_2 go through?

These questions (and many others) about the desirable behavior of call forwarding are well known, but they have not yet been answered. Similar questions appear as open problems in survey papers on feature interaction [Cameron et al. 93, Muller 94]. Usually there are excellent arguments on both sides of these questions, which leads me to believe that there are no right answers to questions posed in this way.

In the absence of consensus on ‘‘correct’’ answers, three other approaches to determining the behavior of forwarding features have been explored, all with apparent flaws. (1) The open questions about forwarding behavior can be left open as options, and subscribers can make their own choices [Muller 94]. It seems unreasonable to ask subscribers to understand and answer questions that telecommunications experts cannot. (2) The behavior can be determined by negotiation at the time of call setup [Griffeth and Velthuijsen 92]. If the negotiating agent is the subscriber, this would seem to add greatly to setup overhead and to the intellectual burden on the subscriber. If the negotiating agent is a program, then this proposal is just a new way of implementing features, and brings no new insight to the question of how features should behave. (3) The behavior can be determined by arbitrary and subjective choices. This is intellectually unsatisfying. Also, like the other two approaches, it tends to complicate system behavior and implementation.

The second difficulty is fitting the specifications of all features (and desired feature interactions) into a single, monolithic specification framework. The complexity and diversity of telecommunications will defeat such efforts in the forseeable future.

For example, the most widely known approach to the feature-interaction problem divides call processing into phases, in each of which at most one feature can be active. Typically the active feature is chosen from a priority list, and is the highest-priority feature that applies to the situation.

A variation on this approach organizes competing features by means of blocking tables [Schessel 92]. A blocking table makes it possible to say of a pair of features that when the first feature is active the second feature cannot be active. There are many related policies based on this general idea [Homayoon and Singh 88].

It should be obvious that none of these approaches is capable of specifying call forwarding and its related features. Even at their most sophisticated and complete, these approaches can do nothing more than specify that features $X_1, X_2, \ldots$ are active and features $Y_1, Y_2, \ldots$ are not active in some situation. They cannot handle multiple applications of the same feature in the same situation (as in multiple forwarding steps), and they cannot manage any coordination of active features (as when screening is applied to a forwarding DN rather than to the dialed DN).

For an example on the opposite side of the argument, a recent specification of a switching system with twelve popular features [Mataga and Zave 95] uses languages in three major families (finite-state automata, grammars, and set theory) to specify three major aspects of the system (the device interface, digit analysis, and connections, respectively). As a result of the multiparadigm approach used, the specification is relatively concise and readable, despite the fact that the switch specified is many times more complex than others specified in the literature [Boumezbeur and Logrippo 93, Kay and Reed 93, Woodcock and Loomes 88].

This paper is a specification case study. It shows how the two difficulties described above can be overcome, at least in the case of call forwarding. The principles illustrated are quite general, and can be applied to other groups of features.

The difficulty of deciding how features should behave is overcome by the following principle: a feature should fit and address a recognizable human situation, and be described in terms of its roles. This rule makes features easy to document and market. Also, the human situation acts as a guideline for choosing useful and predictable feature behavior. It is not necessary to involve subscribers in the details of feature interaction, and yet features behave as the subscribers expect, because both designers and subscribers are thinking in terms of the same human situations. This is illustrated in Section 2, with a situational analysis of forwarding features and blocking features (with which forwarding features interact closely).

The difficulty of achieving success with monolithic specifications is overcome by adopting a view-oriented approach. For each group of features there is an appropriate abstraction, capturing everything relevant to that group of features. The abstraction is formalized in terms of a suitable notation, and becomes a partial specification of the system. A routing view—which includes all forwarding and blocking features—is specified in Sections 3 through 5.

A view specification can, because of its specialization, be extremely powerful. It can specify completely the behavior and interactions of the features that belong to it. It can handle feature sets of realistic complexity, and it can be designed for extensibility to new features in the group.

In the short term, these advantages outweigh the disadvantage that interactions between views are not represented formally. In the long term, we must work toward integration and composition of the views. This subject is discussed in Section 6.

The presentation in this paper relies on three simplifying assumptions, all easily removed for a more realistic treatment of the subject: (1) I assume a homogeneous set of DNs. In more realistic treatments, distinctions can be made between local and toll DNs, etc. (2) I assume that the only interesting outcome of routing is a final destination for the call. In some real situations, each forwarding step results in a separately billable segment of the call. (3) I assume that calls can be regarded as point-to-point during all stages of setup. This does not preclude conferencing features, but it does exclude group setup features such as hunt groups and shared DNs.

As a result of these assumptions, each DN maps onto a unique telephone line, which I shall refer to as a unique telephone for brevity. Each DN also maps onto a unique subscriber who pays the bill for that DN and determines which features apply to it. Thus ‘‘the telephone of a DN,’’ ‘‘the subscriber of a DN,’’ and ‘‘the features applying to a DN’’ are always well-defined terms.

