Applying Supervisory Control Synthesis to Priced Featured Automata and Energy Problems

Davide Basile

Department of Statistics, Computer Science, Applications - University of Florence,
ISTI CNR Pisa

Abstract. Software Product Line Engineering (SPLE) promotes extensive reuse of common aspects in developing new software components. Supervisory Control Theory (SCT) is a methodology to automatically synthesise a controller enforcing given safety requirements. The interplay between SPLE and SCT has recently received attention in the research community. This paper formally tackles the problem of synthesising a most permissive controller (mpc) enforcing a given requirement for a software product line (SPL). Generally, the number of products of an SPL can be exponential in the number of features, and an mpc should be synthesised for every product. To overcome this problem, the product line structure is exploited to synthesise, in the best case, a number of controllers that are linear in the number of features of the SPL.
The SPL is formalised as a (Priced) Featured Automaton ((P)FA), whilst the mpc synthesis is formalised by modelling both the plant and the requirement as Extended Finite-state Automata (EFA), where quantitative aspects can be seamlessly integrated. The contributions are: (i) a formal mapping from FA to EFA; (ii) a mapping of energy problems onto synthesis of EFA; (iii) three-valued logic and partial-order reduction are used to greatly reduce the number of mpc required. Contribution (iii) holds for a wide range of other objectives, not only energy problems.
Both EFA and PFA are endowed with tools implementing algorithms that have been studied for more than a decade and both are adopted in industry. These results pave the way to reuse algorithms and tools that have been separately developed in SPLE and SCT research areas.

Keywords: extended finite automata, featured automata, featured transition systems, priced featured automata, energy problems, controller, synthesis

1 Introduction

Variability-intensive systems have been proven to provide productivity gains, shorter times to market, and greater market coverage [1]. However, such variability may introduce new problems due to the increasing complexity of the system design. Software Product Line Engineering (SPLE) [1, 2], is a well-established paradigm where multiple software systems (called products) are developed at once, and has been used to manage systems variability. Indeed, by systematically reusing common parts, a benefit from economies of scale can be obtained.

According to this paradigm, the management of single products is lifted to the maintenance of a family or product line of related products. Each product is characterised by a set of relevant features. A feature can be seen as a specific functionality of a product, relevant to stakeholders. Feature models define the set of valid products as combinations (also called configurations) of features [3].

The automated analysis of feature models has a long history [3]. However, especially for critical systems, it is not only important to demonstrate that those products are configured correctly, but also that they behave safely. Recently, behavioural variability formalisms have also been received attention [4, 5]. These formalisms allow the specification of a common behaviour that is then adapted to each particular product configuration. It becomes possible to verify that the behaviour is safe.

Generally, verification is carried on by techniques such as model checking or theorem proving. In this paper, the problem of synthesising a family of product specifications starting from a behavioural description of a product line is studied. The synthesis problem is specified in terms of Ramadge and Wonham's *Theory of Supervisory Control for Discrete Event Systems* [6] (SCT), which is concerned with the synthesis of a *controller* that drives the execution of the system while enforcing given requirements.

It is well-known that the problems of synthesising a controller for a requirement and synthesising a strategy for satisfying such (game-theoretic) requirement share common aspects [7–9]. In particular, *energy problems* are present in various areas such as autonomic systems or cyber-physical systems [10]. Energy problems study whether a system can perform all its tasks without exceeding a given amount of resources (called energy) that are consumed at run-time.

Featured Transition Systems (FTS) and Featured Automata (FA) (i.e. FTS with final states) [11] are well-established paradigms for modelling and verifying product-line based systems. These formalisms have been extended to consider, among the others, energy problems, as well as real-time aspects [12, 11, 13]. Several tools are available to automatise the verification of FA [14, 15].

On the other side, Extended Finite-state Automata (EFA) are an extension of Finite State Automata (FSA) to include data variables [16, 17]. Algorithms for synthesising a controller have been specified through EFA [18], and are implemented in tools such as `CIF3` [19].

We argue that both FA and EFA are widely supported formalisms for, respectively, SPLE and SCT, with industrial applications. Hence, in this paper a formal mapping from FA to EFA is provided. This result paves the way to share results, algorithms and tools that have been separately deployed in these two research areas in the last decades and are ready to be used off-the-shelf.

The main contributions of this paper are:

- a formal mapping from FA (and their quantitative version called Priced Featured Automata (PFA)) to EFA. It is proved that each product of an FA can be mapped to a corresponding EFA by simply changing the initial values of the data variables;

- a mapping from energy problems in PFA to special requirements rendered as EFA, to which SCT can be applied. It is proved that both *safety* and *reachability* energy problems for PFA can be solved by translating both the energy requirement and the system to two EFA and by synthesising their controller. To do so, a revisited notion of refinement of EFA is introduced, with useful auxiliary results;
- the problem of synthesising a supervisor for an FA is extended to a wide range of other requirements, not only energy problems. The product line structure is exploited to obtain a potentially exponential gain in the number of controllers to be synthesised.

Overview Preliminary notions on FA, PFA, energy problems and EFA are provided in Section 2. A revisited refinement of EFA is discussed in Section 3. The formal mapping from (P)FA to EFA is in Section 4, whilst the energy problem requirements mapping is in Section 5. A technique for synthesising a small number of controllers from a product line is described in Section 6. Finally, related work and conclusions are in Section 7.

2 Background

In this section Featured Automata (FA) (and their priced version PFA) and Extended Finite-state Automata (EFA) are recalled. These formalisms have been used in different research areas, i.e. SPLE and SCT, and using their original notations would result in ambiguities (e.g. state/location). Hence, in what follows a homogeneous notation is used that is summarised in Table 1. Firstly, FSA are recalled. A *finite state automaton* $\mathcal{A} = (Q, q_0, \Sigma, T, F)$ consists of a finite set Q of states, an initial state q_0, subset $F \subseteq Q$ of accepting states, a set of actions Σ and a finite set $T \subseteq Q \times \Sigma \times Q$ of transitions. A step is denoted with $q \xrightarrow{\sigma} q'$ and is such that $(q, \sigma, q') \in T$. Let $\xrightarrow{w}{}^*$ denote the transitive closure of $\rightarrow$ where $w \in \Sigma^*$. The language recognised by an FSA $\mathcal{A}$ is $\mathscr{L}(\mathcal{A}) = \{w \mid q_0 \xrightarrow{w}{}^* q, q \in F\}$.

