MTS were first recognised as a suitable behavioural model for describing product lines in [19], which provided an algorithm to check the conformance of product behaviour against that of the product family. Subsequent extensions involving notions from interface automata and I/O automata were defined in [23] and [24], respectively. Another line of research led to MTS with an associated set of variability constraints expressed over actions and a dedicated variability model checker that allows one to verify a property for a family and conclude the result to hold for all its products [11, 13].

Compared to FMCA, none of these models can explicitly handle *dynamic* product lines, a characteristic FMCA inherits from [5, 6].

Furthermore, we tackled the problem of synthesising the *mpc* of a family of service contracts fulfilling all feature constraints, all necessary service requests and the maximal number of permitted service requests. Building on an earlier model [8], permitted and necessary transitions are interpreted as controllable and uncontrollable transitions in SCT [16]. In this paper, we partitioned necessary transitions based on their degree of controllability, thus enriching the synthesis algorithm by considering necessary transitions whose controllability can be altered due to non-local information.

SCT was previously applied to Software Product Line Engineering in [14], where the `CIF 3` toolset was used to synthesise all valid products of a product line composed of behavioural components and requirements modelled as automata. Based on the synthesis of the *mpc* in [8], our approach to synthesise a family of services does not consider all actions to be controllable, as in [14], but considers increasing levels of uncontrollability (from urgent to lazy requests). The information related to the specific requirements of each product (required and forbidden features) is also integrated into the synthesis algorithm.

Moreover, whilst the number of products is in general exponential in the number of features, the organisation of the family's products (and their *mpc*) into a partial order makes our approach more scalable. As a result, the obtained *mpc* of the family of services can be synthesised from only a subset of its products, whereas other approaches require to synthesise the *mpc* of each single product.

Software architectural and adaptation patterns for DSPL are studied in [21], which are triggered automatically by monitoring executions. An orchestration is assumed to be derived automatically, whilst we focus on how such an orchestration can be synthesised. Moreover, services are described by means of interfaces (i.e. connectors) and are in correspondence with features, whilst we provide their behavioural representation and we identify features as service actions, to be activated according to specific products. FMCA could be used as the underlying formalism for deploying and updating applications, as well as automatically synthesising a well-behaving orchestration of services.

Finally, the presented theory has been implemented in a prototypical tool, which was used to compute all examples given throughout the paper. It remains to compare our approach with other synthesis algorithms and to quantify how well it scales. To this aim, we would like to model and analyse a real world service-based application, as was done in [4] for a system without variability.

Another direction for future work is to enhance service requests and offers with quantities. In Sect. 2, e.g., Clients could express the actual amount of money they are willing to pay. Reaching an agreement would then amount to finding the optimal trade-off among principals such that each one has a positive pay-off function. This might lead to a formalisation of Quality of Service parameters of Service Level Agreements in our model, allowing us to assess non-functional parameters like reliability or energy consumption in a composition of service contracts.

Acknowledgements. This work has been partially supported by the Tuscany Region project POR FESR 2014-2020 SISTER.

REFERENCES

[1] M. Acher, P. Collet, P. Lahire, and J. Montagnat. 2008. Imaging Services on the Grid as a Product Line: Requirements and Architecture. In *Proceedings SOAPL*.

[2] A. Antonik, M. Huth, K.G. Larsen, U. Nyman, and A. Wąsowski. 2008. 20 Years of Modal and Mixed Specifications. *B. EATCS* 95 (2008), 94–129.

[3] M. Bartoletti, T. Cimoli, and R. Zunino. 2015. Compliance in Behavioural Contracts: A Brief Survey. In *Programming Languages with Applications to Biology and Security (LNCS)*, Vol. 9465. Springer, 103–121.

[4] D. Basile, S. Chiaradonna, F. Di Giandomenico, and S. Gnesi. 2016. A stochastic model-based approach to analyse reliable energy-saving rail road switch heating systems. *J. Rail Transp. Plann. Man.* 6, 2 (2016), 163–181.

[5] D. Basile, P. Degano, and G.L. Ferrari. 2016. Automata for Specifying and Orchestrating Service Contracts. *Log. Meth. Comput. Sci.* 12, 4:6 (2016), 1–51.

[6] D. Basile, P. Degano, G.L. Ferrari, and E. Tuosto. 2016. Relating two automata-based models of orchestration and choreography. *J. Log. Algebr. Meth. Program.* 85, 3 (2016), 425–446.