2 A SITUATIONAL ANALYSIS OF FORWARDING AND BLOCKING FEATURES

Call forwarding is chronically confusing because the name covers two distinct human situations. In the *follow me* situation, the forwarder expects to be in an unusual place, and wants his calls to follow him to that place. In the *delegate* situation, the forwarder expects to be unavailable, and wants to delegate to another person (secretary, colleague working on the same project) or surrogate (message service) the responsibility of answering his calls. In one situation the subscriber expects to be answering his own calls, while in the other situation he expects someone or something else to be answering them.

Distinguishing these two human situations is the key to understanding call forwarding and solving the puzzles concerning how it should behave. Consider Example 1 in Section 1. If the forwarding of d_1 is delegate forwarding, then the call should be routed to d_3. This is because the call has been delegated to the subscriber of d_2, so any forwarding requested from d_2 applies. If, on the other hand, the forwarding of d_1 is follow-me forwarding, then the call should be routed to d_2. The subscriber of d_1 expects to be located near the telephone addressed by DN d_2, and wants his calls to be routed to that telephone. It is likely that the subscriber of d_2 is also somewhere unusual—especially since someone else is using his telephone—and may be forwarding his calls, but the wishes of the subscriber of d_2 have absolutely nothing to do with the subscriber of d_1 in this case.

Here is another way to remember the distinction between the two human situations. I have already pointed out that each DN maps onto a unique telephone and a unique subscriber. In follow-me forwarding, the forwarded-to DN is being used as a key or pointer to a *telephone*. In delegate forwarding, the forwarded-to DN is being used as a key or pointer to a *subscriber*.

The first step in obtaining a satisfactory specification of call forwarding is to clarify which human situations are being addressed by which features. The three standard features are ambiguous in this respect, so I shall specify five new features instead. Table 1 shows the correspondences between standard and new features. Note that there is no Follow Me Busy feature, because it would make no sense in human terms. If the subscriber's telephone is busy, then the subscriber is in his usual place, using his telephone, and there is nowhere to follow him.

standard feature	**new feature**
Call Forwarding Unconditional	Follow Me Delegate
Call Forwarding Busy	Delegate Busy
Call Forwarding No Answer	Follow Me No Answer Delegate No Answer

Table 1

Blocking features are routing features that prevent the completion of calls when they are considered undesirable by one of the people involved. Table 2 shows a representative sample of blocking features, all of which will be specified formally as part of the routing view.

standard feature	**new feature**	**feature description**
Do Not Disturb	Do Not Ring	do not alert this telephone
	Vacation Protection	do not connect calls for this subscriber
Originating Call Screening		block outgoing calls to certain DNs
Terminating Call Screening		block incoming calls from certain DNs
Calling Number Delivery Blocking		refuse to allow caller identification on outgoing call
Anonymous Call Rejection		block incoming calls having no caller identification

Table 2

Like the forwarding features, DND may benefit from a distinction between subscriber and telephone. It will be treated as two new features. Do Not Ring (DNR) prevents the routing of calls to the telephone of the DN, so that no one in the vicinity of the telephone is disturbed or interrupted. Vacation Protection (VP) prevents the routing of calls to the subscriber of the DN, so that a person can go on vacation without returning to a huge accumulation of messages. At the same time, calls for his housesitter, routed to his telephone as a result of follow-me forwarding from his housesitter's DN, will go through.

The questions in Example 2, Section 1, split into a number of cases depending on what kind of forwarding is being used and which meaning of DND is desired: (1) If DNR applies to d_1, it has no effect. (2) If DNR applies to d_2, the call does not go through. (3) If VP applies to d_1 and delegate forwarding is used, then the call goes through, because it is being routed to the subscriber of d_2. (4) If VP applies to d_1 and follow-me forwarding is used, then the call does not go through, because it is being routed to the subscriber of d_1. (5) If VP applies to d_2 and follow-me forwarding is used, then the call goes through, because it is being routed to the subscriber of d_1. (6) If VP applies to d_2 and delegate forwarding is used, then the call does not go through, because it is being routed to the subscriber of d_2.

In contrast, the subscriber/telephone distinction appears to have little relevance to OCS and Terminating Call Screening (TCS). The distrustful intent of these features is so strong, I believe, that the call should be blocked if the screened DN appears anywhere in its routing path. Thus the calls in both Examples 3 and 4 should be blocked.

There is another way of looking at OCS. We could ask whether the purpose of blocking is keep children from talking to distrusted parties, or to keep children from running up large phone bills (for example, by calling 900 numbers). If the intended purpose is financial, then the call proposed in Example 3 would seem to be acceptable, because the extraordinary fees will be billed to the child's friend's parents. However, I do not think that this alternative OCS behavior is desired by responsible parents.

