Feature Logics and refinement

Marc Aiguier  
L.a.M.I., Université d’Évry  
523 place des Terrasses  
91000 Évry, France  
aiguier@lami.univ-evry.fr

Christophe Gaston  
L.a.M.I., Université d’Évry  
523 place des Terrasses  
91000 Évry, France  
gaston@lami.univ-evry.fr

Pascale Le Gall  
L.a.M.I., Université d’Évry  
523 place des Terrasses  
91000 Évry, France  
legall@lami.univ-evry.fr

ABSTRACT
In this paper, we present an institution of feature logics which generalises the earlier approach of [6] in various way. Then we define a refinement theory to deal with the complexity of feature interactions in this generic framework. Indeed, one of the main problem encountered when dealing with feature interaction is the complexity of the detection. There is a general agreement on the point that implementation techniques could be helpful. However, the study of interactions through implementation techniques is still an open problem. The authors furnish answers to encounter this purpose in a logic-independent framework, using algebraic refinement techniques as basic incremental methods to simplify the feature interaction task.

Keywords
feature, feature interaction, formal specification, abstract logical framework, refinement.

1. INTRODUCTION
The concept of feature has emerged in telephone analysis as a way of describing optional services to which telephone users may subscribe. Features are defined to be separately designed, implemented and integrated. In this way, each feature is described on a partial knowledge of the telecommunication system where it is expected to be plugged on: this partial knowledge denotes a set of requirements that the feature of interest aims to update by adding and/or changing some parts of it. Consequently, adding a new feature in a system can lead to modify its previous behaviour and the partiality of the knowledge of the system may lead to uncontrolled modifications. Those latest may prove unexpected or even undesirable from the subscribers point of view. This is usually called feature interaction problem. Detecting unexpected interactions is fundamental for telephone companies. Indeed, more and more features are offered by them. The primary motivation seems to be that services differentiate providers, and are significant sources of revenue in their own right. Consequently, all unexpected interactions unsolved when a new feature is integrated, may have calamitous consequences from an economic point of view. Thus, in order to ensure working-order of systems, all new services has to be assured before being integrated. However, verifying such a constraint is a hard task which often requires rigorous methods in order to verify many (usually infinite) scenario of interactions. In this way, formal methods are right candidates to specify expected behaviours of systems as well as reasoning about them. At present, the concept of feature is recognised to be not limited to telecommunication systems. Any part or aspect of a specification which has a self-contained functional role can be viewed as a feature. Therefore, feature interaction problem can occur also in other large evolving software systems as pointed out by several authors [1, 17, 20, 6].

The feature interaction detection problem has attracted attention from both theoretical and practical point of view. For a presentation of the main motivations and issues see [20]. In this way, many approaches have been developed to deal with this problem in the last decade [4, 7, 17]. Most of them use some existing formal methods (temporal logic [15, 25, 19], automatas [5, 10, 28, 29, 21], or first order logics [27, 8, 11]) to specify features, and then use different tools to detect feature interactions (model checking, formal proofs, testing...). One of the main open problem encountered when dealing with feature oriented designing is that some interactions may appear at different levels of designing. [20] points out that some interactions may be inherited from higher to lower levels but also that some others may disappear through the operation of concretisation: for example, an interaction due to a non-deterministic description of functionality behaviours may disappear in a lower level of designing where those behaviours are replaced by deterministic ones. [20] draws then the conclusions that:

- interactions should be addressed at the levels they emerge from, and
- it is necessary to have a framework that supports the mapping from one level to the next lower one.

The authors of the present contribution agree with those two points and furnish answers to encounter those requirements in a logic-independent framework. We propose then to use algebraic refinement techniques as basic incremental methods to simplify the feature interaction task. The underlying motivation is that interactions are easier to study on more abstract specifications where some details are hidden. Of course, as pointed out in [20], the problem of the correction of the mapping from one level to the next one is essential: the important point is to study interaction preservation when dealing with more concrete specifications. To achieve this purpose, we define such techniques in the frame of our previous works [6] where we have described a logic-independent framework to specify feature and feature systems, and have denoted within both feature integration and
feature interaction notions. The idea to use refinement techniques in feature conception has already been done to model the composition of features and study interactions between them in the framework of the B method [8, 9, 23, 24]. Here, the main contribution is this incremental technique is defined in a logic-independent framework, and then the preservation results established in this paper are so. Therefore, all logics which satisfy all requirements of our generic framework are equipped with a methodology giving a way to study interactions for each abstraction level of the specification.

The paper is organised as follows: In Section 2 and Section 3, we recall basic definitions of feature logics such as defined in [6]. In this way, we start in Section 2.1 by recalling standard notations about institutions [18] and inclusion systems [12]. Then, in Section 2.2, we give basic requirements we expect from base institutions to be extended into feature logics, and introduce in Section 2.3, the institution of feature logic. Section 3 is devoted to give a semantic denotation to both feature integration and feature interactions. In Section 4, we define refinement theory in the generic framework of feature logics and give preservation results of feature interactions in this refinement theory. Finally, we leave recapitulation and briefs comments on future works in Section 5.

2. FEATURE LOGIC

In this section we recall the logical framework for the specification of feature systems such as defined in [6]. The main difference between this presentation and the one introduced in [6] is that this logical framework is itself viewed as an institution. Formally, we show how extending any institution satisfying some basic requirements into an institution of feature logic. Then, we give denotational semantics of the main problems of feature designing, namely: feature integration and feature interaction.

2.1 Preliminaries

We assume the reader to be familiar with the basic notions of category theory [2, 22] like the notions of category, functors, epics, pullback, co-product and their dual.

2.1.1 Institutions.

Institutions have been introduced in [18] to formally axiomatise the notion of logical system from a model theoretical point of view. An institution is a quadruple $(\text{Sig}, \text{Sen}, \text{Mod}, \models)$ where $\text{Sig}$ is a category of signatures, $\text{Sen} : \text{Sig} \rightarrow \text{Set}$ is a functor which maps every signature to its set of sentences, $\text{Mod} : \text{Sig} \rightarrow \text{Cat}$ is a functor which maps every signature to its category of models, and $\models = (\models_{x} | x \in \text{Sig})$ is a $\text{Sig}$-indexed family of contravariant functors which maps every signature to its category of models, and $\models_{x} (\models_{x})$. Given a signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, $\text{Mod}(\sigma) : \text{Mod}(\Sigma') \rightarrow \text{Mod}(\Sigma)$ is called the reduct functor. Moreover, this quadruple satisfies the following property, so-called, satisfaction condition: $\forall (\sigma : \Sigma \rightarrow \Sigma') \in \text{Hom}_{\text{Sig}}, \forall M \in \text{Mod}(\Sigma'), \forall \phi \in \text{Sen}(\Sigma'), M \models_{\Sigma'} \text{Sen}(\sigma)(\phi) \Leftrightarrow \text{Mod}(\sigma)(M) \models_{\Sigma} \phi$. In such a generic framework, a specification is any couple $S = (\Sigma, ax)$ where $\Sigma$ is a signature and $ax \subseteq \text{Sen}(\Sigma)$. The notation $M \models_{\Sigma} \phi$ is extended in the usual way to categories of models and sets of formulas. Given a specification $S = (\Sigma, ax)$ and a signature collection $\Sigma$, all $\phi$ is called a semantic consequence of a specification $S = (\Sigma, ax)$, also denoted $SP \models \phi$, if and only if for every $M \in \text{Mod}(S)$, $M \models_{\Sigma} \phi$. The set of semantic consequences of $S$ is noted $SP^*$.