2.1 Featured Automata

The formalization of Featured Automata and Priced Featured Automata is now recalled. With respect to Featured Transition Systems, Featured Automata also allows to declare a set of final states.

Let N be the domain of *features*. A (*feature*) *guard* γ is a Boolean expression over domain N, and $\mathrm{B}(N)$ denotes the set of such guards. Moreover, $p \models \gamma$ if $p \in 2^N$ satisfies $\gamma \in \mathrm{B}(N)$, and $[\![\gamma]\!] = \{p \in 2^N \mid p \models \gamma\}$ is the set of all intepretations (called *products*) satisfying γ. A special feature guard φ called *feature constraint* is such that $[\![\varphi]\!] \subseteq 2^N$ is the set of *valid products*.

Definition 1 (Featured Automaton). *A featured automaton is a tuple* $\mathcal{A}_f = (Q, q_0, \Sigma, T, F, \varphi)$ *consisting of a finite set* Q *of states, an initial state* q_0*, a subset* $F \subseteq Q$ *of accepting states, a set of actions* Σ*, a finite set* $T \subseteq Q \times \mathrm{B}(N) \times \Sigma \times Q$ *of transitions and a feature constraint* φ*.*

Table 1. Table of symbols

Notation	Object
$\mathcal{A}$	a (weighted) finite state automaton
$\mathscr{L}(\mathcal{A})$	language of an FSA
$\mathcal{A}_f$	a (priced) featured automaton
$\mathcal{A}_e$	an extended finite automaton
$\mathcal{A}_{e_1} \| \mathcal{A}_{e_2}$	parallel composition of EFA
Q	set of states
$q_0 \in Q$	the initial state
$q \in Q$	a state
$F \subseteq Q$	accepting states
Σ	alphabet of actions
$\sigma \in \Sigma$	action
$\mathbf{w}_{\mathcal{A}}(w)$	weight of $w \in \mathscr{L}(\mathcal{A})$
$D = D_1 \times \cdots \times D_i \times \cdots \times D_n$	data variables domain
$d \in D$	variable vector
$d(i) \in D_i$	i-th vector element
$D_1 \otimes D_2$	domain composition
$f \in {D_2}^{D_1}$	update function with domain D_1 and co-domain D_2
$f\|_{D_1}$	projection of f on sub-domain D_1
$f_1 \oplus f_2$	function composition
T	set of transitions
$t \in T$	a transition
N	set of features
φ	feature constraint
$[\![\varphi]\!] \subseteq 2^N$	valid products
$p \in [\![\varphi]\!]$	a product
$proj_p(\mathcal{A}_f)$	projection of a featured automaton on product p
$\mathrm{B}(D)$	set of Boolean guards over domain D
$\gamma \in \mathrm{B}(D)$	Boolean guard over D

The *projection* of a FA $\mathcal{A}_f = (Q, q_0, \Sigma, T, F, \varphi)$ to a product $p \in [\![\varphi]\!]$ is the FSA $proj_p(\mathcal{A}_f) = (Q, q_0, \Sigma, T', F)$ with $T' = \{(q, \sigma, q') \mid (q, \gamma, \sigma, q') \in T, p \models \gamma\}$.

Featured automata are extended to include prices (i.e. real number) on edges, thus yielding so-called priced featured automata (PFA) (a.k.a. real-weighted featured automata) [12].

Definition 2 (Priced Featured Automaton). *A priced featured automaton is a tuple $\mathcal{A}_f = (Q, q_0, \Sigma, T, F, \varphi)$ consisting of a finite set Q of states, an initial state q_0, subset $F \subseteq Q$ of accepting states, a set of actions Σ, a finite set $T \subseteq Q \times \mathrm{B}(N) \times \Sigma \times \mathbb{R} \times Q$ of transitions and a feature constraint φ.*

The projection of a PFA on one of its products yields a weighted finite state automaton (WFSA). This formalism is equivalent to FSA but with a finite set $T \subseteq Q \times \Sigma \times \mathbb{R} \times Q$ of transitions. The projection is $proj_p(\mathcal{A}_f) = (Q, q_0, \Sigma, T, F)$ with $T' = \{(q, \sigma, n, q') \in T \mid (q, \gamma, \sigma, n, q') \in T, p \models \gamma\}$.

Given a WFSA $\mathcal{A}$, its language is defined equivalently to FSA. Moreover, the weight of a trace $w \in \mathscr{L}(\mathcal{A})$ is defined as $\mathbf{w}_{\mathcal{A}}(w) = \{n_1 + \ldots + n_m \mid q_0 \xrightarrow{\sigma_1, n_1} \ldots \xrightarrow{\sigma_m, n_m} q, w = \sigma_1 \ldots \sigma_m\}$.

Energy problems on PFA Energy problems have been introduced to solve a variety of problems related to performance and reliability. Two types of energy problems are traditionally considered. Given a WFSA $\mathcal{A}$, a *reachability* energy problem checks the existence of a trace $w \in \mathscr{L}(\mathcal{A})$ such that for all prefixes w' of w it holds $\mathbf{w}(w') \geq 0$. On the converse, a *safety* energy problem checks that there exists no trace $w \in \mathscr{L}(\mathcal{A})$ and prefix w' of w such that $\mathbf{w}(w') < 0$. Energy problems are extended to PFA by simply considering them on all valid products.

2.2 Extended Finite-state Automata

The goal of SCT is to synthesise a *controller* enforcing given requirements onto a system (called plant). In forbidden states control problems, the controller must enforce that *forbidden* states cannot actually be reached while successful (called *marked*) states are always reachable. The theory distinguishes between *controllable* events, that may be disabled by the controller; and *uncontrollable* events, those always enabled. EFA are used to model both the plant (i.e. the system) and the requirement. The synthesised controller is expressed as another EFA, that is a sub-portion of the composition of the plant with the requirement.

EFA are an extension of FSA with data variables and are now recalled. The domain of definition of n *one-dimensional* data variables is denoted with $D = D_1 \times \cdots \times D_i \times \cdots \times D_n$. An *n-dimensional* variable (vector) of domain D is denoted with d, i.e., $d = [d(1), \ldots, d(i), \ldots, d(n)]$, where $d(i)$ is the *ith* data variable of domain D_i. The set of Boolean expressions γ over domain D is denoted with $\mathrm{B}(D)$. In what follows arithmetic operators $(+, -, \geq, \leq)$ will also be considered into expressions γ, when those are defined over domain D (i.e. numbers).