Anonymous call rejection (ACR) is closely associated with Calling Number Delivery (CND), and both are closely associated with a particular telephone (it makes no sense to subscribe to CND if you do not own a telephone capable of displaying the caller identification, and it makes no sense to subscribe to ACR if you do not subscribe to CND). Thus I interpret these features as applying strictly to the telephone pointed to by the DN, rather than as applying to the subscriber of the DN as portable communication privileges.

As we have seen, situational analysis frequently results in additional new features. This may be regarded as an advantage (more features to sell!) or a disadvantage. If it is regarded as a disadvantage, then the problems raised are easily overcome.

Suppose, for instance, that a switch provider is committed to supplying three forwarding features under the standard names. Then the provider can assign to each of the names on the left of Table 1 one of the features on the right of Table 1. In fact, this is the only way to make the three standard features unambiguous.

Even if the switch provider is committed to supplying three forwarding features under the standard names, he may find it beneficial to secretly implement all five. The routing specification in Sections 4 and 5 makes it trivial to do so. Then, when some telephone company requests a special version of call forwarding, it is likely to be implemented already.

3 A CONTEXT FOR THE ROUTING VIEW

Most descriptions of call processing are heavily temporal in nature, emphasizing states, events, and state transitions. Figure 1 is a typical specification fragment. It is a finite-state automaton giving a partial description of call processing; it shows some of the states, external events, and state transitions required.

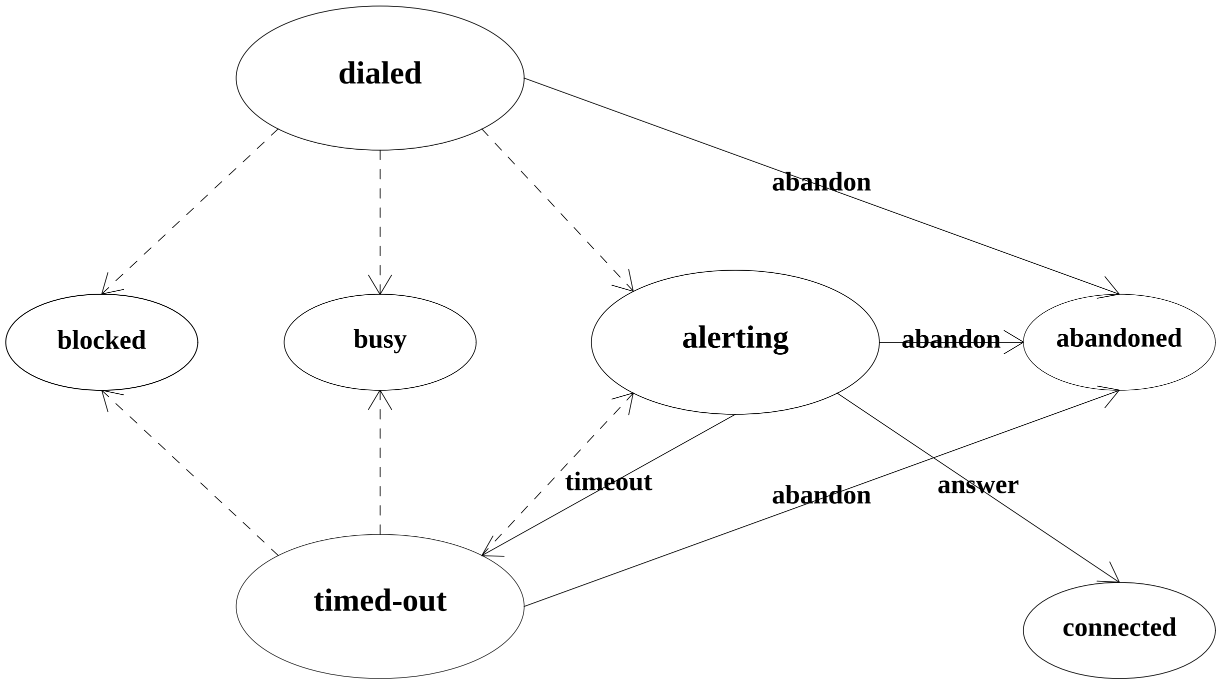

Figure 1 A partial description of call processing.

Routing is an atypical aspect of call processing in that most of its complexity is nontemporal—it concerns static data and relationships. It is easier to understand routing features if their specification is separated from the temporal aspects of call processing and uses a syntax that is appropriate for managing static complexity. Accordingly, Sections 4 and 5 contain a formal specification of routing, expressed in a tabular form of the specification language Z [Spivey 92].

The specification of call processing in this section and the specification of routing in Sections 4 and 5 are both partial—neither one is complete in itself. The formal composition of these and other views will be discussed in Section 6. For now, an informal description of how the views fit together should be sufficient.

Although Figure 1 is incomplete in many respects, it does include all out-transitions from the three large states. The solid arrows represent state transitions stimulated by external events, while the dashed arrows represent state transitions stimulated by computation in a switching system.