Example 1 (CTL* as an institution). CTL* [14] is a branching-time temporal logic where the structure representing all possible executions tree-like rather than linear. It is well-adapted to specify and reason about non-deterministic and/or concurrent processes. When dealing with propositional fragment of logics, a signature is only a set of propositional variables. Thus $\text{Sig}_{\text{CTL}*}$ is simply the category $\Sigma$ whose class contains any set and morphisms are functions between sets. The mathematical definition of a $\Sigma$-model $M$ is given by a set $S$ of states, a binary relation $R \subseteq S \times S$ satisfying: $\forall s \in S, \exists s' \in S, s R s'$, and a total function $I : S \rightarrow \mathcal{P}(\Sigma)$ called labelling function. Therefore, $\Sigma$-models are labelled transition systems where $R$ denotes the transition relation and $L$ is the labelling associating for each state $s$ the set of propositional variables true at $s$. For any signature $\Sigma$, we note $\text{Mod}_{\text{CTL}*}(\Sigma)$ the category whose objects are all $\Sigma$-models and whose morphisms between two $\Sigma$-models $M = (S, R, L)$ and $M' = (S', R', L')$ are total functions $f : S \rightarrow S'$ compatible with relation and labelling functions. Now for all signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, $\text{Mod}_{\text{CTL}*}(\sigma) : \text{Mod}_{\text{CTL}*}(\Sigma') \rightarrow \text{Mod}_{\text{CTL}*}(\Sigma)$ is the functor defined by $\text{Mod}_{\text{CTL}*}(\sigma)((S', R', L')) = (S', R', L')$ where for all $s \in S'$, $L'(s) = \{p \in \text{Sen}(\sigma)(p) \in L'(s)\}$. A $\text{Sigma}$-axiom is a well-formed formula whose the syntactical rules are given by:

\[
\begin{align*}
\text{CTL}^* &::= \text{ATOM} \mid \text{CTL}^* \rightarrow \text{CTL}^* \mid \\
&\quad \forall \text{CTL}^* \land \text{CTL}^* \mid \exists \text{CTL}^*
\end{align*}
\]

\[
\text{ATOM} ::= p \mid \bot
\]

$p$ being any propositional variable of $\Sigma$. We note $\text{Sen}_{\text{CTL}*}(\Sigma)$ the set of all the $\Sigma$-axioms.

Validity of $\Sigma$-axioms is expressed via a binary relation usually denoted by $\models_{\Sigma}$ between $\Sigma$-models and $\Sigma$-axioms. Then, $M \models_{\Sigma} \phi$ if for any sequence of states $\sigma = (s_0, s_1, \ldots, s_n, \ldots)$ (so-called runs) s.t. for every $i < j \in \mathbb{N}$ we have $s_i R s_j$, $M$ satisfies $\phi$, denoted by $(M, \sigma) \models_{\Sigma} \phi$, according to the following inductive definition: for any $i \in \mathbb{N}$, let us note $\sigma^i$ the subsequence $(s_i, s_{i+1}, \ldots, s_n, \ldots)$ of $\sigma$.

- $(M, \sigma) \models_{\Sigma} p$ if $p \in L(s_0)$ for $p \in \Sigma$;
- $(M, \sigma) \not\models_{\Sigma} \bot$;
- $(M, \sigma) \models_{\Sigma} \phi_1 \rightarrow \phi_2$ if $(M, \sigma) \models_{\Sigma} \phi_1$ then $(M, \sigma) \models_{\Sigma} \phi_2$;
- $(M, \sigma) \models_{\Sigma} \forall \phi \text{ iff for all } i \in \mathbb{N}, (M, \sigma^i) \models_{\Sigma} \phi$;
- $(M, \sigma) \models_{\Sigma} \exists \phi \text{ iff there exists } i \in \mathbb{N}, (M, \sigma^i) \models_{\Sigma} \phi$;
- $(M, \sigma) \models_{\Sigma} \exists \phi \text{ iff there is } j \in \mathbb{N} \text{ s.t. } (M, \sigma^j) \models_{\Sigma} \phi$ and for all $k < j \in \mathbb{N}, (M, \sigma^k) \models_{\Sigma} \phi$;
- $(M, \sigma) \models_{\Sigma} \forall \phi \text{ iff for any run } \sigma' \text{ starting to } s_0$, $(M, \sigma') =_{\Sigma} \phi$;
- $(M, \sigma) \models_{\Sigma} \exists \phi \text{ iff there exists a run } \sigma' \text{ starting to } s_0$, $(M, \sigma') =_{\Sigma} \phi$;
We note that the family of relations \( \{ R \subseteq \mathcal{S} \mid R \in \mathcal{S} \} \). The quadruple \((\mathcal{S}, \mathcal{C}, \mathcal{M}, \mathcal{D}, \mathcal{O})\) is then easily an institution (no pruning is done on sets of states throughout signature morphisms).

### 2.1.2 Inclusion systems.

Inclusion systems have been defined in [12] to give abstract basis for combining signatures (and more generally specifications) in a systematic way. The definition of inclusion systems is motivated by the fact that the categorical concept of pushout, usually used to this purpose, is too general because union is defined up to isomorphism. To solve this problem, the pushout must be parameterised by an object denoting the sharing between signatures, and equipped with morphisms denoting inclusions in order to avoid renaming. An inclusion system for a category \( \mathcal{C} \) with pullbacks is then a pair \((\mathcal{I}, \mathcal{E})\) such that \( \mathcal{I} \) and \( \mathcal{E} \) are categories whose class of objects is \( |\mathcal{C}| \), morphisms in \( \mathcal{E} \) are epics, every morphism in \( \mathcal{I} \) can be factored uniquely as \( \iota \circ e \) with \( \iota \in \text{Hom}_\mathcal{I} \) and \( e \in \text{Hom}_\mathcal{E} \). Hom\( \mathcal{I} \) defines a partial order in \( \mathcal{C} \) and the poset \((\mathcal{C},\text{Hom}_\mathcal{I})\) is a lattice where the sup of \( A \) and \( B \) is the sum \( A + B \) and their inf is the unique inclusion pullback denoted \( A \cap B \) and called intersection of \( A \) and \( B \). A category equipped with an inclusion system is strongly inclusive if and only if for all objects \( A, B \in |\mathcal{C}| \), \( A + B \) is the pushout of \( A \cap B \). Morphisms in \( \mathcal{I} \) are called inclusions. In the following, any inclusion \( \iota \in \text{Hom}_\mathcal{I}(A, B) \) will be noted \( A \to B \). Inclusion systems allow to retrieve usual properties of set-theoretical inclusions at this abstraction level (see [12] for more explanations).

Example 2. As signatures in \( \mathcal{C} \) are sets, the most natural inclusion system for \( \mathcal{C} \) is \((\mathcal{I}_{\mathcal{C} \subseteq \mathcal{C}}, \mathcal{E}_{\text{inclusion}})\), where \( \mathcal{I}_{\mathcal{C} \subseteq \mathcal{C}} \) contains all set-theoretical inclusions and \( \mathcal{E}_{\text{inclusion}} \) contains all surjections. Therefore, \( + \) is the set union and \( \cap \) is the set intersection.

### 2.2 Base institution

Here, we give the basic requirements that the base institution must satisfy to be extended into a feature logic. Therefore, we assume that we are given a base institution \( B = (\mathcal{S}, \mathcal{C}, \mathcal{M}) \) satisfying the following requirements:

- \( \mathcal{S} \) is strongly inclusive. Moreover it has an initial object \( \Sigma_i \) which is also initial in \( \mathcal{I} \), where \((\mathcal{I}, \mathcal{E})\) is the inclusion system of \( \mathcal{S} \).
- For every \( \Sigma \to \Sigma' \) in \( \mathcal{I} \), we have \( \text{Sen}(\Sigma) \subseteq \text{Sen}(\Sigma') \).
- For every \( \Sigma, \Sigma' \in |\mathcal{S}| \), \( \text{Sen}(\Sigma \cup \Sigma') = \text{Sen}(\Sigma) \cup \text{Sen}(\Sigma') \).
- For every \( \Sigma, \Sigma', \Sigma'' \in |\mathcal{S}| \), \( \Sigma \cap (\Sigma' + \Sigma'') \) is defined and equals \((\Sigma \cap \Sigma') + (\Sigma \cap \Sigma'') \).