An EFA is an FSA incorporating data variables defined over finite or infinite domains. The transitions of an EFA are augmented by guards over the data variables, and data update functions, which are actions on the data variables.

Definition 3 (Extended Finite Automaton). *An Extended Finite Automaton is a tuple $\mathcal{A}_e = (Q, q_0, \Sigma, T, F, d_0, D)$ consisting of a finite set Q of states, an initial state q_0, a subset $F \subseteq Q$ of accepting states, a set of actions Σ, a finite set $T \subseteq Q \times \mathrm{B}(D) \times \Sigma \times D^D \times Q$ of transitions, a domain of n one-dimensional data variables $D = D_1 \times \cdots \times D_n$ and vector of initial data values $d_0 = d_0^1 \times \cdots \times d_0^n$.*

Given an EFA $\mathcal{A}_e$, subscripts may be used to refer to one of its components, e.g. $Q_{\mathcal{A}_e}$ is the set of states of $\mathcal{A}_e$. The semantics of an EFA $\mathcal{A}_e$ is defined on its configurations $(q, d) \in Q_{\mathcal{A}_e} \times D_{\mathcal{A}_e}$ that are pairs of states and data evaluations. The initial configuration is (q_0, d_0). A transition $t = (q_s, \gamma, \sigma, f, q_t)$ is composed of source state q_s, enabling guard γ , action σ, update function f and target state q_t. A *step* is denoted by $(q, d) \xrightarrow{\sigma} (q', d')$ for a transition $t = (q, \gamma_t, \sigma, f_t, q') \in T_{\mathcal{A}_e}$

such that $\gamma_t(d) = \text{true}, f_t(d) = d'$. Let $\xrightarrow{w}{}^*$ denotes the transitive closure of $\rightarrow$ with $w \in \Sigma^*$. The language of an EFA $\mathcal{A}_e$ is $\mathscr{L}(\mathcal{A}_e) = \{w \mid (q_0, d_0) \xrightarrow{w}{}^* (q, d), q \in F_{\mathcal{A}_e}\}$. States q or q' as well data vectors d or d' and action σ may be omitted from $(q,d) \xrightarrow{\sigma} (q',d')$ when those are immaterial. From now on only reachable configurations (q,d) such that $(q_0,d_0) \rightarrow^* (q,d)$ will be considered.

In the following, $\mathcal{A}'_e$ is a subautomaton of $\mathcal{A}_e$ ($\mathcal{A}'_e \preceq \mathcal{A}_e$), if $\mathcal{A}'_e$ is obtained from $\mathcal{A}_e$ by removing some states of $\mathcal{A}_e$ and their incident transitions, by removing some transitions of $\mathcal{A}_e$ and by possibly strengthening the guards of the transitions of $\mathcal{A}_e$.

EFA can be composed in parallel, provided that they have the same initial data values for all shared variables.

The *composition of domains* $\otimes$ is specified in such a way that the shared sub-domains are not replicated, i.e. $D_1 \otimes D_2 = D'_1 \times D_s \times D'_2$ where D_s is the shared sub-domain between $D_1 = D'_1 \times D_s$ and $D_2 = D_s \times D'_2$.

The projection of an update function f on one of its sub-domain D_s is denoted with $f|_{D_s} \in {D_s}^{D_s}$. Given two update functions $f_i \in {D_i}^{D_i}, i \in \{1,2\}$, their composition is $f_1 \oplus f_2 \in {D_1 \otimes D_2}^{D_1 \otimes D_2}$, $f_1 \oplus f_2 = f_1 \times f_2|_{D'_2} = f_1|_{D'_1} \times f_2$. Basically the composed function requires both operands to behave equivalently on the shared domain.

Intuitively, the composition interleaves all non-shared actions whilst components synchronize on the shared actions. Let $\mathcal{A}_{e_k}$, $k = 1,2$ be two EFA. The parallel composition of $\mathcal{A}_{e_1}$ and $\mathcal{A}_{e_2}$ is $\mathcal{A}_{e_1} \| \mathcal{A}_{e_2} = (Q_{\mathcal{A}_{e_1}} \times Q_{\mathcal{A}_{e_2}}, q_{0_{\mathcal{A}_{e_1}}} \times q_{0_{\mathcal{A}_{e_2}}}, \Sigma_{\mathcal{A}_{e_1}} \cup \Sigma_{\mathcal{A}_{e_2}}, T, F_{\mathcal{A}_{e_1}} \times F_{\mathcal{A}_{e_2}}, d_{0_{\mathcal{A}_{e_1}}} \otimes d_{0_{\mathcal{A}_{e_2}}}, D_{\mathcal{A}_{e_1}} \otimes D_{\mathcal{A}_{e_2}})$ where the set of transitions T is defined such that interleavings happen on actions $\sigma \in (\Sigma_{\mathcal{A}_{e_1}} \setminus \Sigma_{\mathcal{A}_{e_2}}) \cup (\Sigma_{\mathcal{A}_{e_2}} \setminus \Sigma_{\mathcal{A}_{e_1}})$ whilst for synchronizations:

$$\forall \sigma \in \Sigma_{\mathcal{A}_{e_1}} \cap \Sigma_{\mathcal{A}_{e_2}}, \forall (q_1, \gamma_1, \sigma, f_1, q'_1) \in T_{\mathcal{A}_{e_1}}, \forall (q_2, \gamma_2, \sigma, f_2, q'_2) \in T_{\mathcal{A}_{e_2}} :$$

$$((q_1, q_2), \gamma_1 \wedge \gamma_2 \wedge (f_1|_{D_s} = f_2|_{D_s}), \sigma, f_1 \oplus f_2, (q'_1, q'_2)) \in T$$

Supervisory Control of EFA In SCT of EFA the behaviour of the uncontrolled system is specified by a plant EFA $\mathcal{A}_{e_p}$ and the requirement the controller must enforce by another EFA $\mathcal{A}_{e_r}$. By refining $\mathcal{A}_{e_p}$ with respect to $\mathcal{A}_{e_r}$, a refined plant model $\mathcal{A}_{e_g}$ can be obtained with the same behavior as $\mathcal{A}_{e_p}$ such that the executions violating (i.e. not allowed) $\mathcal{A}_{e_r}$ leads to forbidden states in $\mathcal{A}_{e_g}$. The refined EFA $\mathcal{A}_{e_g}$ is such that: $q_{0_{\mathcal{A}_{e_g}}} = q_{0_{\mathcal{A}_p}} \times q_{0_{\mathcal{A}_r}}$; $Q_{\mathcal{A}_{e_g}} = Q_{\mathcal{A}_p} \times (Q_{\mathcal{A}_r} \cup \{q_f\})$ (q_f forbidden state); $F_{\mathcal{A}_{e_g}} = F_{\mathcal{A}_p} \times F_{\mathcal{A}_r}$; and $T_{\mathcal{A}_{e_g}}$ is defined such that only synchronous transitions in $\mathcal{A}_{e_p} \| \mathcal{A}_{e_r}$ are kept whilst interleavings lead to *forbidden* (i.e. deadlocked) configurations.