There are also four relevant state variables, all of type DN: *origin, dialed, tel,* and *subs*. The DN of the caller is saved in *origin,* and the dialed DN is saved in *dialed*. Whenever the value of *tel* is defined, it is the DN whose telephone is the current target of routing. Whenever the value of *subs* is defined, it is the DN whose subscriber is the current target of routing.

Section 4 defines two functions declared as:

$$\textit{dialing-outcome: DN} \rightarrow (\textit{call-state} \times \textit{DN} \times \textit{DN})$$
$$\textit{timeout-outcome: } (\textit{DN} \times \textit{DN}) \rightarrow (\textit{call-state} \times \textit{DN} \times \textit{DN})$$

Both of these are total functions from their arguments to value triples. The last two components of each value triple must be DNs and the first component must be a call state, where *call-state* denotes the enumerated type:

$$\textit{call-state} ::= \textit{blocked} \mid \textit{busy} \mid \textit{alerting} \mid \textit{quick-alerting}$$

These functions are used to control what the system does in the *dialed, alerting,* and *timed-out* states, as follows.

In dialed state: Execute this pseudocode.

```
(new-state,tel,subs) := dialing-outcome(dialed);
if "abandon event has occurred" then "next state is abandoned"
   else if (new-state = blocked) then "next state is blocked"
   else if (new-state = busy) then "next state is busy"
   else if (new-state = alerting)
      then "next state is alerting"
   else if (new-state = quick-alerting)
      then "next state is alerting and set alerting timer".
```

In alerting state: Alert the telephone whose DN is in *tel*. Transition to the next state is strictly based on external events, as shown in Figure 1.

In timed-out state: Execute this pseudocode.

```
(new-state,tel,subs) := timeout-outcome(tel,subs);
if "abandon event has occurred" then "next state is abandoned"
   else if (new-state = blocked) then "next state is blocked"
   else if (new-state = busy) then "next state is busy"
   else if (new-state = alerting)
      then "next state is alerting"
   else if (new-state = quick-alerting)
      then "next state is alerting and set alerting timer".
```

4 FIRST VERSION OF THE ROUTING VIEW: FORWARDING FEATURES

The state of the system must record which features apply to which DNs. For Plain Old Telephone Service (POTS) and the call forwarding features this information is captured in Z as follows:

$[DN, telephone]$
$POTS: DN \rightarrow telephone$
$Del: DN \nrightarrow DN$
$FM: DN \nrightarrow DN$
$DelB: DN \nrightarrow DN$
$DelNA: DN \nrightarrow DN$
$FMNA: DN \nrightarrow DN$

The first line says that *DN* and *telephone* are primitive types of the specification. *POTS*, *Del*, *FM*, *DelB*, *DelNA*, and *FMNA* are all binary relations, or, in other words, sets of ordered pairs. The two different arrow symbols denote different constraints on these sets: a plain arrow is a total function, and a crossed arrow is a partial function. **dom** *R* denotes the domain of relation *R*. Thus $(d_1, d_2) \in R$ if and only if the ordered pair (d_1, d_2) is a member of the relation *R*, and its membership implies that $d_1 \in$ **dom** *R*.

If (d_1, d_2) is a member of some forwarding relation R_f, then d_1 is forwarded to d_2 by means of the feature corresponding to R_f. The fact that R_f is a function means that a DN d_1 can only be forwarded to one DN at a time by means of that feature. The fact that R_f is a partial function means that a DN d_1 need not be forwarded; only those DNs in the domain of R_f are being forwarded by means of that feature. As an extreme case of this, if a forwarding feature is not being offered to subscribers, then its corresponding relation is always empty.

The content of these relations changes over time, as a result of provisioning operations and control commands from telephones. There are only two restrictions on the updating of these relations: (1) Queries and updates must be synchronized in the usual way, so that they appear atomic and are actually serializable. (2) Integrity constraints on the relations must be preserved at all times. For example, the functional nature of functional relations must be preserved.

Some integrity constraints keep the features from being used in nonsensical ways. For example, the following constraint prevents a subscriber from forwarding calls to himself. The first line defines an aggregate relation, while the second line says that it must contain no pairs from the identity relation on DNs.

$$\textit{All-Forwards} == \textit{Del} \cup \textit{FM} \cup \textit{DelB} \cup \textit{DelNA} \cup \textit{FMNA}$$
$$\textit{All-Forwards} \cap \mathbf{id}\ \textit{DN} = \emptyset$$

These constraints prevent a subscriber from requesting two different treatments of the same situation:

$$\mathbf{dom}\ \textit{Del} \cap \mathbf{dom}\ \textit{FM} = \emptyset$$
$$\mathbf{dom}\ \textit{DelNA} \cap \mathbf{dom}\ \textit{FMNA} = \emptyset$$

This section contains the definitions of four functions. *dialing-outcome* and *timeout-outcome* were declared in Section 3. The other two functions are:

$$\textit{subscriber-outcome: DN} \rightarrow (\textit{call-state} \times \textit{DN} \times \textit{DN})$$
$$\textit{telephone-outcome:} (\textit{DN} \times \textit{DN}) \rightarrow (\textit{call-state} \times \textit{DN} \times \textit{DN})$$