The first requirement allows us to avoid renaming throughout inclusions. This is important because a feature generally characterises behaviours of new functionalities from previously defined ones. Therefore, a feature description introduces just a few new symbols. The greatest part of the used symbols refers to already existing functionalities. Now, it is natural to preserve both signature inclusions and signature intersections through the functor \( \text{Sen} \) as expressed by the second and third requirements. The last requirement is a technical point which will be used to obtain most of the results of section 4. It is a categorical characterisation of the usual (set) property that intersection is distributive on union.

### Example 3 (\( \mathcal{C} \) as a base institution). \( \emptyset \) (the empty set) is initial in \( \mathcal{C} \) (the category of sets). Finally, because well-formed formula in \( \mathcal{C} \) are inductively defined from a set of propositional variables, for every signatures \( \Sigma \subseteq \Sigma' \), we trivially have \( \text{Sen}(\Sigma) \subseteq \text{Sen}(\Sigma') \). This concludes that the institution of \( \mathcal{C} \) defined in Example 1 is a base institution.

### 2.3 The institution of feature logics

We are now ready to extend any base institution \( B = (\mathcal{S}, \mathcal{C}, \mathcal{M}) \) with \((\mathcal{I}, \mathcal{E})\) as inclusion system, into an institution \((\mathcal{S}_{\text{inclusion}}, \mathcal{C}_{\text{inclusion}}, \mathcal{M}_{\text{inclusion}})\) devoted to feature specification. As we have already glimpsed in Introduction, a feature is always designed with a partial knowledge of the system on which it is supposed to be plugged on. Roughly speaking, this knowledge can be classified in the following taxonomy:

- Knowledge about the system behaviours the feature aims at enriching or modifying.
- Knowledge about the system behaviours which must be preserved whatever modifications are done on it.

Our definition of feature signature reflects this structuration. Therefore, we obtain:

**Definition 1 (Feature signature)***. A feature signature \( \Sigma_f \) is a triple \((\mathcal{S}_{\text{inclusion}}, \mathcal{I}_{\text{inclusion}}, \Sigma_f)\) such that \( \text{Sen}(\Sigma_f) = (\mathcal{S}, \mathcal{M}) \) is a \( \mathcal{C} \)-specification, \( \text{Inv} \subseteq \text{Sen}(\Sigma) \) and \( \Sigma_f \to \Sigma' \) is a \( \mathcal{C} \)-morphism. Moreover, the following condition holds: \( \mathcal{R}_f = \mathcal{S}_{\text{inclusion}} \).

\( \mathcal{R}_f \) (for "required") denotes the part of the system specification where the feature will be plugged on, \( \mathcal{I}_{\text{inclusion}} \) denotes the behaviours to be preserved on this part, and \( \Sigma_f \to \Sigma' \) denotes the introduction of new syntactical symbols. Three examples of feature signatures are given in Example 4.

**Definition 2 (Feature signature morphism).*** Let \( \Sigma_{f_1} = (\mathcal{S}_{f_1}, \mathcal{I}_{f_1}, \Sigma_1 \to \Sigma') \) and \( \Sigma_{f_2} = (\mathcal{S}_{f_2}, \mathcal{I}_{f_2}, \Sigma_2 \to \Sigma'_2) \) be two feature signatures with \( \mathcal{R}_{f_1} = (\Sigma_{f_1}, \alpha_{f_1}) \) for \( i = 1, 2 \). A feature signature morphism from \( \Sigma_{f_1} \) to \( \Sigma_{f_2} \) is a pair \((\rho : \Sigma_1 \to \Sigma_2, \rho_1 : \mathcal{S}_1 \to \mathcal{S}_2)\) satisfying the following conditions:

- \( \Sigma_1 \to \Sigma_2 \circ \rho_1 = \rho_2 \circ \Sigma_1 \to \Sigma_2 \)
- \( \text{Sen}(\rho_1)(\alpha_{f_1}) \subseteq \alpha_{f_2} \) and \( \text{Sen}(\rho_2)(\mathcal{S}_{f_1}) \subseteq \mathcal{S}_{f_2} \).

The sentences associated to feature signatures are devices to formalise new behaviours of a system. For a feature signature \( \Sigma_f = ((\Sigma, \mathcal{M}), \mathcal{I}_{\text{inclusion}}, \Sigma_f) \), \( \Sigma' \) denotes all the syntactical elements which are useful to this formalisation. Consequently, the set of sentences associated to \( \Sigma_f \) is obviously \( \text{Sen}(\Sigma') \). In the same way, models are supposed to give a semantical counterpart to all the syntactic elements introduced in signatures. Therefore, for every signature \( \Sigma_f = ((\Sigma, \mathcal{M}), \mathcal{I}_{\text{inclusion}}, \Sigma_f) \), it is natural to define
a $\mathcal{F}$-model as an element of $\text{Mod}(\Sigma')$. However, $\Sigma_{\mathcal{F}}$ contains another information: a set of properties which must remain true (Inv). Consequently, $\text{Mod}_{\mathcal{F}\text{-cat}}(\Sigma_{\mathcal{F}})$ is the full sub-category of $\text{Mod}(\Sigma')$ whose objects are all $\Sigma_{\mathcal{F}}$-models which validate Inv (i.e. $\text{Mod}_{\mathcal{F}\text{-cat}}(\Sigma') = \text{Mod}(\Sigma', \text{Inv}))$). Finally, regarding to the definitions of both models and sentences associated to a feature signature, the relation $\models_{\mathcal{F}\text{-cat}}$ is $\models_{\Sigma_{\mathcal{F}}}$.

**Proposition 1** (Institution of Feature Logics). The quadruple $\text{Ins}_{\mathcal{F}\text{-cat}} = (\Sigma_{\mathcal{F}\text{-cat}}, \text{Sen}_{\mathcal{F}\text{-cat}}, \text{Mod}_{\mathcal{F}\text{-cat}}; \models_{\mathcal{F}\text{-cat}})$ is an institution whereby:

- $\Sigma_{\mathcal{F}\text{-cat}}$ is the category of feature signatures and feature signature morphisms equipped with the composition $\odot$ defined as follows: $\forall (\rho_1, \rho_2) : \Sigma_{\mathcal{F}_{\text{cat}}} \to \Sigma_{\mathcal{F}}$ and $(\rho_1 \odot \rho_2) : (\Sigma_{\mathcal{F}} \to \Sigma_{\mathcal{F}})$, $\rho_1 \odot \rho_2 = (\rho_1 \circ \rho_1 \odot \rho_2)$.

- The functor $\text{Sen}_{\mathcal{F}\text{-cat}} : \Sigma_{\mathcal{F}\text{-cat}} \to \text{Set}$ maps:
  
  - each feature signature $\Sigma_{\mathcal{F}} = (\text{Req, Inv, } \Sigma \hookrightarrow \Sigma')$, to $\text{Sen}(\Sigma')$.
  
  - each feature signature morphism $\rho = (\rho_1, \rho_2)$ to $\text{Sen}(\rho_2)$.

- The functor $\text{Mod}_{\mathcal{F}\text{-cat}} : \Sigma_{\mathcal{F}\text{-cat}} \to \text{Cat}$ maps:
  
  - each feature signature $\Sigma_{\mathcal{F}} = (\text{Req, Inv, } \Sigma \hookrightarrow \Sigma')$, to $\text{Mod}(\Sigma', \text{Inv})$.
  
  - each feature signature morphism $\rho = (\rho_1, \rho_2)$ to $\text{Mod}(\rho_2)

where for each feature signature $\Sigma = (\text{Req, Inv, } \Sigma \hookrightarrow \Sigma')$, $\models_{\Sigma_{\mathcal{F}}} = \models_{\Sigma'}$.

