# Formal Modeling for Product Families Engineering

A. Fantechi
DSI - Univ. di Firenze
and ISTI-CNR, Pisa
Via S. Marta 3, I50139 FIRENZE (Italy)
fantechi@dsi.unifi.it

S. Gnesi
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo",
ISTI - CNR,
Via G. Moruzzi 1, I-56124 PISA (Italy)
stefania.gnesi@isti.cnr.it

## Abstract

*In this paper we propose a behavioural model, namely the Generalized Extended Modal Transition Systems, as a basis for the formalization of different notions of variability usually present in product families definitions. In particular, a GEMTS is able to define a family of products by telling at any state of the system whether some (and how many) transitions are optional or mandatory for any derived products of the family. The proposed model is compared with previous proposals also based on Labeled Transition Systems, showing its higher generality, but also pointing out weaknesses that still need to be addressed with more expressive models. Hints on the solution of such weaknesses are given by the use of constraints expressed as temporal logic formulae.*

## 1. Introduction

The development of industrial software systems may often benefit from the adoption of a development cycle based on the so-called *product-families* or *product lines* approach [7, 14]. This approach aims at lowering production costs by sharing an overall reference architecture and concepts of the products, but at the same time allowing them to differ with respect to particular product characteristics in order to, e.g., serve different markets. The production process in product lines is hence organized with the purpose of maximizing the commonalities of the product line and minimizing the cost of variations [25].

A description of a product family (PF) is usually composed by a constant part and a variable part. The first describes aspects common to all products of the family, the second represents those aspects, called *variabilities*, that will be used to differentiate a product from another. The modeling of variability has been extensively studied in the literature, especially that concerning *Feature Modeling*, [2, 8, 16, 13]. In variability modeling the interest is in defining which features or components of a system are optional, alternative, or mandatory; techniques and tools are then developed to show that a product belongs to a family, or to derive instead a product from a family, by means of a proper selection of the features or components.
In this paper we are interested in the behavioural modeling, that is, in describing how a product of a family is able to respond to events in the time, even in presence of variabilities, an aspect that the referred techniques do not typically deal with. Many notations and description techniques have been recently proposed for this purpose, such as variants of UML state diagrams [1, 27] or variants of UML sequence diagrams, for example STAIRS [21]; another proposal is that in [25], where UML state diagrams and sequence diagrams have been enriched with aside notations to describe variation points.
In this paper, the quest for an expressive modeling formalism for families is based on the choice of a basic model, that is Labeled Transition Systems (LTS), which are one of the most popular formal frameworks for modeling and reasoning about the behaviour of a system.

In this paper, we define a general LTS-based framework for describing product families, that is, Generalized Extended Modal Transition Systems (GEMTS), that will allow the case of alternative variabiliy (i.e. the mandatory selection of *at least* one - or some - among several differ-

ent choices) to be modeled. GEMTSs extend the EMTS notion (Extended Modal Transition Systems), we have recently proposed in [11], allowing the modeling of *multiple optionality*, that is, the possibility to select a subset out of the possible choices in a variation point. Starting from a family definition, a product can be derived as a LTS and a *conformance* relation can then be defined between the LTS representing a product and the GEMTS representing the family.
The proposed notions are not intended to be used as such by engineers to describe product families, but rather to give basic modeling concepts on which verification activities can be carried out, when high level specification formalisms for product families are semantically mapped over LTS-based expressions.

In section 2 we introduce behavioural modeling of product families by means of LTS-based formalisms. In section 3 we introduce the GEMTS notion, showing its better adequacy to model product families. A further refinement is given in section 4, by adding to the family definition the possibility of expressing additional constraints over the products by means of the ACTL temporal logic. Section 5 proposes a final discussion of the introduced framework and of its possible extensions.

## 2. Behavioural modeling of product families

Since we are interested in characterizing the dynamic behaviour of a product family, we base our discussion on Labeled Transition Systems (LTS) and therefore we will start defining what a Labeled Transition System is:

**Definition 2.1** *A* Labeled Transition System *(LTS) is a quadruple:* $(S, Act, s_0, \rightarrow)$*, where* $S$ *is a set of states,* $Act$ *is a set of actions used as transition labels,* $s_0 \in S$ *is the initial state, and* $\rightarrow \subseteq S \times Act \times S$ *is the transition relation. If* $(s, t, s') \in \rightarrow$*, we write* $s \xrightarrow{t} s'$*.*

### 2.1. Modeling a family with LTS

When modeling the behaviour of a product as a LTS, products of a family are considered to differ in the actions that they are able to perform at any given state; this means that the definition of a family has to accommodate all the possibilities desired for each derivable product, predicating on the choices that keep a product belonging to the family. We recall some informal definitions concerning variability: By *variation points* we intend those locations in a specification where variability is expressed, by means of listing several choices; variation points may be:
- *alternative*: one and only one of the possibilities should be chosen in a product;
- *optional*: a possibility can be present or not in the product;
- *multiple optional*: at least $m$ out of the $n$ possibilities should be chosen;

**Example 2.1** *In order to duscuss the expressiveness LTS-based modeling, we will use a simple running example, that is, a family of (simplified) vending machines, for which we give the following requirements:*

- *A vending machine is activated by a coin. The only accepted coins are the one euro coin for the European products and the one dollar coin for the US products.*
- *After inserting a coin, the user has to choose whether he wants sugar or not, by pressing one of two buttons. Then, the user may select the drink.*
- *The choice of drinks (coffee, tea, cappuccino) varies between the products. However, every product of the family delivers coffee, and every product of the family delivers at least two different drinks.*
- *After delivering the drink, a done message is displayed, and, optionally, an alert tone is rung.*
- *The machine goes back in the idle state when the cup is taken by the user.*