The definitions of *dialing-outcome* and *timeout-outcome* use *subscriber-outcome* and *telephone-outcome*. The definitions of *subscriber-outcome* and *telephone-outcome* use themselves and each other in a highly recursive manner. The definition of *dialing-outcome* is simply:

$$\textit{dialing-outcome(dialed)} = \textit{subscriber-outcome(dialed)}$$

The definitions of the other three functions have several cases. Case enumeration is common in specifications and often difficult to read, so I have invented a simple tabular notation to make these definitions (in Tables 3 through 5) more readable. The tables could be translated automatically into Z, and so can be regarded as a shorthand.

The top row of each table supplies names for all the formal parameters of the function, both incoming and outgoing. Each subsequent row of the table covers another numbered case, the second column giving the condition under which the case applies, and the third column giving the value of the function in that case. The cases in each table are ordered from top to bottom. Thus each condition must be understood as including, implicitly, the negations of all previous conditions in the table.

case	**of . . .** *subscriber-outcome(old-subs) = (new-state,new-tel,new-subs)*	
1	*old-subs* ∈ **dom** *Del*	*subscriber-outcome(Del(old-subs))*
2	*old-subs* ∈ **dom** *FM*	*telephone-outcome(FM(old-subs),old-subs)*
3	*true*	*telephone-outcome(old-subs,old-subs)*

Table 3

case	**of . . .** *telephone-outcome(old-tel,old-subs) = (new-state,new-tel,new-subs)*	
1	*POTS(old-tel)* ∈ *Busy* ∧ *old-subs* ∈ **dom** *DelB*	*subscriber-outcome(DelB(old-subs))*
2	*POTS(old-tel)* ∈ *Busy*	*(busy,old-tel,old-subs)*
3	*old-subs* ∈ **dom** *DelNA*	*(quick-alerting,old-tel,old-subs)*
4	*old-subs* ∈ **dom** *FMNA* ∧ *old-tel = old-subs*	*(quick-alerting,old-tel,old-subs)*
5	*true*	*(alerting,old-tel,old-subs)*

Table 4

case	**of . . .** *timeout-outcome(old-tel,old-subs) = (new-state,new-tel,new-subs)*	
1	*old-subs* ∈ **dom** *DelNA*	*subscriber-outcome(DelNA(old-subs))*
2	*old-subs* ∈ **dom** *FMNA* ∧ *old-tel = old-subs*	*telephone-outcome(FMNA(old-subs),old-subs)*
3	*true*	*(alerting,old-tel,old-subs)*

Table 5

This line of Z means that *Busy* is a set of telephones:

Busy: $\mathbb{P}$ *telephone*

At any time, the set *Busy* contains all the telephones that are currently busy. (This is typical of the Z specification style.)

As you might expect, recursive evaluations of these functions correspond to forwarding steps, and the argument substitutions have the effect of updating the routing destination at each step. It is important to note, however, that the function definitions maintain two separate routing destinations, the telephone being routed to and the subscriber being routed to. Follow-me forwarding distinguishes these two, so manipulating them separately is the only way to do follow-me forwarding correctly.

Every time a version of delegate forwarding is applied, the function *subscriber-outcome* is evaluated with the DN of the newly delegated subscriber as its argument. This function checks for further unconditional delegate or follow-me forwarding.

Whenever routing reaches a phase in which the status of a particular telephone is relevant, the function *telephone-outcome* is evaluated with the current telephone destination and subscriber destination as its arguments. This function checks the telephone for unavailability. If available then the telephone will be alerted; if no-answer forwarding could apply then the alerting must be quick.

timeout-outcome is only evaluated after quick alerting and a timeout. Because of this history, it would be reasonable to assume that Cases 1 and 2 of Table 5 are exhaustive. However, it is important to realize that the same atomic queries of the relations *DelNA* and *FMNA* before and after quick alerting might conceivably yield different results, because the relations might have changed during that interval of time! This situation will be rare, but formal specifications cannot ignore any possibilities. Case 3 in Table 5 takes care of it.

An additional integrity constraint on the forwarding relations is motivated by a desired routing property, namely guaranteed termination. This constraint:

$\mathit{All\text{-}Delegates} == \mathit{Del} \cup \mathit{DelB} \cup \mathit{DelNA}$
$\mathit{All\text{-}Delegates}^{+} \cap \mathbf{id}\ \mathit{DN} = \varnothing$

says that it is impossible, beginning with a DN and transforming it successively by means of delegation steps, to get back to the original DN. The appendix contains a proof that if the delegate relations do not change during routing and if this integrity constraint on delegate relations holds, then routing is guaranteed to terminate.