It is then possible to provide the plant and requirement models as a given EFA $\mathcal{A}_{e_g}$ such that the requirement is given as a set of forbidden states $Q_f \subseteq Q_{\mathcal{A}_{e_g}}$, and $Q_s = Q_{\mathcal{A}_{e_g}} - Q_f$ is the set of safe states of $\mathcal{A}_{e_g}$. A configuration (q,d) is a *forbidden* configuration iff $q \in Q_f$, otherwise, (q,d) is a *safe* configuration. In $\mathcal{A}_{e_g}$ by construction it is impossible to reach a safe configuration from a forbidden configuration, and no forbidden state is accepting. In the sequel $\mathcal{A}_{e_g}^{safe}$ denotes

the EFA obtained from $\mathcal{A}_{e_g}$ by assigning *false* to the guard of each transition t for which $q_{t_t} \in Q_f$, i.e., the target state of t is a forbidden state. Following the way $\mathcal{A}_{e_g}^{safe}$ is constructed, it holds that $\mathcal{A}_{e_g}^{safe} \preceq \mathcal{A}_{e_g}$. The *safe subautomaton of* $\mathcal{A}_{e_g}$ is denoted with $\mathcal{A}_{e_g}^{safe}$.

Let $\Sigma_c \subseteq \Sigma$ and $\Sigma_u = \Sigma - \Sigma_c$ be the set of controllable and uncontrollable events of $\mathcal{A}_{e_g}$, respectively. The objective of control is to satisfy nonblockingness and safety while satisfying controllability requirements.

In particular, a configuration (q, d) of $\mathcal{A}_{e_g}$ is: (a) *nonblocking if* $q \rightarrow^* q'$ and $q' \in F_{\mathcal{A}_{e_g}}$; (b) *safe* if (q, d) is also a configuration of $\mathcal{A}_{e_g}^{safe}$; and (c) *controllable* if (q, d) is safe and $\forall \sigma \in \Sigma_u . q \xrightarrow{\sigma} q'$, q' is safe. The EFA $\mathcal{A}_{e_g}^{safe}$ is, respectively, nonblocking, safe, and controllable if all its reachable states are such.

A controller assigns a (possibly) stronger guard to each controllable transitions, whilst guards on uncontrollable transitions are kept.

Let $\mathcal{A}_{e_g}^{\mathcal{S}}$ denote the subautomaton obtained from $\mathcal{A}_{e_g}$ by replacing its guards by those provided by its controller $\mathcal{S}$, which is said to be nonblocking if $\mathcal{A}_{e_g}^{\mathcal{S}}$ is nonblocking and safe if $\mathcal{A}_{e_g}^{\mathcal{S}}$ is safe. The existence of a nonblocking and safe controller $\mathcal{S}$ for a given plant $\mathcal{A}_{e_g}$ and (safe) specification $\mathcal{A}_{e_g}^{safe}$ such that $\mathcal{A}_{e_g}^{safe} \preceq \mathcal{A}_{e_g}$ are known: the specification must be nonblocking and controllable [18].

In case such conditions are not satisfied, it is possible to compute a safe and nonblocking controller $\mathcal{S}$ such that $\mathcal{A}_{e_g}^{\mathcal{S}} \preceq \mathcal{A}_{e_g}^{safe}$.

The supremal controllable and nonblocking subautomaton of $\mathcal{A}_{e_g}^{safe}$, called *most permissive controller* (mpc), is denoted with $\mathcal{A}_{e_g}^{\mathcal{K}}$, and is such that for any other controller $\mathcal{S}$ it holds $\mathcal{A}_{e_g}^{\mathcal{S}} \preceq \mathcal{A}_{e_g}^{\mathcal{K}}$. An algorithm for computing $\mathcal{A}_{e_g}^{\mathcal{K}}$ is presented in [18].

3 Refinement of EFA

In this section, a revisited notion of refinement of EFA is introduced. It is a conservative extension of the notion of sub-automaton in Section 2.2. Moreover, some auxiliary results are discussed.

Firstly, the notion of sub-automaton introduced in Section 2.2 will be (conservatively) extended.

Definition 4 (Refinement of EFA). *Let $\mathcal{A}_{e_1}$ and $\mathcal{A}_{e_2}$ be two EFA, then $\mathcal{A}_{e_1}$* refines *$\mathcal{A}_{e_2}$ (denoted $\mathcal{A}_{e_1} \preceq \mathcal{A}_{e_2}$) iff $Q_{\mathcal{A}_{e_1}} \subseteq Q_{\mathcal{A}_{e_2}}$, $q_{0_{\mathcal{A}_{e_1}}} = q_{0_{\mathcal{A}_{e_2}}}$, $\Sigma_{\mathcal{A}_{e_1}} \subseteq \Sigma_{\mathcal{A}_{e_2}}$, $F_{\mathcal{A}_{e_1}} \subseteq F_{\mathcal{A}_{e_2}}$, $D_{\mathcal{A}_{e_2}}$ sub-domain of $D_{\mathcal{A}_{e_1}}$; $\exists tr : T_{\mathcal{A}_{e_1}} \mapsto T_{\mathcal{A}_{e_2}}$ injective where $tr(t_1) = t_2$ such that g_{t_1} implies g_{t_2}, $f_{t_1}|_{D_{\mathcal{A}_{e_2}}} = f_{t_2}$, and t_1 and t_2 are equal on all other components. Moreover, $d_{0_{\mathcal{A}_{e_1}}} = d_{0_{\mathcal{A}_{e_2}}} \times d'_0$ for some d'_0 on $D_{\mathcal{A}_{e_1}} \setminus D_{\mathcal{A}_{e_2}}$ and no dangling nodes and transitions are left in $\mathcal{A}_{e_1}$.*

In the following sections a mapping from FA and energy problems to EFA will be provided. This mapping exploits the fact that in FA all actions are controllable. Moreover, the requirement EFA for the energy problem will be a refinement of the plant EFA. The additional results of this section are used for proving the following main results.