By following Definitions of Section 2.1, a feature specification is any couple $(\Sigma_{\mathcal{F}}, \alpha_{\mathcal{F}})$ where $\Sigma_{\mathcal{F}}$ is a feature signature and $\alpha_{\mathcal{F}} \subseteq \text{Sen}_{\mathcal{F}\text{-cat}}(\Sigma_{\mathcal{F}})$. Some examples of feature specifications using $\text{CTL}^+$ as base institution are given in Example 4.

### 3. FEATURE INTEGRATION AND FEATURE INTERACTIONS

#### 3.1 Feature systems and feature integration

Roughly speaking, you can see a feature system as a set of features provided with a structuring denoting the integration order of features in the system.

**Definition 3** (Feature System). A feature system is a finite sequence of feature specifications $S_{\mathcal{F}} = (\mathcal{F}_i)_{i \leq n}$, such that:

1. $\Sigma_{\mathcal{F}_i}$ is of the form $(\Sigma_{\mathcal{F}}, \alpha_{\mathcal{F}}), \emptyset, \Sigma_{\mathcal{F}} \hookrightarrow \Sigma_{\mathcal{F}_i}$.
2. For all $1 \leq i \leq k \leq n$, if $\Sigma_{\mathcal{F}_i} = (\text{Req}_i, \text{Inv}_i, \Sigma_i \hookrightarrow \Sigma_i')$ and $\Sigma_{\mathcal{F}_k} = (\text{Req}_k, \text{Inv}_k, \Sigma_k \hookrightarrow \Sigma_k')$ then $\Sigma_k(\Sigma_{\mathcal{F}_i} \hookrightarrow \Sigma_k) \exists\Sigma_k$.

$\mathcal{F}_i$ is the basic system (in telecommunications systems, it is usually called POTS as Plain Old Telephone Service), and the second condition ensures that one can not design a feature using already-existing symbols without introducing them in the required specification.

**Example 4**. Herein, we present a small example of feature system, including a simplified basic telephone feature $B$, a call waiting feature $CW$ and a call forward when busy feature $CFB$. These features will be discussed in their respective associated sections.

The basic telephone system feature: $B$. Due to the lack of space, it is not possible to present the whole basic telephone feature. The feature we present here will only express the unique axiom which is required by the features $CW$ and $CFB$. There are $N$ phones involved in the basic system. The basic telephone feature is of the form $B = (((\Sigma_b, \emptyset), \emptyset, \Sigma_b \hookrightarrow \Sigma_b), \alpha_{\Sigma_b})$ where $\Sigma_b$ denotes the empty signature and:

- $(\Sigma_b, \emptyset)$ only contains the following axiom schemata:
  
  $B_1 : \forall (\exists P_i \text{calls } P_j \land \neg \text{Free } P_i) \rightarrow \exists X (P_i \text{ fuse } P_j)\]

  $B_1$ expresses that whenever a call of source $P_i$ and target $P_j$ is done, if $P_i$ is not free, then the call will be refused in the next state.

The call-forwarding feature: $CW$. When a phone $P_i$ is not free, $CW$ allows it to put on hold any call $P_i \text{ (Axiom CW1).}$ When it is the case then possibly the two phones can get engaged at the next time (Axiom CW2), but it will not happen if one of the two phones become free (Axiom CW3). The feature $CW$, is of the form $CW = (((\Sigma_{CW}, \emptyset), \emptyset, \Sigma_{CW} \hookrightarrow \Sigma_{CW}), \alpha_{\Sigma_{CW}})$, where $\Sigma_{CW} = \Sigma_b \cup \bigcup_{\Sigma_{\mathcal{F}} \in \text{Ins}_{\mathcal{F}\text{-cat}}} \{P_i \text{ holds } P_i\}$ and $\alpha_{\Sigma_{CW}}$ is the set containing the three following axioms schemata:

$CW_1 : \forall (\exists P_i \text{ calls } P_j \land \neg \text{Free } P_i) \rightarrow \exists X (P_i \text{ fuse } P_j \land P_j \text{ calls } P_i)\]

$CW_2 : \forall (\exists P_i \text{ holds } P_i \rightarrow \exists X (P_i \text{ online } P_i)\]

$CW_3 : \forall (\exists P_i \text{ calls } P_j \land \neg \text{Free } P_i \land \neg P_j \text{ holds } P_j) \rightarrow \exists X (P_i \text{ fuse } P_j)\]

The call-forward when busy feature: $CFB$. When a phone $P_i$ is not free, $CFB$ allows its users to forward all the incoming calls to another phone (Axiom CFB1). When $CFB$ is not used, the incoming calls are refused (Axiom CFB2). The feature $CFB$ is of the form $CFB = (((\Sigma_{CFB}, \emptyset), \emptyset, \Sigma_{CFB} \hookrightarrow \Sigma_{CFB}), \alpha_{\Sigma_{CFB}})$, where $\Sigma_{CFB} = \Sigma_b \cup \bigcup_{\Sigma_{\mathcal{F}} \in \text{Ins}_{\mathcal{F}\text{-cat}}} \{P_i \text{ fuse } P_j\}$ and $\alpha_{\Sigma_{CFB}}$ is the set containing the two following axiom schemata:

$CFB_1 : \forall (\exists P_i \text{ calls } P_j \land \neg \text{Free } P_i \land \neg (\forall \rightarrow (P_i \text{ fuse } P_j)) \rightarrow \neg (P_i \text{ fuse } P_j))\]

$CFB_2 : \forall (\exists P_i \text{ calls } P_j \land \neg \text{Free } P_i \land \neg (P_i \text{ fuse } P_j) \rightarrow \neg (P_j \text{ calls } P_i))\]

As usual, the class of models associated to a structured specification is grounded on a flat specification (i.e. of the
form \((\sum \alpha x)\). In our context, such a specification will reflect all the modifications done on the basic system in order to integrate all the features described in the system. Now, let us recall that the required specification of a feature denotes the sub-part of the system which is modified, and the axioms of the feature denotes the new intended behaviours. The integration of a feature on a system will then consist on replacing the required axioms by the axioms of the feature in the axiomatic component of the system specification.

**Definition 4** (Integration sequence). Let \(\text{Sys} = (\mathcal{F}_{i})_{i \leq n}\) be a feature system. The integration sequence associated to \(\text{Sys} \) is the sequence of specifications \(\text{Sys}_{\text{int}} = (\text{Sys}_{i})_{i \leq n}\) inductively defined as follows:

- \(\text{Sys}_{1} = (\Sigma_{i}^{\alpha_{1}}, \alpha x_{1})\);
- \(\text{Sys}_{k} = (\Sigma_{k}^{\alpha_{k}}, \alpha x_{k})\) and \(\text{Sys}_{k+1} = (\Sigma_{k+1}^{\alpha_{k+1}}, \alpha x_{k+1})\) with \(\Sigma_{k+1}^{\alpha_{k+1}} = (\alpha_{k}, \Sigma_{k+1}^{\alpha_{k+1}}, \alpha x_{k+1})\) and \(\text{Re}_{\alpha_{k+1}} = (\Sigma_{k+1}^{\alpha_{k+1}}, \alpha x_{k+1})\). Then \(\text{Sys}_{k+1} = (\Sigma_{k+1}^{\alpha_{k+1}}, \alpha x_{k+1}^{*} = (\alpha x_{k}^{*} \vdash \alpha x_{k+1}) \cup \alpha x_{k+1})\).

**Example 5.** With notations of Example 4, let us consider the system \((B, C, W, C, F, B)\). The integration sequence (see Definition 4) for this system is \(\text{Sys}_{1}, \text{Sys}_{2}, \text{Sys}_{3}\) where \(\text{Sys}_{1} = (\Sigma_{0}^{B}, \{B_{1}\})\), \(\text{Sys}_{2} = (\Sigma_{C, W, \{x_{2}\}}\), where \(\alpha x_{2} = \{CW_{1}, CW_{2}, CW_{3}\}\) and \(\text{Sys}_{3} = (\Sigma_{C, W, C, F_{3}, \{x_{3}\}}\), where \(\alpha x_{3} = \{CW_{1}, CW_{2}, CW_{3}, F_{1}, F, B_{2}\}\).

