Deontic logics for modeling behavioural variability

Deontic Logics for Modeling Behavioural Variability
Research in progress[*]

P. Asirelli, M. H. ter Beek, S. Gnesi
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", ISTI–CNR
Via G. Moruzzi 1, I-56124 PISA (Italy)
{asirelli,terbeek,gnesi}@isti.cnr.it

A. Fantechi
DSI–Università di Firenze and ISTI–CNR, Pisa
Via S. Marta 3, I-50139 FIRENZE (Italy)
fantechi@dsi.unifi.it

Abstract

We discuss the application of deontic logics to the modeling of variabilities in product family descriptions. Deontic logics make it possible to express concepts like permission and obligation. As a first result of this line of research, we show how a Modal Transition System, a model that has recently been proposed as an expressive way to deal with behavioural variability in product families, can be completely characterized with deontic logic formulae. We moreover show some exemplary properties that can consequently be proved for product families. These preliminary results pave the way to a wider application of deontic logics to specifiy and verify variability in product families.

1 Introduction

A description of a Product Family (PF) is usually composed of a constant part and a variable part. The first part describes aspects that are common to all products of the family, whereas the second part represents those aspects, called *variabilities*, that will be used to differentiate one product from another. The modeling of variability has been extensively studied in the literature, especially that concerning *Feature modeling* [2, 6, 17]. 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.

[*]Funded by the Italian project D-ASAP (MIUR–PRIN 2007) and by the RSTL project XXL of the Italian National Research Council (CNR).

In this paper we are interested in the modeling of behavioural variability, that is, how the products of a family differ in their ability to respond to events in time: this is an aspect that the referenced techniques do not typically focus on. Many notations and description techniques have been recently proposed for this purpose, such as variants of UML state diagrams [3, 25] or variants of UML sequence diagrams, for example STAIRS [20]; another proposal can be found in [24], where UML state diagrams and sequence diagrams have been enriched with aside notations to describe variation points. At this regard, we rather prefer to look for an expressive modeling formalism for families based on the choice of a basic behavioural model, namely Labelled Transition Systems (LTSs), which is one of the most popular formal frameworks for modeling and reasoning about the behaviour of a system. Modal Transition Systems (MTSs) have been proposed, in several variants, to model a family of such LTSs [18, 13, 8]: in this way it is possible to embed in a single model the behaviour of a family of products that share the basic structure of states and transitions, transitions which however can be seen as mandatory or possible for the products of the family.

Deontic logics [1] have become very popular in computer science in the last few decades to formalize descriptional and behavioural aspects of systems. This is mainly because they provide a natural way to formalize concepts like violation, obligation, permission, and prohibition. Intuitively, they permit one to distinguish between correct (normative) states and actions on the one hand and non-compliant states and actions on the other hand. This makes deontic logics a natural candidate for expressing the variability of a family of products. Recently, a Propositional Deontic Logic (PDL) capable of expressing the permitted

behaviour of a system has been proposed [5]. We want to study in detail the application of this kind of logics to the modeling of behavioural variability. Indeed, PDL appears to be a good candidate to express in a unique framework both behavioural aspects, by means of standard branching-time logic operators, and constraints over the product of a family, which usually require a separate expression in a first-order logic (as seen in [2, 9, 21]).

In this paper, we want to focus our attention on the ability of a logic of finitely characterizing the complete behaviour of a system, that is, given a MTS $\mathcal{A}$, we look for a formula of the logic that is satisfied by all and only those Labelled Transition Systems that can be derived from $\mathcal{A}$ As a first result in this direction, the main contribution of this paper is that we are able to finitely characterize finite state MTSs, using a deontic logic; to this aim, we associate a logical formula (called *characteristic formula* [11, 12, 23]) to each state of the MTS. Consequently, every LTS of the family defined by the MTS satisfies the formula, and no LTS outside the family satisfies it. In this way we establish a link between a common model of behavioural variability and PDL. The deontic logic proposed in this paper is able to describe, in a faithful way, the behaviour of systems modelled by finite state MTSs. Our work can serve as a basis to develop a full logical framework to express behavioural variability and to build verification tools employing efficient model-checking algorithms.

2 Labelled Transition Systems

As said before, a basic element of our research is the concept of a Labelled Transition System, of which we define several variants.