With respect to the sub-automaton relation, the refinement additionally allows to introduce new variables if they do not affect the behaviour of the refined EFA. It is easy to see that a sub-automaton is also a refinement.

Proposition 1. *Let $\mathcal{A}_{e_1}$ and $\mathcal{A}_{e_2}$ be two EFA, then:*

$$\mathcal{A}_{e_1} \text{sub-automaton of } \mathcal{A}_{e_2} \textit{ implies } \mathcal{A}_{e_1} \preceq \mathcal{A}_{e_2}$$

Proof. From the fact that a sub-automaton is obtained by removing states and their incident transitions, by strengthening guards; and by Definition 4 the thesis follows. □

The following lemma states that if the requirement is a refinement of the plant, then the corresponding safe sub-automaton $\mathcal{A}_{e_g}^{safe}$ defined in Section 2.2 is trace-equivalent to the refinement EFA.

Lemma 1. *Let $\mathcal{A}_{e_1}$ and $\mathcal{A}_{e_2}$ be two EFA such that $\mathcal{A}_{e_1} \preceq \mathcal{A}_{e_2}$. Let $\mathcal{A}_{e_g}^{safe}$ be the safe sub-automaton obtained from $\mathcal{A}_{e_2}||\mathcal{A}_{e_1}$ with the procedure in Section 2.2. Then*

$$\mathscr{L}(\mathcal{A}_{e_g}^{safe}) = \mathscr{L}(\mathcal{A}_{e_1})$$

Proof. From the definition of the operator $||$, the set of transitions of $\mathcal{A}_{e_g}^{safe}$ contains the set of transitions T such that : $\forall \sigma \in \Sigma_{\mathcal{A}_{e_1}} \cap \Sigma_{\mathcal{A}_{e_2}}, \forall (q_1, \gamma_1, \sigma, f_1, q_1') \in T_{\mathcal{A}_{e_1}}, \forall (q_2, \gamma_2, \sigma, f_2, q_2') \in T_{\mathcal{A}_{e_2}} : ((q_1, q_2), \gamma_1 \wedge \gamma_2 \wedge (f_1|_{D_s} = f_2|_{D_s}), \sigma, f_1 \oplus f_2, (q_1', q_2')) \in T$. By definition of function composition $\oplus$ and restriction $|$, and by Definition 4 it follows that for each transition $(q_1, q_1) \xrightarrow{\sigma} (q_2, q_2) \in T$ there exists one and only one transition $q_1 \xrightarrow{\sigma} q_2 \in T_{\mathcal{A}_{e_1}}$, equivalent on guard and update function. Moreover, by definition of $\mathcal{A}_{e_g}$ each interleaved transition in $\mathcal{A}_{e_2}||\mathcal{A}_{e_1}$ lead to a forbidden state and thus the guard of such transition is set to *false* in $\mathcal{A}_{e_g}^{safe}$. Hence each transition in $\mathcal{A}_{e_g}^{safe}$ either belongs to T or it is never enabled, and the thesis follows. □

If all actions in $\mathcal{A}_{e_g}^{safe}$ are controllable, then $\mathcal{A}_{e_g}^{safe}$ is trace equivalent to $\mathcal{A}_{e_g}^{\mathcal{K}}$.

Lemma 2. *Let $\mathcal{A}_{e_g}^{safe}$ be a safe automaton and $\mathcal{A}_{e_g}^{\mathcal{K}}$ be its corresponding mpc. If all actions of $\mathcal{A}_{e_g}^{safe}$ are* controllable *then*

$$\mathscr{L}(\mathcal{A}_{e_g}^{safe}) = \mathscr{L}(\mathcal{A}_{e_g}^{\mathcal{K}})$$

Proof. The thesis follows trivially by the fact that $\mathcal{A}_{e_g}^{safe}$ is safe and controllable by definition and hypothesis, and because the trace language only considers non-blocking configurations and $\mathcal{A}_{e_g}^{\mathcal{K}}$ is supremal. □

The following corollary can now be stated.

Corollary 1. *Let $\mathcal{A}_{e_1}$ and $\mathcal{A}_{e_2}$ be two EFA such that $\mathcal{A}_{e_1} \preceq \mathcal{A}_{e_2}$. Let $\mathcal{A}_{e_g}^{safe}$ be the safe sub-automaton obtained from $\mathcal{A}_{e_2}||\mathcal{A}_{e_1}$ with the procedure in Section 2.2. Let $\mathcal{A}_{e_g}^{\mathcal{K}}$ be its corresponding mpc. If all actions of $\mathcal{A}_{e_1}$ are* controllable *then*

$$\mathscr{L}(\mathcal{A}_{e_1}) = \mathscr{L}(\mathcal{A}_{e_g}^{\mathcal{K}})$$

Proof. By Lemma 1 and Lemma 2. □

4 From FA to EFA

The formal mapping $[\![-]\!]$ from FA to EFA is now presented. The mapping is almost straightforward. Indeed, it suffices to use the domain of features N as domain of variables D, where each feature is a boolean variable, and ignore the update function on transitions of the EFA. Finally, each valid product p of a FA will be in correspondence to an EFA by simply mapping the boolean assignment of p to the initial vector of variables d_0. In the following, abusing notation, $d_0(f)$ denotes the element of vector d_0 corresponding to feature f, whilst $p(f)$ is the boolean value assigned by p to f, i.e., the result of applying the interpretation function p.

Definition 5 (Translation from FA to EFA). *Let $\mathcal{A}_f$ be an FA and $p \in [\![\varphi_{\mathcal{A}_f}]\!]$ one of its valid products, then the corresponding EFA $[\![\mathcal{A}_f]\!]_p = \mathcal{A}_{e_p}$ is such that*

$$Q_{\mathcal{A}_{e_p}} = Q_{\mathcal{A}_f}, q_{0_{\mathcal{A}_{e_p}}} = q_{0_{\mathcal{A}_f}}, \Sigma_{\mathcal{A}_{e_p}} = \Sigma_{\mathcal{A}_f}, F_{\mathcal{A}_{e_p}} = F_{\mathcal{A}_f}, D = N,$$
$$\forall f \in N. d_0(f) = p(f)$$
$$T_{\mathcal{A}_{e_p}} = \{(q_s, \gamma, \sigma, \emptyset, q_f) \mid (q_s, \gamma, \sigma, q_f) \in T_{\mathcal{A}_f}\}$$

The following proposition states that the translation of Definition 5 for any two valid products yields the same EFA, except for the initial data vector d_0.