Let us note that in any feature system, several features may be designed referring to the same part of the system. Consequently, some axioms belonging to the required specifications of a feature specification \(\mathcal{F}\) may no longer occur at the integration step of \(\mathcal{F}\). Indeed, it is possible that a feature \(\mathcal{F}\) introducing the same set of requirements has been integrated before \(\mathcal{F}\). Nevertheless, it is quite natural to impose that this required specification was a sub-specification of the system some time in the integration history before the integration of \(\mathcal{F}\). When this constraint is satisfied for all features of the system, we will say that this latest is sensible. This is formally defined as follows:

**Definition 5** (Sensible system). With all the notations of Definition 4, \(\text{Sys} = (\mathcal{F}_{i})_{i \leq n}\) is said to be sensible if and only if for all \(2 \leq i \leq n\), there exists \(j < i\) such that if \(\alpha x_{i}\) is the axiomatic component of the required specification of \(\mathcal{F}_{i}\) and \(\alpha x_{j}^{*}\) is the axiomatic component of the \(j^{k}\) specification in the integration sequence associated to \(\text{Sys}\) then we have: \(\alpha x_{j} \subseteq \alpha x_{i}^{*}\).

The last specification of the sequence of integration associated to \(\text{Sys}\) denotes clearly the system after that all the features introduced by \(\text{Sys}\) have been integrated. Let us recall that the sets of invariants introduced by the features signatures of the system are the properties that must remain true. Therefore, the non-validation of feature invariants enlightens so-called feature interference (cf. [20]) which are undesirable interactions. The semantic of feature systems is then defined as follows:

**Definition 6** (Feature system model). Let \(\text{Sys} = (\mathcal{F}_{i})_{i \leq n}\) be a feature system. A Sys-model is any element \(M\) of \(\text{Mod}(\text{Sys}_{i})\) such that for all \(2 \leq i \leq n\), \(M \models \text{In}_{\alpha_{i}}\). Let us note \(\text{Mod}(\text{Sys})\) the class of all the Sys-models.

Consequently, a feature system is consistent if there is at least one (un-trivial) model in \(\text{Mod}(\text{Sys})\). Therefore, feature integration problems at the specification level are reduced to verify that feature systems obtained after the integration are consistent.

### 3.2 Feature interactions

Before giving denotational semantics of feature interactions, we need to give a categorical axiomatisation of the set minus notion. This will be useful to denote sentences of interest when comparing a feature and the system on which feature interactions are studied.

#### 3.2.1 Difference between signatures

As it is argued in [12], even though set inclusions are the simplest natural example of inclusion system, inclusion systems in general may have properties that are quite different from those of sets. For example, any inclusion \(A \subseteq B\) in set theory is "split" in the sense that \(B\) can be written as a disjoint union \(A \sqcup C\) where \(C\) is the "difference" between \(B\) and \(A\). This property does not hold for all inclusion systems. For example, given a many sorted signature inclusion \((\{s\}, \emptyset) \subseteq \{s, \{f : s \to s\}\}\), the difference \(\emptyset \setminus \{f : s \to s\}\) is not a signature. The expected result should be \(\emptyset \setminus \emptyset\). Nevertheless, this notion can be expressed in a categorical manner if we do not ground this axiomatisation on property of disjoint union. Indeed, we can characterise the difference between \(B\) and \(A\), denoted \(B \setminus A\), as the greatest set of \(X\) included in \(B\) such that \(X \cap A = \emptyset\). Therefore, we assume in the following that every base inclusion system \(B = (\Sigma_{B}, \text{Sen}_{B}, \text{Mod}_{B}, \emptyset)\) satisfies the following supplementary requirement: for all signatures \(\Sigma\) and \(\Sigma' \in \Sigma_{B}\), let us define \(\Sigma_{B} \setminus \Sigma'\) as the full sub-category of \(\Sigma\), equipped with the class of objects \(|\Sigma_{B} \setminus \Sigma'|\) defined as follows: \(|\Sigma_{B} \setminus \Sigma'| = \{X | X \cap \Sigma = \emptyset \land X \models \Sigma'\}\). Then, \(\Sigma_{B} \setminus \Sigma'\) has a terminal object that we note \(\Sigma' \models \Sigma\) and call the difference between \(\Sigma'\) and \(\Sigma\).

#### 3.2.2 Semantical denotation of feature interactions

As we glimpsed in Introduction, the feature interaction problem is strictly speaking a "break" of the modularity. Succinctly, it characterises unexpected behaviours of a feature when it is plugged on a system. In this way, the feature interaction problem consists in comparing "what is true for the feature" with "what is true for the system". More formally, true properties for a feature specification \(\mathcal{F}\) is naturally \(\mathcal{F}\). Now, when dealing with the system specification, the characterisation is not so obvious. The problem is that most properties in \(\text{Sys}\) (the set of true properties for \(\text{Sys}\)) do not involve the feature \(\mathcal{F}\): all those which do not contain any syntactical elements introduced by \(\mathcal{F}\). Those latest belong to the set of formulas defined from the vocabulary introduced by the system minus the vocabulary introduced by the feature. Therefore, this set must be removed from \(\text{Sys}\).

**Definition 7** (Representative properties). Let \(\text{Sys} = (\mathcal{F}_{i})_{i \leq n}\) be a feature system where for all \(i \leq n\) \(\Sigma_{\text{Sys}_{i}}\) is of the form \((\text{Re}_{i}, \text{In}_{i}, \Sigma_{i} \models \Sigma'_{i})\). Let \(\text{Sys}_{\text{rep}} = (\Sigma_{\text{rep}}, \alpha x_{\text{rep}}^{*})\) be the final specification of the integration sequence associated to \(\text{Sys}\). Let us note \(\text{In}_{\text{rep}} = \bigcup_{1 \leq i \leq n} \text{In}_{\alpha_{i}}\).

The set of representative properties for \(\mathcal{F}_{i}\) in \(\text{Sys}\) is the set \(\text{Sys}_{\text{rep}}^{*}\), defined as follows:

\[\text{Sys}_{\text{rep}}^{*} = \end{equation}
The existence of interactions between a feature and a system can be exhibited by the existence of, either some properties (dealing with the feature) of the system which are not properties of the feature, or on the contrary some properties of the feature which are not properties of the system. These properties are respectively called emerging properties and non-conformance properties. Formally, they are defined as follows:

**Definition 8 (Feature Interaction).** Let $\text{Sys} = (\mathcal{F}_i)_{i \in \mathbb{N}}$ be a feature system. For each $i \in \{1, \ldots, n\}$, let us call:

1. **Set of emerging properties for $\mathcal{F}_i$ w.r.t. $\text{Sys}$**, the set $\mathcal{EP}(\mathcal{F}_i, \mathcal{S}_{\text{sys}})$ defined by: $\mathcal{EP}(\mathcal{F}_i, \mathcal{S}_{\text{sys}}) = \mathcal{S}_{\text{sys}}^* \setminus \mathcal{F}_i^*$;

2. **Set of non-conformance properties for $\mathcal{F}_i$ w.r.t. $\text{Sys}$**, the set $\mathcal{NCP}(\mathcal{F}_i, \mathcal{S}_{\text{sys}})$ defined by: $\mathcal{NCP}(\mathcal{F}_i, \mathcal{S}_{\text{sys}}) = \mathcal{F}_i^* \setminus \mathcal{S}_{\text{sys}}^*_{\mathcal{F}_i}$.

Sentences in those two sets do not necessarily denote "bad" behaviours. The decision about the qualitative nature of them is an expert matter. The aim is then to specify this knowledge in order to select among those sentences the bad ones. The interest here is that we can take benefit of our formal setting to achieve such a purpose by using formal tools of the underlying base institution such as the calculus or more efficient methods as functional test, model checking, ...