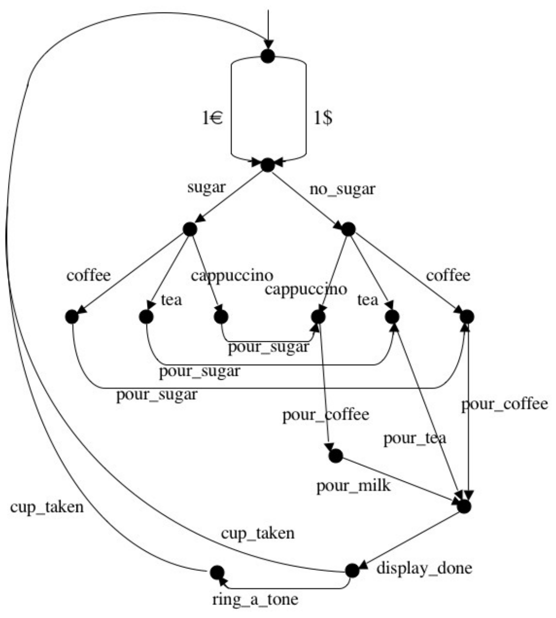

**Figure 1. LTS Model of a product family**

The proposed example has three variation points (namely, the choice of the coins, the choice among provided drinks and the optional alert tone) which are:
- the first, *alternative*

- the second, *multiple optional*: at least two out of three possibilities should be chosen;
- the last, simply *optional*.

We can build a LTS representing all the possible behaviours conceived for the family as that represented in Figure 1: this means that the definition of a family has to accommodate all the possibilities desired for the possible derived products, predicating on the choices that keep a product belonging to the family.

We can notice that the given LTS cannot distinguish required transitions from optional ones, since variation points in the family definition are modeled as nondeterministic choices (i.e. alternative branches), independently from the type of variability, and cannot be distinguished by pure nondeterministic choices.

Starting from the LTS representing the family a set of LTSs representing possible products may be derived. The derivation of a product will amount to choose some alternative branches from any node of a LTS. The following simple algorithm can be used to operate choices by visiting one after the other the states of the LTS:

Given a LTS $T$, visit the graph of $T$ in a breadth-first fashion, applying at each state $s$ the step `visit(s)`, as defined below:

```
visit(s):
   select-transitions(s);
   while( removable transitions exist )
      { if a transition  t  is removable:
            remove t from the graph
            if the state s',  target of the transition t,
             is no more reachable by any other state:
                  remove the state s';
                  mark all its outgoing transitions
                         as "removable";
      }
```

The call to `select-transitions(s)` is actually a request to the user, for each outgoing transition `t` from `s`, to tell whether he/she considers the transition `t` present in the derived product. If not, the transition is marked as "removable". The choice needs to be guided by the family requirements, but this is not enforced by the algorithm, since no knowledge about variation points is embedded in the model.

In our coffee machine example we have that a first product of the family, an audible coffee and cappuccino machine for the US market, has the behaviour given by the LTS in Figure 2. Another product of the family, a silent tea and coffee vending machine for the EU market, is represented by the LTS in Figure 3.

Both the products above can be derived from the family model by means of the algorithm given above. However, many other products may be derived following this algorithm: for example a product that accepts both euro and dollar coins, or a product that does not allow a user to ask for sugar. It is easy to notice that the two latter examples do

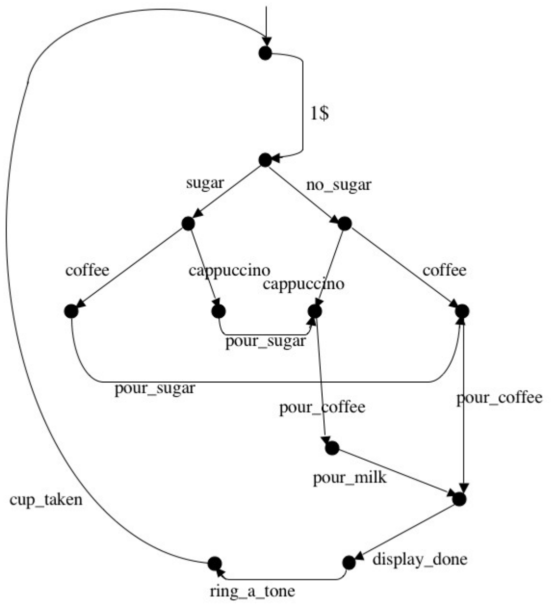

**Figure 2. A vending machine for the US**

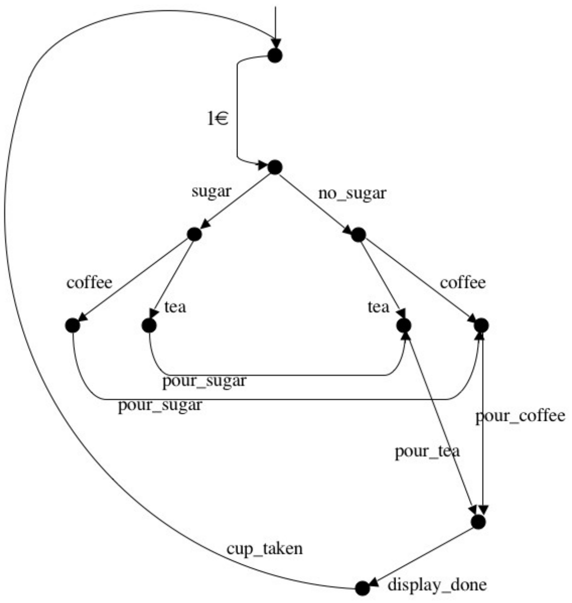

**Figure 3. A vending machine for the EU**

not satisfy the given family requirements, while the two former ones do: responsibility over conformance to the family requirements is left to the user's choices .