Proposition 2. *Let $\mathcal{A}_f$ be an FA and $[\![\varphi_{\mathcal{A}_f}]\!]$ its set of valid products. Then $\forall p, p' \in [\![\varphi_{\mathcal{A}_f}]\!], p \neq p'$ where $[\![\mathcal{A}_f]\!]_p = \mathcal{A}_{e_p}$ and $[\![\mathcal{A}_f]\!]_{p'} = \mathcal{A}_{e_{p'}}$ it holds that*

$$Q_{\mathcal{A}_{e_p}} = Q_{\mathcal{A}_{e_{p'}}}, q_{0_{\mathcal{A}_{e_p}}} = q_{0_{\mathcal{A}_{e_{p'}}}}, \Sigma_{\mathcal{A}_{e_p}} = \Sigma_{\mathcal{A}_{e_{p'}}}, F_{\mathcal{A}_{e_p}} = F_{\mathcal{A}_{e_{p'}}}, T_{\mathcal{A}_{e_p}} = T_{\mathcal{A}_{e_{p'}}}$$

Proof. The proof is immediate by Definition 5. □

An immediate consequence of Proposition 2 is that the translation from FA to EFA can be performed only once for all valid products. Then it suffices to initialise d_0 with the corresponding valid product p to obtain the corresponding EFA of valid product p.

The following theorem states that the translation from FA to EFA for product p is trace-equivalent to the projection of FA on p.

Theorem 1. *Let $\mathcal{A}_f$ be an FA and p be one of its valid products, then:*

$$\mathscr{L}([\![\mathcal{A}_f]\!]_p) = \mathscr{L}(proj_p(\mathcal{A}_f))$$

Proof. Firstly we prove $\mathscr{L}([\![\mathcal{A}_f]\!]_p) \subseteq \mathscr{L}(proj_p(\mathcal{A}_f))$. By contradiction, assume there exists $w \in \mathscr{L}([\![\mathcal{A}_f]\!]_p)$ such that $w \notin \mathscr{L}(proj_p(\mathcal{A}_f))$. This means that there exists a transition $t = (q, \gamma, \sigma, \emptyset, q_f) \in T_{\mathcal{A}_{e_p}}$ used for recognising w such that $(q, \sigma, q_f) \notin T_{proj_p(\mathcal{A}_f)}$. By definition of $proj_p$, this is the case only if for $(q, \gamma, \sigma, q_f) \in T_{\mathcal{A}_f}, p \not\models \gamma$. By Definition 5 for all transitions $t \in T_{\mathcal{A}_{e_p}}$ the update function is empty, hence t can only be fired from the configuration (q, d_0) of $\mathcal{A}_{e_p}$,

and this is possible only if $d_0 \models \gamma$. Again, by Definition 5 $\forall f \in N.d(f) = p(f)$, and since $p \not\models \gamma$ it follows $d_0 \not\models \gamma$ and the transition t is not enabled in (q, d_0), a contradiction.

The proof for $\mathscr{L}(proj_p(\mathcal{A}_f)) \subseteq \mathscr{L}([\![\mathcal{A}_f]\!]_p)$ is similar. By contradiction, assume there exists $w \in \mathscr{L}(proj_p(\mathcal{A}_f))$ such that $w \notin \mathscr{L}([\![\mathcal{A}_f]\!]_p)$. This means that there exists a transition $(q, \sigma, q_f) \in T_{proj_p(\mathcal{A}_f)}$ used for recognising w such that $t = (q, \gamma, \sigma, \emptyset, q_f) \in T_{\mathcal{A}_{e_p}}$ is never enabled in state q. Let $(q, \gamma, \sigma, q_f) \in T_{\mathcal{A}_f}$ be the corresponding transition in the FA. By definition of $proj_p$ it holds that $p \models \gamma$. By Definition 5 for all transitions $t \in T_{\mathcal{A}_{e_p}}$ the update function is empty, hence t can only be fired from the configuration (q, d_0) of $\mathcal{A}_{e_p}$, and this is possible only if $d_0 \models \gamma$. Again, by Definition 5 $\forall f \in N.d(f) = p(f)$, and $p \models \gamma$ implies $d_0 \models \gamma$ and the transition t is enabled in (q, d_0), a contradiction. □

Remark. The most permissive controller synthesis has not been applied in Definition 5 because transitions of FA are always controllable, and non-blockingness is trivially enforced by considering the language of the automaton (cf. Lemma 2). However, through Definition 5 it would be possible to seamlessly introduce uncontrollable transitions in an FA so to apply the mpc synthesis.

5 Mapping Energy Problems on EFA

An important outcome of the translation in Definition 5 is that the *mpc synthesis* for a generic requirement (rendered as an EFA) can be applied to any FA. Indeed, it suffices to synthesise the mpc of each valid product.

As an example, in this section both energy problems (reachability and safety, cf. Section 2.1) will be mapped into a single mpc synthesis problem. This mapping is almost straightforward: it suffices to add a new variable to the domain of features that acts as a counter c: initially c is set to zero, each transition t requires $c+n \geq 0$ for being enabled (where n is the weight of t) and c is updated by $f(c) = c + n$.

Definition 6 (Energy problem mapping to EFA). *Let $\mathcal{A}_{e_p} = [\![A_f]\!]_p$ be an EFA obtained from Definition 5. The corresponding requirement $\mathcal{A}_{e_r}$ for the* energy *problem on $\mathcal{A}_f$ and product p is:*

$$Q_{\mathcal{A}_{e_r}} = Q_{\mathcal{A}_{e_p}}, q_{0_{\mathcal{A}_{e_r}}} = q_{0_{\mathcal{A}_{e_p}}}, \Sigma_{\mathcal{A}_{e_r}} = \Sigma_{\mathcal{A}_{e_p}}, F_{\mathcal{A}_{e_r}} = F_{\mathcal{A}_{e_p}}, D_{\mathcal{A}_{e_r}} = D_{\mathcal{A}_{e_p}} \times \mathbb{R}$$

$$d_{0_{\mathcal{A}_{e_r}}} = [d_{0_{\mathcal{A}_{e_p}}}(0), \ldots, d_{0_{\mathcal{A}_{e_p}}}(m-1), 0] \text{ where } m \text{ is the length of } d_0$$

$$T_{\mathcal{A}_{e_r}} = \{(q_s, \gamma \wedge (d(m) + n \geq 0), \sigma, n, f(d(m)) = d(m) + n, q_t) \mid (q_s, \gamma, \sigma, n, q_t) \in T_{\mathcal{A}_f}\}$$

Some auxiliary results are now discussed, useful for proving the main result of this section. Firstly, the (energy) requirement computed through Definition 6 is a refinement of the corresponding plant EFA.