This proof is important primarily because it validates my beliefs about the specification, how it works, and why it captures the correct way to do call forwarding (the follow-me relations need not be checked for forwarding loops because follow-me forwarding cannot be applied recursively). As a practical means of preventing forwarding loops in telecommunications systems, however, this result may be inadequate because maintaining the constraint on delegate relations may be too expensive, or because the actual updating of delegate relations concurrently with call routing makes it too unreliable. If so, developers can rely on older strategies such as fixed limits on the number of forwarding steps.

Two other notations were tried on this specification and rejected. Originally it seemed that the use of relational algebra would yield a better (more concise and more declarative) specification than recursive function definitions. Using relational algebra, the specification of call forwarding would be a relation R defined by applying various relational operators to the forwarding relations. R would be such that $(d_1, d_2) \in R$ if and only if calls dialed to d_1 should be routed to d_2. This approach fails because the outcome of routing is a three-tuple rather than a single DN. Although a binary relation on three-tuples is conceivable, it would be awkward to manipulate and no improvement in readability over the tables.

The tabular notation was inspired by the use of tables in a method for program documentation [Parnas et al. 94]. Interestingly, these tables work poorly on the routing specification for much the same reason that relational algebra does. The routing specification is a function defined by carrying two interdependent variables through recursive function evaluations. Analysis of each case leads to determination of three closely interdependent output variables. The program-documentation tables specify each output variable separately, so that the close associations among the variable values cannot be exploited.

5 SECOND VERSION OF THE ROUTING VIEW: BLOCKING FEATURES

Like forwarding features, each blocking feature has a relation recording which DNs it applies to. In Z these are declared as follows:

$\mathit{DNR}: \mathbb{P}\ \mathit{DN}$
$\mathit{VP}: \mathbb{P}\ \mathit{DN}$
$\mathit{OCS}: \mathit{DN} \leftrightarrow \mathit{DN}$
$\mathit{TCS}: \mathit{DN} \leftrightarrow \mathit{DN}$
$\mathit{CNDB}: \mathbb{P}\ \mathit{DN}$
$\mathit{ACR}: \mathbb{P}\ \mathit{DN}$

A double-headed arrow indicates an unconstrained binary relation. If (d_1, d_2) is a member of some screening relation R_s, then calls from d_1 to d_2 are prohibited by means of the feature corresponding to R_s.

DNR, VP, CNDB, and *ACR* are all unary relations, or, in other words, sets of DNs. If a DN *d* is a member of one of these sets, then the corresponding feature applies to *d.*

The formal specification of blocking relies on three aggregate relations, defined as follows:

Block-Telephone == *(DN* $\times$ *DNR)* $\cup$ *(CNDB* $\times$ *ACR)*
Block-Subscriber == *(DN* $\times$ *VP)*
Block-Route-Through == *OCS* $\cup$ *TCS*

(DN $\times$ *DNR)* is a binary relation containing all ordered pairs of DNs in which the first element belongs to *DN* and the second element belongs to *DNR.* *(CNDB* $\times$ *ACR)* is a binary relation containing all ordered pairs of DNs in which the first element belongs to *CNDB* and the second element belongs to *ACR.*

Block-Telephone contains all pairs (d_1, d_2) such that a call from d_1 should be blocked if it is routed to the telephone of d_2. *Block-Subscriber* contains all pairs (d_1, d_2) such that a call from d_1 should be blocked if it is routed to the subscriber of d_2. *Block-Route-Through* contains all pairs (d_1, d_2) such that a call from d_1 should be blocked if the telephone or subscriber of d_2 appears anywhere in its routing path. The explanations in Section 2 should make it clear why these aggregate relations are defined as they are.

To use these new relations to block undesired calls, we simply add appropriate lookups to the routing view. Specifically, Tables 3 and 4 are replaced by Tables 6 and 7.

case	**of . . .** *subscriber-outcome(old-subs) = (new-state,new-tel,new-subs)*	
1	*(origin,old-subs)* ∈ *Block-Route-Through*	*(blocked,old-subs,old-subs)*
2	*old-subs* ∈ **dom** *Del*	*subscriber-outcome(Del(old-subs))*
3	*(origin,old-subs)* ∈ *Block-Subscriber*	*(blocked,old-subs,old-subs)*
4	*old-subs* ∈ **dom** *FM*	*telephone-outcome(FM(old-subs),old-subs)*
5	*true*	*telephone-outcome(old-subs,old-subs)*

Table 6

The key to this treatment of blocking is obviously the *tel* and *subs* DNs, which are updated at every routing step and always indicate what is filling the roles of destination telephone and destination subscriber, respectively. They provide the exact information needed to specify blocking features that are, like the forwarding features, tailored precisely to meet human needs.