## 2.2. Simulation

Any LTS associated to a product derived by the previous algorithm has a *simulation* relation [23] with the LTS representing the family.

**Definition 2.2** *Let $T_1 = (S_1, Act, s_{0_1}, \rightarrow_1)$ and $T_2 = (S_2, Act, s_{0_1}, \rightarrow_2)$ We say that $s_2 \in S_2$* simulates *$s_1 \in S_1$ (written $s_1 \preceq_s s_2$) if there exists a* strong simulation *that relates $s_1$ and $s_2$.*
*$\mathcal{R} \subseteq S_1 \times S_2$ is a strong simulation if $\forall (s_1, s_2) \in \mathcal{R}$ (where $\sigma \in T_1 \cup T_2$),*

- *$s_1 \xrightarrow{\sigma}_1 s_1'$ implies $\exists s_2' : s_2 \xrightarrow{\sigma}_2 s_2'$ and $(s_1', s_2') \in \mathcal{R}$.*

*The above definition is naturally extended to LTSs by considering their initial states: A LTS $T_2$ simulates $T_1$ (written $T_1 \preceq_s T_2$) iff ($s_{0_1} \preceq_s s_{0_1}$)*

In particular, any LTS $P$ defining a product of a family defined by the LTS $F$ is such that $F$ simulates $P$ ($P \preceq_s F$) since all the alternative choices possible in each product $P$ are included in the LTS $F$ describing the family.

## 2.3. Modal Transition Systems

To overcome the low distinguishing power of LTSs when modeling variation points, Modal Transition Systems (MTS) [18] have been proposed to capture variability [12]: in a MTS, transitions may be possible or required, that is, optional or mandatory for a product.

**Definition 2.3** *A MTS $F = (S_F, Act, s_0, \rightarrow_\Box, \rightarrow_\Diamond)$ is defined as a LTS, having two distinct transition relations, namely $\rightarrow_\Box \subseteq S_F \times Act \times S_F$ is the must transition relation, which expresses* required *transitions, $\rightarrow_\Diamond \subseteq S_F \times Act \times S_F$ is the may transition relation, which expresses* possible *transitions.*

A MTS defines a family of LTSs, in the sense that each LTS $P = (S_P, Act, \rightarrow, p_0)$ of the family can be obtained from the MTS $F$ by considering the transition relation $\rightarrow$ to be $\rightarrow_\Box \cup R$, with $R \subset \rightarrow_\Diamond$, and pruning the states that are not reachable from the initial state.

For example, the MTS of Fig. 4 can be drawn to define our family of coffee machines, where .

A similar framework proposed for the same purpose are Modal I/O Automata [17], based on the concept of I/O Automata, which are very close to Labeled Transition Systems, but exploit a characterization of actions in input, output or internal actions. This framework does not add in expressiveness of variability w.r.t. MTSs.

## 2.4. Derivation of products

The "*is a product of F*" relation below, called also "conformance" relation, links a MTS representing a family with a LTS representing a product.

**Definition 2.4** *We say that $P$* is a product of *$F$, written $P \vdash F$, if and only if $p_0 \vdash s_0$, where:*
*$p \vdash f$ if and only if*

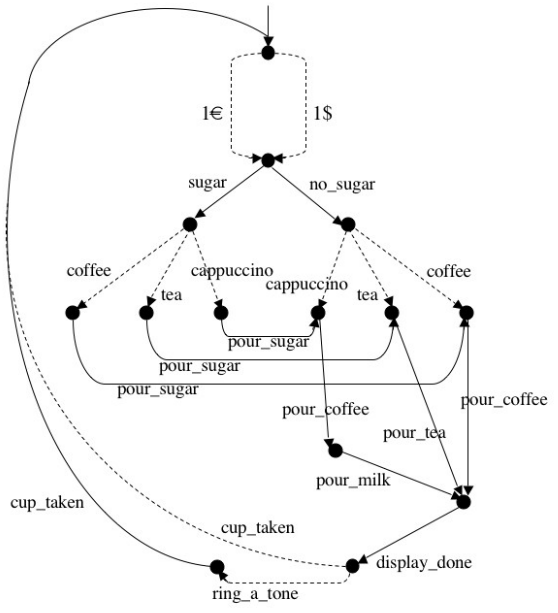

**Figure 4. Modeling a product family with a MTS (solid arcs are *required* transitions, dashed arcs are *possible* transitions)**

- $f \xrightarrow{a}_\Box f' \implies \exists p' \in S_P : p \xrightarrow{a} p'$ *and* $p' \vdash f'$

- $p \xrightarrow{a} p' \implies \exists f' \in S_F : f \xrightarrow{a}_\Diamond f'$ *and* $p' \vdash f'$

The LTS of a product that conforms to a family MTS can be obtained by applying the same derivation algorithm seen in section 2.1, in which the `select-transitions(s)` procedure allows only the possible transitions outgoing from state `s` to be selected as removable; the required transitions cannot be selected as removable. in the graphical syntax, this amount to remove the selected dashed lines and to make the remaining dashed lines become solid. Hence, the set of products derivable are a subset of those derivable by the family LTS and, consequently, any derivable product is still simulated by the LTS family.

The MTS represented in Figure 4 allows the products of Figures 2 and 3 to be derived. However, as happens in the case of LTSs, starting from a MTS it is again possible to derive products that do not satisfy the requirements given in section 2.1: for example, a product that accepts *both* euro and dollar coins can be derived. This is because MTSs are not completely adequate to model the case of alternative variabiliy (i.e. the mandatory selection of at least one among several different choices). In section 3 we address this problem with the proposal of new LTS variants.