Lemma 3. *Let $\mathcal{A}_f$ be an FA and p one of its valid products, $\mathcal{A}_{e_p}$ be the plant EFA and $\mathcal{A}_{e_r}$ be the requirement EFA obtained respectively from Definition 5 and Definition 6. Then it holds*

$$\forall p \in [\![\varphi_{\mathcal{A}_f}]\!].\mathcal{A}_{e_r} \preceq \mathcal{A}_{e_p}$$

Proof. By Definition 4; condition on the set of states, initial state, alphabet and final states hold. Moreover, $D_{\mathcal{A}_{e_r}} = D_{\mathcal{A}_{e_p}} \times \mathbb{R}$ hence $D_{\mathcal{A}_{e_p}}$ is a sub-domain of $D_{\mathcal{A}_{e_r}}$. Finally, $T_{\mathcal{A}_{e_r}} = \{(q_s, \gamma \wedge (d(m) + n \geq 0), \sigma, n, f(d(m)) = d(m) + n, q_t) \mid (q_s, \gamma, \sigma, n, q_t) \in T_{\mathcal{A}_f}\}$ hence $\gamma \wedge (d(m) + n \geq 0)$ implies γ and f is only defined for $d(m)$. □

The following result ensures that the requirement EFA $\mathcal{A}_{e_r}$ of Definition 6 only enforces configurations reachable through traces with non-negative sum of weights.

Lemma 4. *Let $\mathcal{A}_f$ be an FA and p one of its valid products, $\mathcal{A}_{e_p}$ be the plant EFA and $\mathcal{A}_{e_r}$ be the requirement EFA obtained respectively from Definition 5 and Definition 6. Moreover, let $\mathcal{A}_{e_g}^{\mathcal{K}}$ be the mpc of $\mathcal{A}_{e_p}||\mathcal{A}_{e_r}$; and let $|d_0| = r$ be the cardinality of the data vector of $\mathcal{A}_{e_g}^{\mathcal{K}}$. Then, for every trace $ww' \in \mathscr{L}(\mathcal{A}_{e_g}^{\mathcal{K}})$ such that $(q_0, d_0) \xrightarrow{w} (q, d)$ it holds that $d(r-1) \geq 0$.*

Proof. From Corollary 1 and Lemma 3 it suffices to consider $ww' \in \mathscr{L}(\mathcal{A}_{e_r})$.

By Definition 5 and Definition 6, it holds that for every data vector d of $\mathcal{A}_{e_r}$, the elements $d(0), \ldots, d(r-2)$ correspond to features of the corresponding valid product p, whilst $d(r-1)$ is a real-valued variable. By Definition 6, for any trace $\pi = (q_0, d_0) \xrightarrow{\sigma_1, n_1} \ldots \xrightarrow{\sigma_m, n_m} (q, d)$ it holds that $d(r-1) = 0 + n_1 + \ldots + n_m$.

We proceed by induction on the lenght of the trace π. In the base case $\pi = (q_0, d_0) \xrightarrow{\sigma_1, n_1} (q, d')$ and $d'(r-1) = 0 + n_1$. Let $t = (q_0, \gamma \wedge (d(r-1) + n1 \geq 0), \sigma, f(d(r-1)) = d(r-1) + n_1, q)$ be the transition fired in π. The fact that t is enabled (and fired) in π implies $n_1 \geq 0$, which in turns implies $d'(r-1) = d(r-1) + n_1 \geq 0$.

The path $\pi = (q_0, d_0) \xrightarrow{\sigma_1, n_1} \ldots \xrightarrow{\sigma_{m-1}, n_{m-1}} (q, d) \xrightarrow{\sigma_m, n_m} (q', d')$ is considered for the inductive case, and by inductive hypothesis $d(r-1) \geq 0$. Let $t = (q, \gamma \wedge (d(r-1) + n_m \geq 0), \sigma, f(d(r-1)) = d(r-1) + n_m, q')$ be the last transition fired in π. Again, the fact that t is enabled (and fired) in π implies $n_m \geq -d(r-1)$, which implies $d'(r-1) = d(r-1) + n_m \geq 0$. □

The following theorem states a correspondence between solving an energy problem (safety or reachability) for an FA and computing the mpc for the corresponding EFA translation (of plant and requirement).

Theorem 2 (Energy Games and mpc). *Let $\mathcal{A}_f$ be an FA and p one of its valid products, $\mathcal{A}_{e_p}$ be the plant EFA and $\mathcal{A}_{e_r}$ be the requirement EFA obtained respectively from Definition 5 and Definition 6. Moreover, let $\mathcal{A}_{e_g}^{\mathcal{K}}$ be the mpc of $\mathcal{A}_{e_p}||\mathcal{A}_{e_r}$. The following hold:*

1. *the* reachability *energy problem on $proj_p(\mathcal{A}_f)$ has a solution iff $\mathscr{L}(\mathcal{A}_{e_g}^{\mathcal{K}}) \neq \emptyset$;*

2. *the* safety *energy problem on* $proj_p(\mathcal{A}_f)$ *has a solution iff it holds that* $\mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g}) = \mathscr{L}(proj_p(\mathcal{A}_f))$*;*
3. *the* safety *energy problem on* $\mathcal{A}^{\mathcal{K}}_{e_g}$ *has a solution.*

Proof. Firstly, for each $w \in \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g})$ such that $(q_0, d_0) \xrightarrow{w}{}^{*} (q, d)$, it holds that $d(r-1) = \mathbf{w}(w)$.