Definition 2.1 (Labelled Transition System) *A* Labelled Transition System *(LTS) is a quadruple* $(Q, q_0, Act, \rightarrow)$*, in which*

- Q *is a set of states;*
- $q_0 \in Q$ *is the initial state;*
- *Act is a finite set of observable events (actions);*
- $\rightarrow \subseteq Q \times Act \times Q$ *is the transition relation; instead of* $(q, \alpha, q') \in \rightarrow$ *we will often write* $q \xrightarrow{\alpha} q'$*.*

Definition 2.2 (Modal Transition System) *A* Modal Transition System *(MTS) is a quintuple* $(S, s_0, Act, \rightarrow_\Box, \rightarrow_\Diamond)$ *such that* $(S, s_0, Act, \rightarrow_\Box \cup \rightarrow_\Diamond)$ *is a LTS. A MTS has two distinct transition relations: the* must *transition relation* $\rightarrow_\Box$ *expresses* required *transitions, while the* may *transition relation* $\rightarrow_\Diamond$ *expresses* possible *transitions.*

A MTS defines a family of LTSs, in the sense that each LTS $P = (S_P, p_0, Act, \rightarrow)$ of the family can be obtained from the MTS $F = (S_F, f_0, Act, \rightarrow_\Box, \rightarrow_\Diamond)$ by considering its transition relation $\rightarrow$ to be $\rightarrow_\Box \cup R$, with $R \subset \rightarrow_\Diamond$, and pruning the states that are not reachable from its initial state p_0. The "P *is a product of* F" relation below, also called "conformance" relation, links a MTS F representing a family with a LTS P representing a product.

Definition 2.3 (Conformance relation) *We say that* P is a product of F*, denoted by* $P \vdash F$*, if and only if* $p_0 \vdash s_0$*, where* $p \vdash f$ *if and only if*

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

Another extension of LTSs is obtained by labelling its states with atomic propositions, leading to the concept of doubly-labelled transition systems [7].

Definition 2.4 (Doubly-Labelled Transition System) *A* Doubly-Labelled Transition System *(L^2TS) is a quintuple* $(Q, q_0, Act, \rightarrow, AP, L)$*, in which*

- $(Q, q_0, Act, \rightarrow)$ *is a LTS;*
- *AP is a set of atomic propositions;*
- $L : Q \longrightarrow 2^{AP}$ *is a labelling function that associates a subset of AP to each state of the LTS.*

3 A deontic logic

Deontic logics are an active field of research in formal logic for many years now. Many different deontic logic systems have been developed and in particular the use of modal systems has had a lot of success in the deontic community [1]. The way such logics formalize concepts such as violation, obligation, permission and prohibition is very useful for system specification, where these concepts arise naturally. In particular, deontic logics seem to be very useful to formalize product families specifications, since they allow one to capture the notion of possible and compulsory features.

Our starting point is the Propositional Deontic Logic (PDL) defined in [5]. PDL is able to express both the evolution in time of a system by means of an action, and the fact that certain actions are permitted or not in a given state. The original definition considers actions from a set *Act*, each action producing a set of events from a set E. The set of events produced by an action $\alpha \in Act$ is named $I(\alpha)$. The logic we propose in this paper is a temporal extension of PDL,

in a style reminiscent of the extension proposed in [5]. The syntax of the logic is:

$$\phi ::= tt \mid p \mid \neg\phi \mid \phi \wedge \phi' \mid A\pi \mid E\pi \mid [\alpha]\phi \mid P(\alpha) \mid P_w(\alpha)$$
$$\pi ::= \phi \; U \; \phi'$$

As usual, *ff* abbreviates $\neg tt$, $\phi \vee \phi'$ abbreviates $\neg(\neg\phi \wedge \neg\phi')$, and $\phi \implies \phi'$ abbreviates $\neg\phi \vee \phi'$. Moreover, $EF\phi$ abbreviates $E\,(tt\; U\; \phi)$ and $AG\phi$ abbreviates $\neg EF\neg\phi$. Finally, the informal meaning of the three non-conventional modalities, explained below in more detail, is:

- $[\alpha]\phi$: after any possible execution of α, ϕ holds;
- $P(\alpha)$: every way of executing α is allowed;
- $P_w(\alpha)$: some way of executing α is allowed.

The first of these three modalities thus provides the possibility to express evolution, while the other two provide the possibility to express (weak) permission. Furthermore, $\langle\alpha\rangle\phi$ abbreviates $\neg[\alpha]\neg\phi$.