## 3. An extended modeling framework for product families

### 3.1. Extended Modal Transition Systems

In [11] we have defined, for a finer modeling of variation points, the notion of Extended Modal Transition Systems, in which at any state of the system, it can be defined whether to choose at least (or at most) one from a subset of the outgoing transitions. Indeed a similar definition (One Variant Modal Transistion Systems - 1MTS) has been independently proposed for the same purpose, by [26], as a variant of the Disjunctive Modal Transition Systems (DMTS) defined in [19]. We postpone the comparison with these variants of Transition Systems to section 3.3.

**Definition 3.1** *An* Extended Modal Transition System *(EMTS) is a quintuple* $(S, Act, s_0, \Box, \diamond)$*, where* $S$ *is a set of states,* $Act$ *is a set of actions,* $s_0 \in S$ *is the initial state,* $\Box \subseteq S \times 2^{Act \times S}$ *is the* at least 1-of-n *transition relation, and* $\diamond \subseteq S \times 2^{Act \times S}$ *is the* at most 1-of-n *transition relation.*

We write respectively: $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box} s_1, s_2 \ldots, s_n$ and $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond} s_1, s_2 \ldots, s_n$ to denote elements of the two relations, meaning that in the first case any product of the family should have at least one of the $n$ transitions $s \xrightarrow{a_i} s_i$, while in the second case any product of the family should have at most 1 of the $n$ transitions (that is, it can also have no transition from this set). Note that in the arrow-like writing style, the number of the actions on the arrow must coincide with that of target states, and order counts as well, since each action is paired to the corresponding state.

Figure 5 shows two examples of simple EMTSs: the first means that action $a$ is possible and one of the actions $b, c$ is required. The second means that one of the actions $a, b, c, d$ is possible while one of the actions $c, d, e$ is required.

Figure 6 exemplifies the conjunction:
$s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box} s_1, s_2 \ldots, s_n$
*and* $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond} s_1, s_2 \ldots, s_n$,
which means: any product of the family should have *exactly* 1 of the $n$ transitions $s \xrightarrow{a_i} s_i$; this defines the relation $\Box \cap \diamond$ as the relation *exactly 1 of n*: elements of this relation could be written for brevity $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box\diamond} s_1, s_2 \ldots, s_n$. In Figure 6, exactly one of the actions $a, b, c$, should be performed.

A MTS can be defined as an EMTS where only singleton pairs from $Act \times S$ appear in the relations $\Box$ and $\diamond$ (that is, $n = 1$): this is obvious considering that the two relations can be read "at least 1-of-1" and "at most 1-of-1".

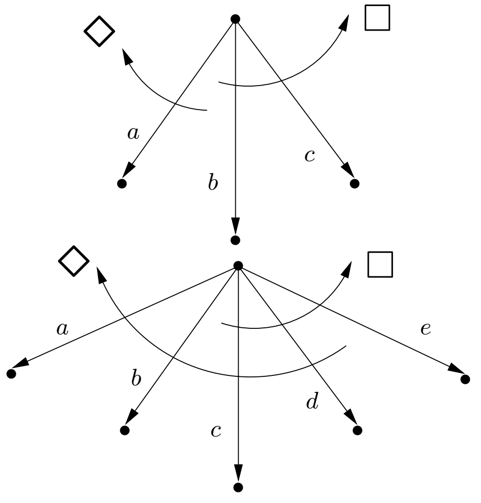

**Figure 5. Examples of EMTS**

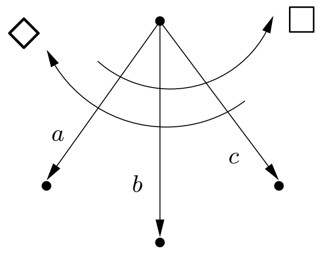

**Figure 6. Example of "exactly one" EMTS**

An EMLTS defines a family of LTSs according to the following definition:

**Definition 3.2** *An LTS* $P = (S_P, Act, \rightarrow, p_0)$ *belongs to the family (EMLTS)* $F = (S_F, Act, f_0, \Box, \diamond)$ *(we say also* is a product of F*, written* $P \vdash F$ *) if and only if* $p_0 \vdash f_0$*, where:*
$p \vdash f$ *if and only if*

- $f \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box} f_1, f_2 \ldots, f_n \implies$ $\exists I \subseteq \{1,..,n\}, |I| \geq 1 : \forall i \in I, p \xrightarrow{a_i} p_i$ *and* $p_i \vdash f_i$
- $f \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond} f_1, f_2 \ldots, s_n \implies$ $\nexists I \subseteq \{1,..,n\}, |I| > 1 : \forall i \in I, p \xrightarrow{a_i} p_i$ *and* $p_i \vdash f_i$
- $p \xrightarrow{a} p' \implies \exists A \subseteq Act, S \subseteq S_F, f' \in S_F :$ $(a, f') \in A \times S, (f, A \times S) \in \Box$ *or* $(f, A \times S) \in \diamond$*, and* $p' \vdash f'$

### 3.2. Generalized Extended Modal Transition Systems

A more general notion is actually needed if we want to model *multiple optionality*, that is, the fact that a product

**Table 1. Meaning of modalities**

| modality | MTS | DMTS | 1-MTS | EMTS | GEMTS |
|---|---|---|---|---|---|
| $\diamond$ (may) | at most 1-of-1 | at most n-of-n | at most 1-of-n | at most 1-of-n | at most k-of-n |
| $\Box$ (must) | at least 1-of-1 | at least 1-of-n | exactly 1-of-n | at least 1-of-n | at least k-of-n |

may have (at least, at most, exactly) $k$ of the $n$ choices proposed by the family.