[7] D. Basile, F. Di Giandomenico, and S. Gnesi. 2017. FMCAT: Supporting Dynamic Service Product Lines. In *Proceedings SPLC*, Vol. 2. ACM.

[8] D. Basile, F. Di Giandomenico, S. Gnesi, P. Degano, and G.L. Ferrari. 2017. Specifying Variability in Service Contracts. In *Proceedings VaMoS*. ACM, 20–27.

[9] D. Basile, M.H. ter Beek, F. Di Giandomenico, S. Gnesi, P. Degano, G.L. Ferrari, and A. Legay. 2017. *Controller Synthesis of Contract-based Service Product Lines: Extended Version.* Technical Report 2017-TR-003. ISTI–CNR. http://puma.isti.cnr.it/rmydownload.php?filename=cnr.isti/cnr.isti/2017-TR-003/2017-TR-003.pdf

[10] D.S. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Proceedings SPLC (LNCS)*, Vol. 3714. Springer, 7–20.

[11] M.H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. 2016. Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints. *J. Log. Algebr. Meth. Program.* 85 (2016), 287–315.

[12] M.H. ter Beek, S. Gnesi, and M.N. Njima. 2011. Product Lines for Service Oriented Applications - PL for SOA. In *Proceedings WWV (EPTCS)*, Vol. 61. 34–48.

[13] M.H. ter Beek and F. Mazzanti. 2014. VMC: Recent Advances and Challenges Ahead. In *Proceedings SPLC*, Vol. 2. ACM, 70–77.

[14] M.H. ter Beek, M.A. Reniers, and E.P. de Vink. 2016. Supervisory Controller Synthesis for Product Lines Using CIF 3. In *Proceedings ISoLA (LNCS)*, Vol. 9952. Springer, 856–873.

[15] D. Benavides, S. Segura, and A. Ruiz-Cortés. 2010. Automated Analysis of Feature Models 20 Years Later: a Literature Review. *Inf. Syst.* 35, 6 (2010), 615–636.

[16] C.G. Cassandras and S. Lafortune. 2006. *Introduction to Discrete Event Systems.* Springer, New York.

[17] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. 2013. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. *IEEE Trans. Softw. Eng.* 39, 8 (2013), 1069–1089.

[18] G. Cledou and L.S. Barbosa. 2017. Modeling Families of Public Licensing Services: A Case Study. In *Proceedings FormaliSE*. ACM, 37–43.

[19] D. Fischbein, S. Uchitel, and V.A. Braberman. 2006. A foundation for behavioural conformance in software product line architectures. In *Proceedings ROSATEA*. ACM, 39–48.

[20] D. Georgakopoulos and M.P. Papazoglou (Eds.). 2008. *Service-oriented Computing*. MIT Press, Cambridge, MA, USA.

[21] H. Gomaa and K. Hashimoto. 2011. Dynamic Software Adaptation for Service-Oriented Product Lines. In *Proceedings SPLC*, Vol. 2. ACM, 35:1–35:8.

[22] S. Gunther and T. Berger. 2008. Service-Oriented Product Lines: A Development Process and Feature Management Model for Web Services. In *Proceedings SOAPL*.

[23] K. Larsen, U. Nyman, and A.Wąsowski.2007.Modal I/OAutomata for Interface and Product Line Theories. In *Proceedings ESOP (LNCS)*, Vol. 4421. Springer, 64–79.

[24] K. Lauenroth, K. Pohl, and S. Töhning. 2009. Model Checking of Domain Artifacts in Product Line Engineering. In *Proceedings ASE*. IEEE, 269–280.

[25] M. Mannion. 2002. Using First-Order Logic for Product Line Model Validation. In *Proceedings SPLC (LNCS)*, Vol. 2379. Springer, 176–187.

[26] F.M. Medeiros, E.S. de Almeida, and S.R. de Lemos Meira. 2009. Towards an Approach for Service-Oriented Product Line Architectures. In *Proceedings SOAPL*.

[27] M. Raatikainen, V. Myllärniemi, and T. Männistö. 2007. Comparison of Service and Software Product Family Modeling. In *Proceedings SOAPL*.

[28] P.J. Ramadge and W.M. Wonham. 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.* 25, 1 (1987), 206–230.