The two variants of permission are rather common in the literature on deontic logic. The operator $P(\alpha)$ tells whether or not an action α is allowed to be performed. It can be called a strong permission since it requires that every way of performing α has to be allowed (e.g. if we were to say that *driving* is allowed, it would mean that also *driving while drinking beer* is allowed). Not surprisingly, permission has been a polemical notion since the very beginning of deontic logic. Some have proposed a weak version [22] in which to be allowed to perform an action means that this action is allowed only in some contexts. We stick to [5] and use both notions of permission. The latter is denoted by the operator $P_w(\alpha)$, which must be read as α is weakly allowed. The two versions differ in their properties (see [5] for details).

In [5] both variants of permission are used to define obligation $O(\alpha)$ as $P(\alpha) \wedge \neg P_w(\neg\alpha)$, i.e. α is obligated if and only if it is strongly permitted and no other action is weakly allowed. This definition avoids Ross's paradox $O(\alpha) \implies O(\alpha \vee \alpha')$, which can be read as "if you are obliged to send a letter, then you are obliged to send it or burn it".

The formal semantics of our logic is given below by means of an interpretation over L^{2}TS, mimicking the original semantics of PDL in [5]. To this aim, the L^{2}TS used as an interpretation structure is defined as a sixtuple $(W, w_0, E, \rightarrow, AP \cup E, L)$, in which the transitions are labelled over the set of events E and the states (corresponding to the *worlds* of the standard interpretation) are labelled with atomic propositions as well as with the events allowed in the states. To this purpose, we also use a relation $P \subseteq W \times E$ to denote which events are permitted in which world, with the understanding that $P(w, e)$ if and only if $e \in L(w)$.

Definition 3.1 (Semantics) *The satisfaction relation of our deontic logic is defined as follows:*

- $w \models tt$ *always holds;*
- $w \models p$ *iff* $p \in L(w)$*;*
- $w \models \neg\phi$ *iff not* $w \models \phi$*;*
- $w \models \phi \wedge \phi'$ *iff* $w \models \phi$ *and* $w \models \phi'$*;*
- $w \models A\pi$ *iff* $\sigma \models \pi$ *for all paths* σ *that start with state* w*;*
- $w \models E\pi$ *iff there exists a path* σ *that starts with state* w *such that* $\sigma \models \pi$*;*
- $w \models [\alpha]\phi$ *iff* $\forall e \in \mathcal{I}(\alpha) : w \xrightarrow{e} w'$ *implies* $w' \models \phi$*;*
- $w \models P(\alpha)$ *iff* $\forall e \in \mathcal{I}(\alpha) : P(w, e)$ *holds;*
- $w \models P_w(\alpha)$ *iff* $\exists e \in \mathcal{I}(\alpha) : P(w, e)$ *holds;*
- $\sigma \models [\phi \; U \; \phi']$ *iff there exists a state* s_j*, for some* $j \geq 0$*, on the path* σ *such that for all states* s_k*, with* $j \leq k$*,* $s_k \models \phi'$ *while for all states* s_i*, with* $0 \leq i < j$*,* $s_i \models \phi$*.*

4 A deontic characteristic formula for MTSs

In this section, we show how a unique deontic logic formula can completely characterize a family of LTSs by separating the structure of the LTS (taken care of by the box fomulae) from the optional/mandatory nature of the transitions (taken care of by the permission formulae). Since a MTS is a compact expression of a family of LTSs, this is equivalent to saying that we are able to characterize a MTS with a deontic logic formula. The result we show here is rather preliminary, in the sense that it currently needs the following simplifying assumptions, but it nevertheless shows the potentiality of our deontic logic.

- First, we adopt a strict interpretation to the permitted events that label the transitions of a L^{2}TS, namely we assume that $w \xrightarrow{e}$ implies $P(w, e)$, that is, only permitted actions are executed.
- We then assume, for any action α, that $I(\alpha) = \{e_\alpha\}$, that is, actions and events are indistinguishable.
- We also assume that a MTS defines the family of those LTSs that are derived from a corresponding family of L^{2}TSs, simply ignoring the predicates on the states.
- Last, we use a simpler form of MTSs, in which transitions leaving the same state are either all box transitions or all diamond transitions. As we show next, this assumption allows us to distinguish box states and diamond states, and have a single transition relation.