It is a matter of discussion whether multiple optionality is really useful in family engineering. Actually, it is rarely the case that a functional requirement on the products of a family gives bounds on the number of the possible features present in a product. However, non functional requirements may do: for example, energy consumption or budget considerations may give an upper bound to the number of provided features, while marketing strategies may suggest a lower bound, under which the product looses its market. Also, upper and lower bounds may be used to define subfamilies on the basis of non functional aspects. In our running example, the requirement that "every product of the family delivers at least two different drinks" is not dictated by functional needs, but by marketing strategies.

In order to give a full range of behavioural models for variability types, we introduce the concept of Generalized Extended Modal Transition Systems, which is able to model multiple optionality.

**Definition 3.3** *A* Generalized Extended Modal Transition System *(GEMTS) is a quintuple* $(S, Act, s_0, \Box, \diamond)$*, where* $S$ *is a set of states,* $Act$ *is a set of actions,* $s_0 \in S$ *is the initial state,* $\Box \subseteq S \times 2^{Act \times S} \times N$ *is the* at least k -of- n *transition relation, and* $\diamond \subseteq S \times 2^{Act \times S} \times N$ *is the* at most k -of- n *transition relation.*

We write respectively: $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box_k} s_1, s_2 \ldots, s_n$ and $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond_k} s_1, s_2 \ldots, s_n$ to denote elements of the two relations, meaning that in the first case any product of the family should have at least $k$ of the $n$ transitions $s \xrightarrow{a_i} s_i$, while in the second case any product of the family should have at most $k$ of the $n$ transitions (that is, it can also have no transition from this set). Again, the number of the actions on the arrow must coincide with that of target states, and order counts as well, since each action is paired to the corresponding state. Note that $0 < k \leq n$ should always hold, otherwise the relation is meaningless.

The two defined transition relations have the following properties, that derive straightforwardly from the interpretation of a GEMTS as model of a family of LTSs:

- $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box_k} s_1, s_2 \ldots, s_n \implies$ $s \xrightarrow{a_2,\ldots,a_n}_{\Box_{k-1}} s_2 \ldots, s_n$
- $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box_k} s_1, s_2 \ldots, s_n$ *and* $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond_k} s_1, s_2 \ldots, s_n$ means: any product of the family should have *exactly* $k$ of the $n$ transitions $s \xrightarrow{a_i} s_i$; this defines the relation $\Box \cap \diamond$ as the relation *exactly k of n*.
- If we let $k$ to be equal to n, the may relation includes the must relation: $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box_n} s_1, s_2 \ldots, s_n \implies$ $s \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond_n} s_1, s_2 \ldots, s_n$. Inclusion does not hold if $k \neq n$.
- An EMTS can be defined as a GEMTS in which the relations $\Box$ and $\diamond$ are restricted to the constant value $k = 1$: this is obvious considering that the two relations can be read "at least 1-of-n" and "at most 1-of-n". Consequently, a MTS can be defined as a GEMTS in which the relations $\Box$ and $\diamond$ are restricted to the constant value $k = 1$, and where only singleton pairs from $Act \times S$ appear (that is, $n = 1$). Note therefore that $s \xrightarrow{a}_{\Box_1} s' \implies s \xrightarrow{a}_{\diamond_1} s'$ (if a transition is required is also possible, consistently with the definition of MTSs).

Note that the "at least 1-of-1" transitions coincide with the "exactly 1-of-1" ones and can be considered as the usual LTS transitions. For this reason, in the graphical notation we adopt for GEMTSs, in order to avoid notation overloading, we use the box and diamond symbols only in states corresponding to variation points: other states in the GEMTS with only "exactly 1-of-1" outgoing transitions are drawn as usual in LTS. This limited usage of the modality notation helps the reader to concentrate on variation points. For more simplicity, box and diamond symbols will be used without the number suffix when $n = 1$.

## 3.3. Comparison with other LTS-based family models

We now compare the expressiveness of our proposal with other variants of MTSs that have been proposed to model variability, namely the 1-selecting MTSs (1MTS) [26], and the Disjunctive Modal Transition Systems (DMTS) [19]. DMTSs and 1-MTSs are analogous to GEMTSs for the fact that they all use hypertransitions, that is, transition relations where the target is a set of states, in contrast with MTSs, in which the modal transition relations have only one target state. This allows modalities to predicate on a particular choice of a subset of transitions, rather than just on a single transition: note anyway that the possibility to have different (hyper)transitions from the same state is maintained in all the models. The meaning of the modalities that denote transition relations differs as shown in Table 1. Since the *exactly 1-of-n* relation is expressible as a conjunction of the *at most 1-of-n* and *at least 1-of-n*, and the *at most n-of-n* is a particular case of the *at most k-of-n*, from Table 1 it emerges that GEMTSs provide the most general modeling framework.