The use of aggregate blocking relations makes this specification more extensible than it would be if the tables used the feature relations directly. When a new blocking feature is proposed, there is a good chance that it can be added by modifying the aggregate relations alone, without touching the (more complex) tables.

case	of . . . *telephone-outcome(old-tel,old-subs) = (new-state,new-tel,new-subs)*	
1	*(origin,old-tel)* $\in$ *Block-Route-Through*	*(blocked,old-tel,old-subs)*
2	*POTS(old-tel)* $\in$ *Busy* $\wedge$ *old-subs* $\in$ **dom** *DelB*	*subscriber-outcome(DelB(old-subs))*
3	*(origin,old-tel)* $\in$ *Block-Telephone*	*(blocked,old-tel,old-subs)*
4	*POTS(old-tel)* $\in$ *Busy*	*(busy,old-tel,old-subs)*
5	*old-subs* $\in$ **dom** *DelNA*	*(quick-alerting,old-tel,old-subs)*
6	*old-subs* $\in$ **dom** *FMNA* $\wedge$ *old-tel = old-subs*	*(quick-alerting,old-tel,old-subs)*
7	*true*	*(alerting,old-tel,old-subs)*

Table 7

6 VIEW INTEGRATION

The integration of the routing view with other views has not been formalized, but it is easy to describe informally. Evaluations of the routing functions are embedded in the transitions of a finite-state automata for call processing. The feature relations are used by the routing view and updated by whichever view does provisioning and management of subscriber data. The variables *tel* and *subs* are updated by this view, and can be used by feature specifications that need to know which DNs are playing the roles of destination telephone and destination subscriber.

For example, in the case of a collect call, the destination subscriber—not destination telephone—should be billed. If a subscriber receives a call at someone else’s telephone by means of follow-me forwarding, and accepts the charges, it seems only fair to bill the responsible subscriber rather than his helpless host. In the case of automatic retry (of a call to a busy telephone), the destination telephone should be monitored and retried, because that is the telephone whose busy state invoked the feature.

Techniques exist for formalizing the composition semantics of a wide variety of specification languages [Zave and Jackson 93]. These techniques might be sufficient to formalize the relationships described informally above.

I do not wish, however, to pass off a hard problem as an easy one. In general, finding appropriate view specifications for telecommunications systems is hard work, and composing them to form a consistent, coherent whole is even harder. But even if view integration eludes us in the forseeable future, there are still reasons to believe that the approach to telecommunications specification illustrated in this paper is the right one.

For one reason, within a well-chosen view, problems of realistic complexity—problems of behavior, description, and analysis—can be solved. There is no evidence that general-purpose approaches, trying to include all features in the same view, can achieve these insights or solve problems of this complexity.

For an additional reason, a view with real explanatory power is worth having, even if it is not formally integrated with other views. For example, the routing specification presented in this paper is extremely easy to implement and test. It provides much better documentation of the features than is usually available.

For a final reason, a collection of useful (but unintegrated) views is exactly the raw material we need to make sense of telecommunications. If we had such a collection, formal methods for development of telecommunications software would improve much more rapidly than they are improving now.

7 ACKNOWLEDGMENTS

Ramesh Bharadwaj, Gertjan Noordman, Monica Sentoff, and Jos van de Brug gave me useful and timely information on this subject. Tim Griffin, Michael Jackson, Peter Mataga, and Sue Tripp provided many helpful and insightful comments. Also, Sue Tripp pointed out the broad applicability of the subscriber/telephone distinction.

8 REFERENCES

Bouma, L.G. and Velthuijsen, H., editors (1994) *Feature interactions in telecommunications systems* (Proceedings of the Second International Workshop on Feature Interactions in Telecommunications Systems). IOS Press, Amsterdam.

Boumezbeur, R. and Logrippo, L. (1993) Specifying telephone systems in LOTOS. *IEEE Communications*, **31**, 38-45.

Cameron, E.J., Griffeth, N., Lin, Y.-J., Nilson, M.E., Schnure, W.K., and Velthuijsen, H. (1993) Towards a feature interaction benchmark. *IEEE Communications*, **31**, 64-69.

Griffeth, N. and Velthuijsen, H. (1992) The negotiating agent model for rapid feature development, in *Proceedings of the Eighth International Conference on Software Engineering for Telecommunications Systems and Services*, Institution of Electrical Engineers, London.

Homayoon, S. and Singh, H. (1988) Methods of addressing the interactions of intelligent network services with embedded switch services. *IEEE Communications*, **26**, 42-46,70.

IWFI (1992) International Workshop on Feature Interactions in Telecommunications Software Systems, workshop notes. St. Petersburg, Florida.

Kay, A., and Reed, J.N. (1993) A rely and guarantee method for Timed CSP: A specification and design of a telephone exchange. *IEEE Transactions on Software Engineering*, **19**, 625-639.

Mataga, P., and Zave, P. (1995) Multiparadigm specification of an AT&T switching system, in *Applications of formal methods* (Hinchey, M.G., and Bowen, J.P., editors), Prentice-Hall International, London.

Muller, J. (1994) Service and service feature interaction: Service creation, service management and service execution aspects. ETSI/NA6 Working Document.

Parnas, D.L., Madey, J., and Iglewski, M. (1994) Precise documentation of well-structured programs. *IEEE Transactions on Software Engineering*, **20**, 948-976.