Definition 4.1 (Alternative def. MTS) *A MTS is a quintuple* $(BS, DS, s_0, Act, \rightarrow)$ *such that* $(BS \cup DS, s_0, Act, \rightarrow)$ *is a LTS and* $BS \cap DS = \varnothing$. *A MTS has two distinct sets of states: the* box *states* BS *and the* diamond *states* DS.

At this point, we define the characteristic formula $\mathcal{FC}(M)$ of a (simple) MTS $M = (BS, DS, s_0, Act, \rightarrow)$ as $\mathcal{FC}(s_0)$, where

$$\mathcal{FC}(s) = \begin{cases} (\bigvee_i P_w(\alpha_i)) \wedge ((\bigwedge_i [\alpha_i]\mathcal{FC}(s_i)) & \text{if } s \in DS \\ (\bigwedge_i O(\alpha_i)) \wedge ((\bigwedge_i [\alpha_i]\mathcal{FC}(s_i)) & \text{if } s \in BS \\ \text{and } \forall i\colon s \xrightarrow{e_i} s_i \text{ with } I(\alpha_i) = \{e_i\} \end{cases}$$

If we define the characteristic formula in an equational form using the expressions above, we obtain one equation for each state of the MTS, and the equations have a number of terms equal to two times the number of transitions leaving the relevant state. An attempt to write a single characteristic formula gives a formula exponential in size with respect to the number of states, and needs some form of fixed point expression for expressing cycles in the MTS (see [12]).

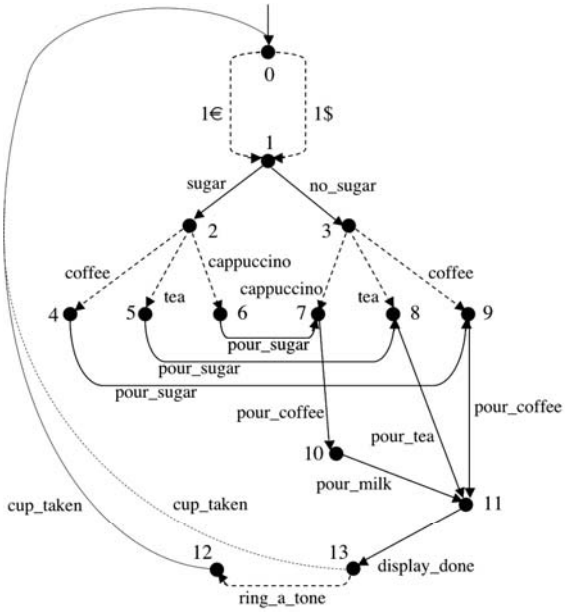

Figure 1. A MTS modeling a product family.

5 An example

Let us consider the example introduced in [8], that is, a family of coffee machines represented by the MTS depicted in Fig. 1, which allows products to differ for the two different currencies accepted, for the three drinks delivered and for the presence of a ring tone after delivery. In the figure, solid arcs are *required* transitions and dashed arcs are *possible* transitions, that is, states with outgoing solid arcs belong to BS, and states with outgoing dashed arcs belong to DS.

The characteristic formula in equational form is given by the following set of equations:

$$\begin{aligned}
\phi_0 &= (P_w(1€) \vee P_w(1\$)) \wedge ([1€]\phi_1 \wedge [1\$]\phi_1) \\
\phi_1 &= (O(\text{sugar}) \wedge O(\text{no_sugar})) \\
&\quad \wedge ([\text{sugar}]\phi_2 \wedge [\text{no_sugar}]\phi_3) \\
\phi_2 &= (P_w(\text{coffee}) \vee P_w(\text{cappuccino}) \vee P_w(\text{tea})) \\
&\quad \wedge ([\text{coffee}]\phi_4 \wedge [\text{cappuccino}]\phi_5 \wedge [\text{tea}]\phi_6) \\
\phi_3 &= (P_w(\text{coffee}) \vee P_w(\text{cappuccino}) \vee P_w(\text{tea})) \\
&\quad \wedge ([\text{coffee}]\phi_7 \wedge [\text{cappuccino}]\phi_8 \wedge [\text{tea}]\phi_9) \\
\phi_4 &= O(\text{pour_sugar}) \wedge [\text{pour_sugar}]\phi_7 \\
\phi_5 &= O(\text{pour_sugar}) \wedge [\text{pour_sugar}]\phi_8 \\
\phi_6 &= O(\text{pour_sugar}) \wedge [\text{pour_sugar}]\phi_9 \\
\phi_7 &= O(\text{pour_coffee}) \wedge [\text{pour_coffee}]\phi_{10} \\
\phi_8 &= O(\text{pour_tea}) \wedge [\text{pour_tea}]\phi_{11} \\
\phi_9 &= O(\text{pour_coffee}) \wedge [\text{pour_coffee}]\phi_{11} \\
\phi_{10} &= O(\text{pour_milk}) \wedge [\text{pour_milk}]\phi_{11} \\
\phi_{11} &= O(\text{display_done}) \wedge [\text{display_done}]\phi_{12} \\
\phi_{12} &= (P_w(\text{cup_taken}) \vee P_w(\text{ring_a_tone})) \\
&\quad \wedge ([\text{cup_taken}]\phi_0 \wedge [\text{ring_a_tone}]\phi_{13}) \\
\phi_{13} &= O(\text{cup_taken}) \wedge [\text{cup_taken}]\phi_0
\end{aligned}$$