## 3.4. Deriving products from GEMTSs

A GEMTS defines a family of LTSs according to the following definition:

**Definition 3.4** *A LTS $P = (S_P, Act, \rightarrow, p_0)$ belongs to the family (GEMTS) $F = (S_F, Act, f_0, \Box, \diamond)$ (we say also* P is a product of F, *or* P conforms to F, *written $P \vdash F$ ) if and only if $p_0 \vdash f_0$, where:*

*$p \vdash f$ if and only if*

- $f \xrightarrow{a_1,a_2,\ldots,a_n}_{\Box_k} f_1, f_2 \ldots, f_n \implies \exists I \subseteq \{1,\ldots,n\}, k \leq |I| \leq n :$ $\forall i \in I, p \xrightarrow{a_i} p_i$ *and* $p_i \vdash f_i$
- $f \xrightarrow{a_1,a_2,\ldots,a_n}_{\diamond_k} f_1, f_2 \ldots, s_n \implies \nexists I \subseteq \{1,\ldots,n\}, k < |I| \leq n :$ $\forall i \in I, p \xrightarrow{a_i} p_i$ *and* $p_i \vdash f_i$
- $p \xrightarrow{a} p' \implies \exists k, A \subseteq Act, S \subseteq S_F, f' \in S_F :$ $(a, f') \in A \times S_F, (f, A \times S, k) \in \Box$ *or* $(f, A \times S, k) \in \diamond$, *and* $p' \vdash f'$

We can read the previous clauses as saying: a product of a family has at least (at most) $k$ of the $n$ transitions specified in the family; moreover, if a product performs an action, this should be found among the $n$ transitions specified for the family.

Notice that the second clause admits the case in which none of the transitions is present in the product. A GEMTS $F$ that does not admit any product (that is, there exist no LTS $P$ such that $P \vdash F$) is an inconsistent definition of family. Consider the GEMTS defined by the following relations, where $s$ is the initial state:

$$\begin{aligned}\Box = \diamond = \{&(s, \{(a_1, s_1), (a_2, s_2)\}),\\ &(s, \{(a_3, s_3), (a_2, s_2)\}),\\ &(s, \{(a_1 s_1), (a_3, s_3)\})\}.\end{aligned}$$

The equality of the $must$ and $may$ relations means that we have an "exactly one out of two" choice. But if we select $a_1$ from the first choice, this is selected also for the third choice, and these two selection prevent $a_2$ and $a_3$ to be selected, hence giving an inconsistency: this means that no LTS can be derived as a product from this GEMTS.

The requirements given in section 2.1 for our coffee machine can be now fully expressed by the GEMTS shown in Figure 7.

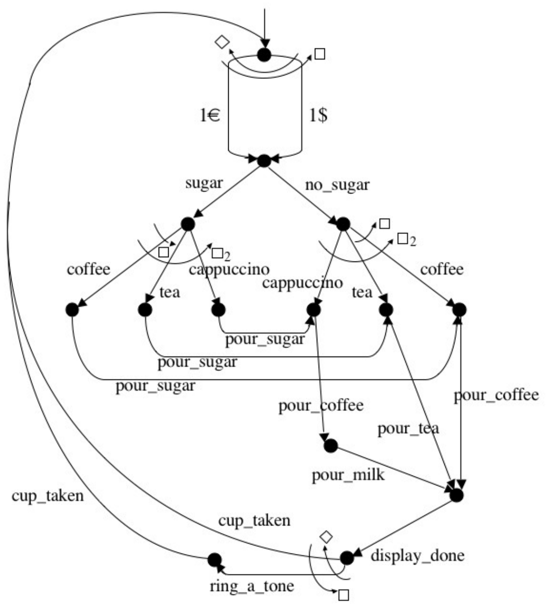

**Figure 7. Vending machines family**

The LTS of a product can be obtained from the family GEMTS by applying again the same derivation algorithm seen in section 2.1, in which the `select-transitions(s)` procedure enforces the selection of the user to respect the *at least k-of-n* and *at most k-of-n* relations applicable on the transitions outgoing from state `s`. Hence, the set of products derivable are, again, a subset of those derivable by the family LTS (note that the family LTS can be obtained from the family GEMTS by just removing the modality notations in the graphical syntax). Therefore, any derivable product is still simulated by the LTS family.

The US and European vending machines of Figures 2 and 3 are both derivable from the family GEMTS of Figure 7. Notice that now it is no more possible to derive the machine accepting both euro and dollar coins.

## 4. Dependency between variation points

In the LTS variants analysed so far, optionality and necessity of behavioural elements are defined for single variation points, with no links between choices to be done in different variation points. Consider again the vending machine example of section 2.1, where requirements are added due to some imaginary commercial policy that imposes that all the vending machines sold in the US have to ring a tone when the drink is ready, while all the ones sold in the EU have to be silent. If we want to consider this requirement on the whole family of vending machines, we need to enforce a dependency between the choices made at the two relevant variation points. There is no way to include such an additional constraint in a LTS-based definition, hence we necessarily should resort to some other formalism.

A first attempt to address the problem can be done however remaining inside the realm of formal tools defined over LTSs. Among the formal tools available to reason on LTSs, one of the most powerful is temporal logic, that can be used to define properties that LTSs should satisfy. Temporal logic can be used as well to express constraints for a family: such constraints may for example be used to relate the choices made in two variation points.

### 4.1. The logic ACTL

The ACTL temporal logic [9] is the action based version of CTL [10]. ACTL is a logic suitable to express properties of reactive systems whose behaviour is characterized by the actions they perform.

ACTL is defined as a logic of state formulas (denoted by $\phi$), in which a path quantifier prefixes an arbitrary path formula (denoted by $\gamma$). The syntax of the ACTL operators used in this paper and their informal meaning is reported in Table 2, while we refer to [9] for their formal semantics:the formal semantics of ACTL is given over LTSs, defining when a LTS $S$ satisfies a formula $\phi$ (written $S \models \phi$). The problem of verifying whether a LTS $S$ is a model for a formula $\phi$ is called *model-checking*, and is supported by efficient verification tools [6].