Schessel, L. (1992) Administrable feature interaction concept, in *Proceedings of the Fourteenth International Switching Symposium*, Institute of Electronics, Information, and Communications Engineers.

Spivey, J.M. (1992) *The Z notation: A reference manual (second edition).* Prentice-Hall International, London.

Woodcock, J., and Loomes, M. (1988) *Software engineering mathematics: Formal methods demystified.* Pitman Publishing, London.

Zave, P., and Jackson, M. (1993) Conjunction as composition. *ACM Transactions on Software Engineering and Methodology*, **2**, 379-411.

9 APPENDIX: PROOF OF TERMINATION

For the forwarding specification given in Section 4, if the delegate relations do not change during routing, and if the irreflexive transitive closure of all the delegate relations is disjoint from the identity relation, then routing is guaranteed to terminate.

First, consider what can happen during quick alerting. The call may be abandoned or answered; in either case termination has been achieved. So the only possibility to consider is a timeout followed by evaluation of *timeout-outcome*. Case 3 of *timeout-outcome* (Table 5) also guarantees termination, so only the first two cases are worrisome. Table 8 gives an alternate definition of *telephone-outcome* (Table 4) in which the problematic Cases 1 and 2 of *timeout-outcome* have been collapsed into Cases 3 and 4 of *telephone-outcome*.

case	**of . . .** *telephone-outcome(old-tel,old-subs) = (new-state,new-tel,new-subs)*	
1	*POTS(old-tel)* $\in$ *Busy* $\wedge$ *old-subs* $\in$ **dom** *DelB*	*subscriber-outcome(DelB(old-subs))*
2	*POTS(old-tel)* $\in$ *Busy*	*(busy,old-tel,old-subs)*
3	*old-subs* $\in$ **dom** *DelNA*	*subscriber-outcome(DelNA(old-subs))*
4	*old-subs* $\in$ **dom** *FMNA* $\wedge$ *old-tel = old-subs*	*telephone-outcome(FMNA(old-subs),old-subs)*
5	*true*	*(alerting,old-tel,old-subs)*

Table 8

As a result of these case eliminations, it is sufficient to prove termination of evaluation of *subscriber-outcome* as defined in Table 3, using *telephone-outcome* as defined in Table 8.

All-Delegates^{+} can be represented by an acyclic graph. For DN *d* in this graph, let *len(d)* be the length of the unique path from *d* to a final node. For DN *d* not in this graph, let *len(d) = 0.* Thus if $d \notin$ **dom** *All-Delegates* then *len(d) = 0,* and *len(d)* is finite for all $d \in DN$.

I shall now prove that an evaluation of *subscriber-outcome(d), len(d) > 0,* either terminates or results in a recursive evaluation *subscriber-outcome(d'), len(d') = len(d) – 1.*

In the definition of *subscriber-outcome*, Case 1 clearly satisfies the hypothesis. Cases 2 and 3 lead to evaluations of *telephone-outcome(FM(d),d)* and *telephone-outcome(d,d),* respectively. Now consider these evaluations.

In the definition of *telephone-outcome*, all cases except Case 4 clearly satisfy the hypothesis. The evaluation of *telephone-outcome(FM(d),d)* cannot invoke Case 4, so all we need to consider is application of Case 4 to *telephone-outcome(d,d).* This results in evaluation of *telephone-outcome(FMNA(d),d).* The latter evaluation cannot invoke Case 4 of the definition of *telephone-outcome*, so it must satisfy the hypothesis.

Finally, I shall prove that an evaluation of *subscriber-outcome(d), len(d) = 0,* terminates.

In the definition of *subscriber-outcome*, Case 1 cannot apply. Cases 2 and 3 lead to evaluations of *telephone-outcome(FM(d),d)* and *telephone-outcome(d,d),* respectively. Now consider these evaluations.

In the definition of *telephone-outcome*, Cases 2 and 5 terminate, and Cases 1 and 3 cannot apply. The evaluation of *telephone-outcome(FM(d),d)* cannot invoke Case 4, so all we need to consider is application of Case 4 to *telephone-outcome(d,d).* This results in evaluation of *telephone-outcome(FMNA(d),d)*. In the evaluation of *telephone-outcome(FMNA(d),d),* Cases 1, 3, and 4 cannot apply, and Cases 2 and 5 terminate.

By induction on *len(d),* all evaluations of *subscriber-outcome(d)* terminate.

10 BIOGRAPHY

Pamela Zave received the A.B. degree in English from Cornell University, Ithaca, New York, and the Ph.D. degree in computer sciences from the University of Wisconsin—Madison. She began her career as an Assistant Professor of Computer Science at the University of Maryland, College Park. Since 1981 she has been with AT&T Bell Laboratories at Murray Hill, New Jersey, and is now a Distinguished Member of Technical Staff in the Software Specification and Analysis Research Department.