Note that the characteristic formula given above does not allow, from any state of the considered MTS, to derive a LTS such that the corresponding state has no outgoing transitions, even in the case of diamond states.

The characteristic formula of a MTS implies any other property which is satisfied by the MTS, and can thus serve as a basis for the logical verification over MTSs. Actually, this approach is not as efficient as model-checking ones, but the definition of the characteristic formula may serve as a basis for a deeper study of the application of deontic logics to the verification of properties of families of products.

We now show two exemplary formulae that use deontic operators to formalize properties of the products derived by the family of coffee machines represented by the MTS depicted in Fig. 1.

1. The family permits to derive a product in which it is permitted to get a coffee with 1€:

$$P_w(1€) \implies [1€]\, E\, (tt\, U\, P_w(\text{coffee}))$$

2. The family obliges every product to provide the possibility to ask for sugar:

$$A\, (tt\, U\, O(\text{sugar}))$$

Notice that the deontic operators predicate on the relations between the products and the family, although they are defined on particular states of their behaviour.

It can be seen that the formulae above can be derived, using the axiom system given in [5], from the characteristic formula of the MTS of Fig. 1. This proves that the above properties are actually verified for the coffee machines that belong to the family described by the MTS.

6 Conclusions

We have shown how deontic logics can express the variability of a family, in particular by showing the capability of a deontic logic formula to finitely characterize a finite state Modal Transition System, a formalism proposed to capture the behavioural variability of a family. The logical framework has allowed us to prove simple behavioural properties of the example MTS shown.

These results, although very preliminary, lay the basis of further research in this direction, in which we aim at exploiting the full power of PDL for what concerns the expression of behavioural variability: the presence of CTL-like temporal operators allows more complex behavioural properties to be defined and therefore more expressive descriptions of behavioural variability to be supported. In particular, the dependency between variation points could be addressed in a uniform setting. Another interesting direction is the adoption of model-checking techniques to build efficient verification tools aimed at verifying, on the family definition, properties which are inherited by all the products of the family.

It would also be interesting to have a look at the variability of dynamic models in the large, taking into account both the problem of modelling variability in business process models and that of using goal modelling—which intrinsically includes variability—to model variable processes.

Finally, it remains to study to what degree the complexity of the proposed logic and verification framework can be hidden from the end user, or be made more user friendly, in order to support developers in practice.

References

[1] L. Åqvist, Deontic Logic. In D. Gabbay and F. Guenthner (Eds.): Handbook of Philosophical Logic (2nd Edition), Volume 8. Kluwer Academic, Dordrecht, 2002, 147-264.

[2] D.S. Batory, Feature Models, Grammars, and Propositional Formulas. In J.H. Obbink and K. Pohl (Eds.): *Proceedings Software Product Lines Conference (SPLC'05)*, LNCS 3714, 2005, 7–20.

[3] J. Bayer, S. Gerard, O. Haugen, J. Mansell, B. Møller-Pedersen, J. Oldevik, P. Tessier, J.-P. Thibault and T. Widen, Consolidated Product Line Variability Modeling. Chapter 6 of [16], 2006, 195–241.

[4] 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. In S. Leue and P. Merino (Eds.): *Proceedings Formal Methods for Industrial Critical Systems (FMICS'07)*, LNCS 4916, Springer, 2008, 133–148.