**Example 6.** Let us consider the feature system of Example 5.

It is easy to see that $\mathcal{EP}(\mathcal{CW}, \mathcal{S}_{\text{sys}}) \neq \emptyset$. Indeed from $\mathcal{CW}$ and $\mathcal{CFU}$ we deduce that the formula:

$$\forall Q(P_k \text{ calls } P_j \land \neg \text{Free}_j) \land
\begin{align*}
&\text{CF}_i P_j \rightarrow \chi [P_k \text{ calls } P_k \land \neg \text{Free}_k])
\end{align*}$$

is an element of $\mathcal{S}_{\text{sys}}^*_{\mathcal{CW}}$. It is easy to see that it is not an element of $\mathcal{CW}^*$. It reveals a sub-specification interaction between $\mathcal{CW}$ and the system $\mathcal{S}_{\text{sys}}$. The reason is that the activation conditions for $\mathcal{CW}$ and $\mathcal{CFU}$ are the same (roughly speaking: the callee is not free).

Another example of interaction is revealed by the fact that $B_1 \in \mathcal{NCP}(\mathcal{F}, \mathcal{S}_{\text{sys}})$ which is quite natural: both $\mathcal{CW}$ and $\mathcal{CFU}$ aim at modifying the behavior expressed by this axiom.

4. **Feature Refinement**

Usually, algebraic refinement of specifications denotes a simple syntactical extension of signatures. Axioms of the abstract specification are removed and replaced by more concrete ones. In our framework, the syntactical refinement of a feature specification $\mathcal{F} = (\Sigma_{\mathcal{F}}, \alpha_{\mathcal{F}})$ with $\Sigma_{\mathcal{F}} = (\text{Reg}_i, \text{Inv}_i, \Sigma_i \rightarrow \Sigma_i')$ will then consist on increasing $\Sigma_i'$ replacing $\alpha_{\mathcal{F}}$ by new axioms $\alpha_{\mathcal{F}}^*$. Both requirement specification $\text{Reg}$ and set of invariants $\text{Inv}$ are preserved because they denote the behaviours of features which are in front of $\mathcal{F}$ in the integration order. Therefore, they are not concerned by the refinement of $\mathcal{F}$.

**Definition 9 (Feature Refinement).** A feature specification $\mathcal{F}_{\text{imp}} = (\Sigma_{\mathcal{F}_{\text{imp}}}, \alpha_{\mathcal{F}_{\text{imp}}})$ with $\Sigma_{\mathcal{F}_{\text{imp}}} = (\text{Reg}_i, \text{Inv}_i, \Sigma_i \rightarrow \Sigma_{\text{imp}})$ is a refinement of a feature specification $\mathcal{F} = (\Sigma_{\mathcal{F}}, \alpha_{\mathcal{F}})$ with $\Sigma_{\mathcal{F}} = (\text{Reg}_i, \text{Inv}_i, \Sigma_i \rightarrow \Sigma_i')$ if and only if $\Sigma_i \rightarrow \Sigma_{\text{imp}}$.

**Example 7.** Let us consider the basic telephone system $B$ presented in Example 4. We can refine this feature by adding in the signature a proposition which denotes an authorisation of communication (between phones) given by the system.

$\mathcal{B}_{\text{imp}}$ is then defined as follows:

$\mathcal{B}_{\text{imp}} = ((\Sigma_{\mathcal{F}}, \emptyset, 0, \Sigma_i \rightarrow \Sigma_{\text{imp}}), \alpha_{\mathcal{B}_{\text{imp}}})$ where $\Sigma_i$ denotes the empty signature and:

$$\Sigma_{\text{imp}} = \Sigma_B \cup \bigcup_{i \in \mathbb{N}} \left\{ \text{sys}_{\mathcal{F}_{\text{imp}}} \left( P_i \right) \right\}$$

$\alpha_{\mathcal{B}}$ only contains the following axiom schemata:

$$B_{\text{imp}1} : \forall (Q(P_i \text{ calls } P_i) \land \neg \text{Free}_i) \rightarrow \forall (\neg \text{sys}_{\mathcal{F}_{\text{imp}}} (P_i))$$

$$B_{\text{imp}2} : \forall (\neg \text{sys}_{\mathcal{F}_{\text{imp}}} (P_i) \rightarrow P_i \text{ resets } P_i))$$

Axiom $B_{\text{imp}1}$ expresses that whenever a call of source $P_i$ and target $P_i$ is done, if $P_i$ is not free, the system refuses the communication. Axiom $B_{\text{imp}2}$ expresses that whenever the system refuses a communication, this communication does not happen.

In algebraic approaches, a refinement will be correct provided that the behaviour of the implementation is indistinguishable from the behaviour of the higher level specification under consideration. Given a refinement $S_{\mathcal{F}_{\text{imp}}}$ of a specification $S = (\Sigma, \alpha)$ in any institution where a refinement theory has been defined, its correctness should be then expressed as follows: $S^* = S_{\mathcal{F}_{\text{imp}}}^* \cap \text{Sen}(\Sigma)$. However, it is more usual to define correctness of implementations on classes of models. This is due to the analogy usually done between models and specifications realisations (i.e., programs). When dealing with loose semantics, the major aim of refinement on the semantical level is then to cut down the class of specification models. Therefore, given a refinement $S_{\mathcal{F}_{\text{imp}}} = (\Sigma_{\text{imp}}, \alpha_{\mathcal{F}_{\text{imp}}})$ of a specification $S = (\Sigma, \alpha)$, this leads to the following definition of correctness: $\text{Mod}(\Sigma \rightarrow \Sigma_{\text{imp}}) \subseteq \text{Mod}(S_{\mathcal{F}_{\text{imp}}}^*)$. From such a definition, we easily show that $S_{\mathcal{F}_{\text{imp}}}^* \subseteq \text{Mod}(S_{\mathcal{F}_{\text{imp}}}^*)$. However, the equation $S^* = S_{\mathcal{F}_{\text{imp}}}^* \cap \text{Sen}(\Sigma)$ does not hold in general. To solve this problem, different type of refinement semantics have been defined (cf. [13] for a complete survey on this subject) whose the most promising seems to be the observational approach [3].

Here, as we are mainly interested in the effect of refinements on feature interactions, feature correctness is simply defined on semantical consequences, that is:

**Definition 10 (Correctness).** $\mathcal{F}_{\text{imp}}$ is a correct refinement of $\mathcal{F}$ if and only if $S^* = S_{\mathcal{F}_{\text{imp}}}^* \cap \text{Sen}_{\text{fat}}(\Sigma_{\mathcal{F}})$.

**Example 8.** If we consider the refinement of Example 7, it is easy to see that $B_1$ is a semantical consequence of $B_{\text{imp}1}$ and $B_{\text{imp}2}$. Now any model $\langle S, R, L \rangle$ of $B$ can be extended in one (or several) model(s) of $B_{\text{imp}}$ which only have the property that for any state $s \in S$ such that $P_i \text{ calls } P_j \in L(s)$ and $\text{Free}_P \notin L(s)$. Therefore for all $s \in S$, $\text{sys}_{\mathcal{F}_{\text{imp}}} (P_i, P_j) \notin L(s)$. Thus $\text{Mod}_{\text{fat}}(\Sigma_B \rightarrow \Sigma_{\text{imp}}) \subseteq \text{Mod}_{\text{fat}}(B_{\text{imp}})$. From satisfaction condition we obtain $B^* = B_{\text{imp}}^* \cap \text{Sen}_{\text{fat}}(\Sigma_{\mathcal{F}})$, where $\Sigma_{\mathcal{F}}$ is the signature of $B$. 


Of course, as an immediate consequence of Definition 10, we have: $\text{Mod}_{\text{ref}}(\Sigma_F \mapsto \Sigma_{F_{\text{imp}}}) \subseteq \text{Mod}_{\text{ref}}(\mathcal{F})$. This directly follows from the satisfaction condition.