### 4.2. ACTL formuale as constraints

Temporal Logic formulae can be used to define constraints for a family:

**Definition 4.1** *A Constrained Family is defined as a pair $(F, \phi)$ of a GEMTS $F$ and an ACTL formula $\phi$; a LTS $P$ is member of the family $(F, \phi)$ if: $P \vdash F$ and $P \models \phi$.*

The constraints over vending machines respecting the policy that all vending machines sold in the US have to ring a tone when the drink is ready, while all the ones sold in the EU have to be silent, is given by the following ACTL formula:

$$AG[dollar]AF < ring\_a\_tone > true$$
$$\&AG[euro]AG \sim< ring\_a\_tone > true$$

which reads: whenever the model is able to perform a $dollar$ action, then it will eventually ring a tone; if instead the model is able to perform a $euro$ action, then it will never be able to ring a tone. Note that this formula does not tell where the variation points in the model are: this information is encoded instead by the GEMTS model and the formula represents an additional constraint. The products represented by the LTSs of Figures 2 and 3 satisfy such constraint, since the former rings a tone after inserting one dollar, hence satisfying the left hand of the conjunction, while the latter does not ring after inserting one euro, thus satisfying the right-hand-side of the conjunction: in both cases the left-hand-side is satisfied trivially by the absence of the guarding action. Other products that conform to the GEMTS of Figure 7, such as the one that rings a tone after inserting one euro, do not satisfy the constraint.

For a constrained family, derivation of a conforming product can be achieved by the derivation algorithm seen in section 3.4, followed by a run of a model-checker to check whether or not the constraints are satisfied. In principle, it would be possible to integrate a model-checking algorithm in the derivation algorithm, so that only transitions satisfying the constraints can be selected by the user; this issue is left to further investigation.

## 5. Open issues and conclusions

We have proposed an extended LTS-based formalism for the behavioural modeling of different types of variabilities in a family definition, and we have shown its higher generality w.r.t. other proposals in the literature.

However, a problem common to all the considered proposals, our own included, is related to the expression of dependency among variation points: we have hence proposed to address the problem by adding constraints expressed in the ACTL temporal logic. This is satisfactory in the particular example shown, where related variation points can be easily identified by the actions performed at the states encoding variation points. This is not always the case, and therefore we need a mean to express general constraints such that, in deriving a product, "if a given choice is done at a variation point $a$, a consequent choice is enforced at the variation point $b$". Hence, we need to include in the model some orthogonal, non-functional, information used only to constrain the functional behaviour of the admitted products.

At this regard, a notable LTS-based modeling framework for behaviour of product families is the VLTS (Variant LTS) one, proposed in [24], where possible transitions are labeled with a guard predicating on the values of external variables: each product defines an assignment to such variables, and hence different variation points where transitions have labels referring to the same variable are not independent. For example, the $dollar$ transition and the

**Table 2. Syntax and informal semantics of ACTL**

| | | | |
|---|---|---|---|
| $\phi$ | $::=$ | $true$ | "any behaviour is possible" |
| | | $\sim \phi$ | "$\phi$ is impossible" |
| | | $\phi \ \& \ \phi'$ | "$\phi$ and $\phi'$" |
| | | $E\gamma$ | "there exists a possible execution in which $\gamma$" |
| | | $A\gamma$ | "for each of the possible executions $\gamma$" |
| | | $[a]\phi$ | " for all next states reachable with $a$, $\phi$ is true" |
| | | $< a > \phi$ | "there exists a next state reachable with $a$, in which $\phi$" |
| $\gamma$ | $::=$ | $F\phi'$ | "there exists a future state in which $\phi$ holds" |
| | | $G\phi'$ | "in any future state $\phi$ holds$\phi$" |

$ring\_a\_tone$ transition in the vending machine examples would both be guarded by a predicate saying that the variable $nation$ is assigned the value $US$.

Such predicates resemble the use of tags to label different choices in a variation point, which is a common strategy in product family engineering: as an example, tags were introduced in Use Cases to define the Product Line Use Case notation [4]. We have shown in [5] how those tags can be used to verify the compliance of a product to the definition of a family, with a mechanism inspired by [22], that could be applied to VLTSs as well.

In the VLTS approach the mechanism of verification of constraints is hence separated by the mechanisms used to reason about conformance, equivalence, and satisfaction of logic formulae that employs classic LTS-based verification techniques. The idea behind our GEMTS modeling framework was instead to propose a unique verification framework. The work [20] proposes the so called "color-blind" Transition Systems, in which the definition of constraints is delegated to the immersion of the family specification in an environment defined with the same formalism, which constrains the behaviour of the family transition system. Our opinion is that an approach based on temporal logics is more suitable to express constraints at an abstract level. For this reason we are going to investigate more powerful models and logics, such as those defined in [3].

The adoption of temporal logic gives the added benefit, not addressed in this paper, to reason formally over family specifications. In particular, the availability of efficient verification tools is attractive, since it enables early formal analysis to be conducted once for all at the family level, rather than repeated on the single products. For this purpose, we need to know which classes of properties that are enjoyed by family definitions are preserved in the derivation process. In the framework considered in this paper, the fact that products are related to family definitions by simulation allows to exploit the result that the universal fragment of ACTL is preserved by the simulation relation. This amounts to say that a universal property satisfied by a GEMTS family definition is inherited by any product LTS derived with the proposed algorithm.

More elaborate product conformance relations have been proposed, such as the refinement notions between MTSs of [12]: the authors allow for unobservable actions to represent internal behaviour still to be refined at the level of the family. Specific notions of refinement (*weak* and *branching* refinement) take into proper account unobservable actions: the product MTS is a refinement of the family MTS, allowing new states to be introduced in a product with respect to the family. Such a definition of the conformance relation breaks the simplicity of the simulation relation we have considered throughout our paper, which is based on the assumption that the states of a product are a subset of the states of the family. The refinement notions of [12] can nevertheless be applied to the LTS variants introduced in this paper as well, and this is considered as a future research issue. Further study is needed to characterize properties that are preserved by such conformance relations. Investigating more powerful logics can be necessary, and this should also take into account the possibility of introducing more powerful models to deal with dependency between variation points.