[5] P.F. Castro and T.S.E. Maibaum, A Complete and Compact Propositional Deontic Logic. In C.B. Jones, Zh. Liu and J. Woodcock (Eds.): *International Colloquium Theoretical Aspects of Computing (ICTAC'07)*, LNCS 4711, Springer, 2007, 109–123.

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

[7] R. De Nicola and F.W. Vaandrager, Three Logics for Branching Bisimulation. *Journal of the ACM* 42, 2 (1995), 458–487.

[8] A. Fantechi and S. Gnesi, Formal Modeling for Product Families Engineering. In *Proceedings Software Product Lines Conference (SPLC'08)*, IEEE, 2008, 193–202.

[9] A. Fantechi, S. Gnesi, G. Lami and E. Nesti, A Methodology for the Derivation and Verification of Use Cases for Product Lines. In R.L. Nord (Ed.): *Proceedings Software Product Lines Conference (SPLC'04)*, LNCS 3154, Springer, 2004, 255–265.

[10] A. Fantechi, S. Gnesi, A. Lapadula, F. Mazzanti, R. Pugliese and F. Tiezzi, A model checking approach for verifying COWS specifications. In J.L. Fiadeiro and P. Inverardi (Eds.): *Proceedings Fundamental Approaches to Software Engineering (FASE'08)*, LNCS 4961, Springer, 2008, 230–245.

[11] A. Fantechi, S. Gnesi and G. Ristori, Compositionality and Bisimulation: A Negative Result. *Information Processing Letters* 39, 2 (1991), 109–114.

[12] A. Fantechi, S. Gnesi and G. Ristori, Modeling Transition Systems within an Action Based Logic. Technical Report B4-49-12, IEI–CNR, Dec. 1995.

[13] D. Fischbein, S. Uchitel and V.A. Braberman, A Foundation for Behavioural Conformance in Software Product Line Architectures. In R.M. Hierons and H.

Muccini (Eds.): *Proceedings Role of Software Architecture for Testing and Analysis (ROSATEA'06)*, ACM, 2006, 39–48.

[14] A. Gruler, M. Leucker and K.D. Scheidemann, Modeling and Model Checking Software Product Lines. In G. Barthe and F.S. de Boer (Eds.): *Proceedings Formal Methods for Open Object-Based Distributed Systems (FMOODS'08)*, LNCS 5051, Springer, 2008, 113–131.

[15] G. Halmans and K. Pohl, Communicating the Variability of a Software-Product Family to Customers, *Software and Systems Modeling* 2, 1 (2003), 15–36.

[16] T. Käkölä and J.C. Dueñas (Eds.): *Software Product Lines—Research Issues in Engineering and Management*, Springer, Berlin, 2006.

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

[18] K.G. Larsen, U. Nyman and A. Wąsowski, Modal I/O Automata for Interface and Product Line Theories. In R. De Nicola (Ed.): *Proceedings European Symposium on Programming Languages and Systems (ESOP'07)*, LNCS 4421, Springer, 2007, 64–79.

[19] K.G. Larsen, U. Nyman and A. Wąsowski, Modeling software product lines using color-blind transition systems. *International Journal on Software Tools for Technology Transfer* 9, 5–6 (2007), 471–487.

[20] M.S. Lund and K. Stølen, A Fully General Operational Semantics for UML 2.0 Sequence Diagrams with Potential and Mandatory Choice. In J. Misra, T. Nipkow and E. Sekerinski (Eds.): *Proceedings Formal Methods (FM'06)*, LNCS 4085, Springer, 2006, 380–395.

[21] M. Mannion and J. Camara, Theorem Proving for Product Line Model Verification. In F. van der Linden (Ed.): *Proceedings Software Product-Family Engineering (PFE'03)*, LNCS 3014, Springer, 2004, 211–224.

[22] J.-J.Ch. Meyer, A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic. *Notre Dame Journal of Formal Logic* 29, 1 (1988), 109–136.

[23] B. Steffen, Characteristic Formulae. In G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi Della Rocca (Eds.): *Proceedings International Colloquium Automata, Languages and Programming (ICALP'89)*, LNCS 372, Springer, 1989, 723–732.

[24] K. Pohl, G. Böckle and F. van der Linden, *Software Product Line Engineering—Foundations, Principles, and Techniques*, Springer, Berlin, 2005.

[25] T. Ziadi and J.-M. Jézéquel, Product Line Engineering with the UML: Deriving Products. Chapter 15 of [16], 2006, 557–586.