Let us consider a feature $\mathcal{F}$, a refinement $F_{\text{imp}}$ of $\mathcal{F}$ and a system $S_{\text{sys}}$ which contains $F_{\text{imp}}$ in its definition. To define the system which takes into account the implementation, the first idea could be to replace $F_{\text{imp}}$ by $F_{\text{imp}}$ in $S_{\text{sys}}$. However, it is not sufficient. Indeed, features defined after $F$ in $S_{\text{sys}}$ may introduce required specifications that are grounded on axioms introduced by $F$. Consequently, in the refined system, these axioms must be replaced by some axioms of $F_{\text{imp}}$. But how choosing axioms to substitute? Most of logics used in computer science satisfy the following property, so-called consequence compactness [26]: given a specification $SP = (\Sigma, ax)$, if $\text{SP} = \varphi$ then there exists a finite subset $S$ of $ax$ such that $(\Sigma, S) = \varphi$. Moreover in practice, it is always possible to choose among all these subsets, one which is minimal (in terms of cardinality). Therefore, in the following, we suppose the base institution on which the feature logic is defined satisfies such a property. From there, we obtain the following definitions:

**Definition 11 (Refinement axioms).** Let $F_{\text{imp}} = (\Sigma_{F_{\text{imp}}}, ax_{F_{\text{imp}}})$ be a refinement of $F = (\Sigma, ax_F)$ with $\Sigma_F = (\text{Req}, \text{Inv}, \Sigma \mapsto \Sigma')$ and $\Sigma_{F_{\text{imp}}} = (\text{Req}, \text{Inv}, \Sigma \mapsto \Sigma_{\text{imp}})$. For all $\varphi \in ax_F$ let us call refinement axioms of $\varphi$ (with respect to $F_{\text{imp}}$) any subset $ax_\varphi$ of $ax_F$ such that $(\Sigma_{F_{\text{imp}}}, ax_\varphi) = \text{sat}_\varphi$.  

**Example 9.** Regarding to Example 7, we only have to impose $ax_{B_1} = \{B_{1_{\text{imp}}}, B_{1_{\text{imp}2}}\}$.  

Of course, we have to impose that the new vocabulary introduced by refinement is not already introduced by some other features of the system. This gives rise to the notion of compatible system:

**Definition 12 (Compatible system).** Let $F_{\text{imp}} = (\Sigma_{F_{\text{imp}}}, ax_{F_{\text{imp}}})$ be a refinement of $F = (\Sigma, ax_F)$ with $\Sigma_F = (\text{Req}, \text{Inv}, \Sigma \mapsto \Sigma')$ and $\Sigma_{F_{\text{imp}}} = (\text{Req}, \text{Inv}, \Sigma \mapsto \Sigma_{\text{imp}})$. Let $S_{\text{sys}} = (\mathcal{F}, j)$ be a sensible feature system which contains $F$.

The refinement $F_{\text{imp}}$ of $F$ is said to be compatible with $S_{\text{sys}}$ if and only if for all $\Sigma_{j}$ with $j \leq n$ we have $\Sigma_{i} \cap \Sigma_{\text{imp}} = \Sigma_{i} \cap \Sigma_{j}$. Therefore, the system implementation will be obtained by replacing all formulas introduced by $F_{\text{imp}}$ by any refinement axioms. However, another condition has to be imposed in order to define a concrete system which is lucky enough to be an implementation of the abstract one. This condition states that, for any two abstract formulas $\varphi$ and $\varphi'$, their refinement axioms are distincts ($ax_\varphi \cap ax_{\varphi'} = \emptyset$). The reason is that $\varphi$ and $\varphi'$ will be replaced by their refinements axioms in the concrete system (see Definition 14). Regarding to Definition 4 removing or adding $\varphi$ (through an integration step) at the abstract level must have the same effect than removing or adding $ax_\varphi$ (through an integration step) at the concrete level. If the condition $ax_\varphi \cap ax_{\varphi'} = \emptyset$ does not hold, we compromised the fact that the formula $\varphi'$ is still implemented by some axioms in the concrete system. This gives rise to the notion of sound refinement axioms:

**Definition 13 (Soundness).** With all the notations of Definition 11, a collection of refinements axioms $(ax_\varphi | \varphi \in ax_F)$ is sound if and only if for all $\varphi, \varphi' \in ax_F$, $ax_{\varphi} \cap ax_{\varphi'} = \emptyset$.

This condition is generally satisfied. Indeed, specification implementation makes usually in practice a correspondence one-to-one between operations of the implementing specification and operations of the specification to be implemented. Moreover, this correspondence is defined in a constructive way. This leads to distinct groups of equations, each one defining the implementation of one operation of the abstract specification.

**Definition 14 (Implemented systems).** Let $F = (\Sigma_F, ax_F)$ be a feature. Let $F_{\text{imp}}$ be a correct refinement of $F$ compatible with a system $S_{\text{sys}}$. Let $(ax_\varphi)_{\varphi \in ax_F}$ be a sound collection of refinements axioms. Let us note $S_{\text{sys}F_{\text{imp}}}$ the system obtained from $S_{\text{sys}}$ by replacing $F$ with $F_{\text{imp}}$ and every $\varphi \in ax_F$ occurring in some requirements specifications, or in some axiomatic components of some feature in $S_{\text{sys}}$, with $ax_\varphi$. Afterwards, $S_{\text{sys}F_{\text{imp}}}$ will be called implemented system of $S_{\text{sys}}$ with respect to $(ax_\varphi)_{\varphi \in ax_F}$, and it will be noted $S_{\text{sys}F_{\text{imp}}}$.

**Example 10.** Regarding to the system presented in Example 5 and the refinement presented in Example 7, we only have to replace the required specification $(\Sigma_{n}, (B1)$ of both $CW$ and $CFB$ by $(\Sigma_{F_{\text{imp}}}, (B1_{\text{imp}}, (B1_{\text{imp}2})$. From there, we obtain the following fundamental property, so-called horizontal refinement [13]:

**Theorem 1 (Horizontal refinement).** With all the notations of Definition 14, we have: $S_{\text{sys}} \subseteq S_{\text{sys}F_{\text{imp}}}$. The proof of Theorem 1 is given in the appendix.

Therefore, horizontal refinement deals with refinement of subparts of system specifications when they are structured in "blocks". Here blocks are features.

We are now in position of stating our non-conformance and emerging properties preservation results. These results are suitably adapted from [16].

**Theorem 2 (Preservation results).** With all notations of Definition 14, if we note $\Sigma_{n+}^{\text{sys}}$ the signature of the last specification of the integration sequence associated to $S_{\text{sys}}$, both following results hold:

1. $\text{CP}(F, S_{\text{sys}}) \subseteq \text{CP}(F_{\text{imp}}, S_{\text{sys}F_{\text{imp}}}) \cap \text{Sen}(\Sigma_{n+}^{\text{sys}})$. 
2. $\text{NCP}(F_{\text{imp}}, S_{\text{sys}F_{\text{imp}}}) \cap \text{Sen}_{\text{ref}}(\Sigma_F) \subseteq \text{NCP}(F, S_{\text{sys}})$. 

The proof of Theorem 2 is given in the appendix.

These results mean that refinement techniques can be used to study emerging properties on some abstractions of a given feature specification. The idea is then to reduce the complexity of feature interactions by hiding some details (e.g. the user point of view can be viewed as an abstraction of its complete specification). However Theorem 2 states that refinements make non-conformance properties disappear through refinement steps. This is usually the consequence of design choices which can naturally solve some interactions (e.g. an interaction due to a non-determination of a functionality which becomes deterministic). In [16], supplementary conditions have been given in order to have complete preservation results on non-conformance properties throughout abstraction specifications (i.e. where some details are hidden). Under these supplementary conditions, interactions can also be studied through abstraction of feature specification.
5. CONCLUSION

In this paper, after having recalled the general framework dedicated to specify feature and feature-oriented systems, as well as the definitions of both emerging and non-conformance properties such as introduced in [3], we have defined a theory of feature refinement in a logic-independent way. If such a refinement is correct, then the refined feature can be replaced by its refinement in any system and this preserve the global behaviour of the system (Theorem 1). Moreover, we have studied feature interaction preservations through refinements. We have shown that emerging properties are preserved but some non-conformance properties can disappear (Theorem 2). This is due to the design choices. Feature specifications written by using CTF as base institution have been given in this paper in order to illustrate all definitions. Other examples of feature logics can be found in [3, 16].