On the other hand, we are fully aware of the limitations of LTS-based modeling w.r.t. the richer formalisms that are preferred for the actual description of the behaviour of the system, which include, as in the case of UML State Diagrams, the possibility to associate actions and predicates to the state, as well as guards on the transitions. Once again, the proposed LTS-based notions are not intended to be used as such by engineers to describe product families, but rather to give basic modeling concepts on which to base the verification activities.
We leave to further work is a deeper analysis of the verification issues: here we have only sketched the possibility that properties proved to be verified by the family are preserved by the derivation of products. A more extensive treatment of this topic is necessary, also to best motivate the need for the proposed LTS variants.

**Acknowledgments** We wish to thank Maurice ter Beek for his valuable support.

## References

[1] J. Bayer, S. Gerard, O.Haugen, J.Mansell, B. Moller-Pedersen, J. Oldevik, P. Tessier, J.P. Thibault, T. Widen, Consolidated Product Line Variability Modeling. Chapter 6 of [15].

[2] D. S. Batory, Feature Models, Grammars, and Propositional Formulas. SPLC 2005, Rennes, France, September 26-29, 2005. LNCS 3714.

[3] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti, An action/state-based model-checking approach for the analysis of communication protocols for Service-Oriented Applications, FMICS 2007, Berlin, July 2007. LNCS 4916.

[4] A. Bertolino, A. Fantechi, S. Gnesi, G. Lami, A. Maccari, Use Case Description of Requirements for Product Lines, REPL'02, Essen, Germany, Sept. 2002.

[5] A. Bertolino, A. Fantechi, S. Gnesi, G. Lami, Product Line Use Cases, Chapter 11 of [15].

[6] E. Clarke, O. Grumberg, and D. Peled, Model Checking. MIT PRESS, 1999.

[7] P. C. Clements and L. Northrop, Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley, August 2001.

[8] K. Czarnecki, K., U.W. Eisenecker. Generative Programming: Methods, Tools, and Applications, Addison- Wesley, Boston, MA, 2000.

[9] R. De Nicola, F. W. Vaandrager, Action versus State based Logics for Transition Systems. Proceedings Ecole de Printemps on Semantics of Concurrency. LNCS 469, pp. 407-419, 1990.

[10] E. A. Emerson, J. Halpern, ″Sometime″ and ″Not never″ revisited: On branching versus linear time temporal logic. *JACM* 33:151-178, 1986.

[11] A. Fantechi, S. Gnesi, A behavioural model for product families . In Proceedings ESEC/FSE 2007, pp. 525-528, ACM, 2007.

[12] D. Fischbein, S. Uchitel, Sebastian, V. Braberman, A Foundation for Behavioural Conformance in Software Product Line Architectures, 2nd ROSATEA Workshop, Portland, Maine, 2006

[13] G. Halmans, K. Pohl, Communicating the Variability of a Software-Product Family to Customers, Journal of Software and Systems Modeling, Springer, 2003

[14] M. Jazayeri, A. Ran, F. van der Linden, Software Architecture for Product Families: Principles and Practice, Addison-Wesley, London, 1998.

[15] T. Kakola, J.C., Duenas (Ed.), Software Product Lines: Research Issues in Engineering and Management, Springer-Verlag, ISBN: 3540332529, Oct. 2006.

[16] K. Kang, S. Choen, J. Hess, W. Novak, S. Peterson. Feature Oriented Domain Analysis (FODA) Feasibility Study. Technical report CMU/SEI-90-TR-21, Carnegie Mellon University, Nov. 1990.

[17] K.G. Larsen, U. Nyman, A. Wasowski. Modal I/O Automata for Interface and Product Line Theories. ESOP'07, LNCS 4421, Springer, 2007.

[18] K. Larsen and B. Thomsen, A Modal Process Logic. In LICS88, IEEE CS, pp. 203-210.

[19] K.G. Larsen, L. Xinxin, Equation solving using modal transition systems, LICS 1990, IEEE CS, pp. 108-117.

[20] K. G. Larsen, U. Nyman, A. Wasowski, Modeling software product lines using color-blind transition systems. STTT 9(5-6): 471-487 (2007).

[21] M. S. Lund, K. Stølen, A Fully General Operational Semantics for UML 2.0 Sequence Diagrams with Potential and Mandatory Choice. FM'2006, LNCS 4085, 380-395.

[22] M. Mannion, J. Camara, Theorem Proving for Product Line Model Verification, Fifth International Workshop on Product Family Engineering, PFE-5, Siena 4-6 November, 2003, LNCS 3014.

[23] R. Milner. An Algebraic Definition of Simulation Between Programs. IJCAI 1971, London, UK, William Kaufmann, 1971.

[24] H. Muccini, A. Bucchiarone, Formal Behavioral Specification of a Product Line Architecture, Dipartimento di Informatica, University of L'Aquila, TR 014/2004.

[25] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering - Foundations, Principles, and Techniques, Springer, Heidelberg, July 2005.

[26] H. Schmidt, H. Fecher, Comparing Disjunctive Modal Transition Systems with an One-Selecting Variant, NWPT'06, Reykjavik, Iceland, October 2006

[27] T. Ziadi, J.M. Jezequel, Product Line Engineering with the UML: Deriving Products, Chapter 15 of [15].