Several research issues still need to be explored:

• extending this refinement theory in order to refine a feature specification by a feature system.

• taking benefits of our logic-independent setting to investigate towards other techniques well studied in this framework such as functional testing.

Appendix

Proof of Proposition 1

Let us note : \( F_{imp} = \{(\text{Req. Inv}, \Sigma \mapsto \Sigma_{imp}^\delta); \text{ar}_{F_{imp}}\} \) and \( F = \{(\text{Req. Inv}, \Sigma \mapsto \Sigma); \text{ar}_F\} \). Moreover let \( SP_{sys}^\delta = \{(\Sigma_{sys}^\delta, \text{ar}_{sys}^\delta)\} \) (respectively \( SP_{sys} = \{(\Sigma_{sys}, \text{ar}_{sys})\} \) be the latest specification of the integration sequence (Definition 4) of the system \( Sys \) (respectively \( Sys_{imp} \)).

It is easy to prove by induction on natural numbers that \( \Sigma_{sys}^\delta = \Sigma_{sys}^\delta + \Sigma_{imp}^\delta \), and then \( \Sigma_{sys}^\delta \mapsto \Sigma_{sys}^\delta \). Now let us note \( Inv = \bigcup_{1 \leq n} I_{inv} \). Then the set of all invarians introduced by all the features of \( Sys \) is the one introduced by all the features of \( Sys_{imp} \). From there, it is easy to prove by induction on the integration step that for every \( x \) we have both following cases to consider:

• if \( \phi \in ax_{sys}^\delta \cap ax_{sys} \), the soundness condition of Definition 13 ensures that \( ax_{sys} \subseteq ax_{sys}^\delta \) and Inv such that \( \Sigma_{sys}^\delta \mapsto \Sigma_{sys}^\delta \).

• if \( \phi \in ax_{sys}^\delta \) \( \cap ax_{sys} \) then obviously \( \phi \in ax_{sys}^\delta \).

Therefore, we have \( \Sigma_{sys}^\delta = \Sigma_{sys}^\delta + \Sigma_{imp}^\delta \) and Inv such that for the satisfaction condition gives:

\( \forall \phi \in ax_{sys}^\delta \cap ax_{sys} \), the soundness condition of Definition 13 ensures that \( ax_{sys} \subseteq ax_{sys}^\delta \) and Inv such that \( \Sigma_{sys}^\delta \mapsto \Sigma_{sys}^\delta \). That is: \( \text{Mod}(\Sigma_{sys}^\delta \mapsto \Sigma_{sys}^\delta) \subseteq \text{Mod}(\Sigma_{sys}^\delta) \) and then: \( Sys_{sys} \subseteq Sys_{imp} \).

Proof of Theorem 2

All results need the following technical results:

**Lemma 1.** With notations of Theorem 1, we have: \( Sys_{sys} \subseteq Sys_{imp} \).

**Proof.**

\( \Sigma_{sys}^\delta \mapsto \Sigma_{imp}^\delta = \text{sup}(X \setminus X \cap \Sigma_{imp}^\delta = \emptyset \land X \mapsto \Sigma_{sys}^\delta) \).

Thus because \( \Sigma_{sys}^\delta = \Sigma_{sys}^\delta + \Sigma_{imp}^\delta \) and \( \Sigma_{sys}^\delta \mapsto \Sigma_{imp}^\delta = \text{sup}(X \setminus X \cap \Sigma_{imp}^\delta = \emptyset \land X \mapsto (\Sigma_{sys}^\delta + \Sigma_{imp}^\delta) = X) \).

But \( X \cap (\Sigma_{sys}^\delta + \Sigma_{imp}^\delta) = X \cap \Sigma_{sys}^\delta + X \cap \Sigma_{imp}^\delta = X \cap \Sigma_{sys}^\delta \) (see section 2.2). Thus: \( \Sigma_{sys}^\delta \mapsto \Sigma_{imp}^\delta = \text{sup}(X \setminus X \cap \Sigma_{imp}^\delta = \emptyset \land X \cap \Sigma_{sys}^\delta = X) \).

In other words: \( \Sigma_{sys}^\delta \mapsto \Sigma_{imp}^\delta = \text{sup}(X \setminus X \cap \Sigma_{imp}^\delta = \emptyset \land X \mapsto \Sigma_{sys}^\delta) \).

From Definition 12, for all \( X \mapsto \Sigma_{sys}^\delta \), we have \( X \cap \Sigma_{imp}^\delta = X \cap \Sigma_\Sigma \). Therefore, we directly conclude \( \Sigma_{sys}^\delta \mapsto \Sigma_{imp}^\delta \subseteq \Sigma_{sys}^\delta \subseteq \Sigma_\Sigma \).

Now from Definition 7: \( Sys_{sys} \subseteq \{\phi \in Sys_{sys} \setminus \phi \notin Sen(\Sigma_{sys}^\delta \cap \Sigma_\Sigma)\} \) and \( Sys_{imp,sys} \subseteq \{\phi \in Sys_{imp} \setminus \phi \notin Sen(\Sigma_{sys}^\delta \cap \Sigma_\Sigma)\} \).

Therefore, from Theorem 1 we deduce: \( Sys_{sys} \subseteq Sys_{imp,sys} \).

**FACT 1.** With notations of Definition 13, if we note \( \Sigma_{sys}^\delta \) the signature of the latest specification of the integration sequence associated to \( Sys \), then:

\( Sen_{fesat}(\Sigma_{imp}) \cap Sen(\Sigma_{sys}^\delta) = Sen_{fesat}(\Sigma_\Sigma) \).

**Proof.**

The last condition of Definition 12 and distributivity of \( \cap \) on \( + \) allow us to deduce:

\( Sen_{imp}(\Sigma_{sys}^\delta) \cap Sen(\Sigma_{sys}^\delta) = Sen_{fesat}(\Sigma_\Sigma) \).

Now from Definition 10: \( \mathcal{F}^\ast = \mathcal{F}_{imp} \cap Sen_{fesat}(\Sigma_\Sigma) \), thus we can write:

\( Sys_{sys} \subseteq Sys_{imp} \cap Sen(\Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \).

From Fact 1:

\( Sen_{fesat}(\Sigma_{imp}) \cap Sen(\Sigma_{sys}^\delta) = Sen_{fesat}(\Sigma_\Sigma) \),

we can thus write:

\( Sys_{sys} \subseteq (Sys_{imp} \cap Sen(\Sigma_{sys}^\delta)) \cap Sen_{fesat}(\Sigma_\Sigma) \).

So we easily conclude:

\( \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \subseteq \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \).

**Proof of NCP(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \subseteq \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \subseteq \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \).

From Lemma 1: \( Sys_{sys} \subseteq Sys_{imp} \cap Sen_{fesat}(\Sigma_\Sigma) \).

But from Definition 10: \( \mathcal{F}^\ast = \mathcal{F}_{imp} \cap Sen_{fesat}(\Sigma_\Sigma) \), thus we deduce:

\( \mathcal{F}_{imp} \cap Sen_{fesat}(\Sigma_\Sigma) \subseteq \mathcal{F}^\ast \subseteq \mathcal{F}^\ast \cap Sys_{sys} \).

In other words:

\( \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \subseteq \mathcal{F}(\mathcal{F}_{imp}, \Sigma_{sys}^\delta) \cap Sen_{fesat}(\Sigma_\Sigma) \).
6. REFERENCES