1. Recall that a *reachability* energy problem checks the existence of a trace $w \in \mathscr{L}(proj_p(\mathcal{A}_f))$ such that for all prefixes w' of w it holds $\mathbf{w}(w') \geq 0$. By Theorem 1, Lemma 4 and the hypothesis $\mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g}) \neq \emptyset$ the thesis follows.
2. Recall that a *safety* energy problem checks that there exists no trace $w \in \mathscr{L}(proj_p(\mathcal{A}_f))$ and prefix w' of w such that $\mathbf{w}(w') < 0$.
   For the if case, assume $\mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g}) = \mathscr{L}(proj_p(\mathcal{A}_f))$; the thesis follows directly by Lemma 4.
   For the only if case, assume that the *safety* energy problem has a solution on $proj_p(\mathcal{A}_f)$. By Corollary 1 it suffices to prove that $\mathscr{L}(\mathcal{A}_{e_r}) = \mathscr{L}(proj_p(\mathcal{A}_f))$ holds. By Theorem 1, it suffices to prove that $\mathscr{L}(\mathcal{A}_{e_r}) = \mathscr{L}(\mathcal{A}_{e_p})$ holds. This equivalence follows directly from the hypothesis and Definition 6.
3. It follows directly by Lemma 4. □

The mpc synthesis produces more information than the one obtained by separately solving both reachability and safety energy problem. Indeed, if only the reachability energy problem has a solution, but not the safety one, then the mpc synthesis computes the *largest safe* sub-automaton of $proj_p(\mathcal{A}_f)$. Finally, the synthesised mpc can be easily turned into a winning strategy for the corresponding energy problem.

Remark As noticed in the previous section, uncontrollable actions can be seamlessly introduced and the results presented in this section can be extended also to energy *games*.

6 Partial Order Reduction

A drawback of product line modelling is that generally valid products are exponential in the number of features. For example, consider the feature constraint $\varphi = \bigvee_{i \in 1...20} f_i$. The cardinality of $[\![\varphi]\!]$ is of $2^{20} - 1$, i.e. one million of valid products. The techniques described in the previous sections should then compute one million controllers, without exploiting any of the common parts these different products share. Indeed, all valid products are interpretations satisfying the same propositional formula. The plant automaton (or the FA) is the same for all valid products. Finally, the requirement to be enforced by the mpc is the same for all valid products.

The following section will discuss the relations between the different controllers of valid products and how this information can be exploited for verification or synthesis purposes.

Three-valued logic for valid products The main intuition behind the partial order reduction is now sketched. Generally, for a product line modelled as a propositional logic formula φ, the set of valid products is represented by the set of *total* interpretation functions satisfying φ. Those functions assign a truth value to each feature in φ. In a *three-valued* logic, atoms can be evaluated either to true, false, or "don't care" (denoted by $\bullet$). By switching from propositional logic to a three-valued logic it is possible to drastically reduce the number of (augmented) valid products to check.

For example, given $\varphi = a \vee b$, the interpretation functions such that $a = \mathit{true}$ and $b = \bullet$, and vice versa, are both satisfying φ. Hence, in this case three (augmented) valid products are considered. The projection on a valid product p simply ignores those features f such that $p(f) = \bullet$. From now on, the set of valid products $[\![\varphi]\!]$ is intended to consider also such augmented valid products.

All valid products $[\![\varphi]\!]$ of a product line can be partially ordered by set inclusion. Given a valid product p, let $\mathit{Forbidden}(p) = \{\, f \in N \mid p(f) = \mathit{false} \,\}$.

Definition 7 (Partially ordering $[\![\varphi]\!]$). *Let $[\![\varphi]\!]$ be the set of valid products. Then $([\![\varphi]\!], \preceq)$ is a partially ordered set, where*

$$p \preceq p' \text{ iff } \mathit{Forbidden}(p') \subseteq \mathit{Forbidden}(p)$$

Note that the relation $\preceq$ is overloaded for both valid products and EFA: the operands of $\preceq$ are used for identifying which relation is used. Before providing the main results of this section, by Theorem 8 in [11] without loss of generality it is assumed that each guard of an FA or PFA is composed of only one feature. Hence, if for a given product p such feature has value $\bullet$, the corresponding transition will be maintained in proj_p. The following results hold for every mpc $\mathcal{A}^{\mathcal{K}}_{e_g}$ of a given requirement $\mathcal{A}_{e_r}$.

Lemma 5 shows that, intuitively, the partial order on valid products induces a refinement of (mapped) EFA.

Lemma 5. *Let $p, p' \in [\![\varphi_{\mathcal{A}}]\!]$ be two valid products of an (P)FA $\mathcal{A}_f$, and let $\mathcal{A}_{e_p}, \mathcal{A}_{e_{p'}}$ be the corresponding EFA computed in Definition 5. Then:*

$$p \preceq p' \;\; \textit{implies} \;\; \mathscr{L}(\mathcal{A}_{e_p}) \subseteq \mathscr{L}(\mathcal{A}_{e_{p'}})$$

Proof. Since $T_{proj_p(\mathcal{A}_f)} \subseteq T_{proj_{p'}(\mathcal{A}_f)}$ holds because $p \preceq p'$ (recall that each feature guard is composed of only one feature), by Definition 5 the thesis holds. □

Lemma 5 is extended in Corollary 2 to also consider the mpc of a given requirement. In particular, the mpc $\mathcal{A}^{\mathcal{K}}_{e_g}$ for a valid product p is a refinement of the mpc $\mathcal{A}^{\mathcal{K}}_{e_{g'}}$ of a product p' such that $p \preceq p'$.

Corollary 2. *Let $p, p' \in [\![\varphi_{\mathcal{A}}]\!]$ be two valid products of an (P)FA $\mathcal{A}_f$, and let $\mathcal{A}_{e_p}, \mathcal{A}_{e_{p'}}$ be the corresponding EFA computed in Definition 5. Moreover, let $\mathcal{A}_{e_r}$ be an EFA requirement and $\mathcal{A}^{\mathcal{K}}_{e_g}, \mathcal{A}^{\mathcal{K}}_{e_{g'}}$ their corresponding mpc. Then:*

$$p \preceq p' \;\; \textit{implies} \;\; \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g}) \subseteq \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_{g'}})$$

Proof. By contradiction, let $w \in \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g}) \setminus \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_{g'}})$, and let $t = (q, \gamma, \sigma, f, q')$ be the first transition in the path recognizing w that is not enabled in $\mathcal{A}^{\mathcal{K}}_{e_{g'}}$. By Lemma 2 it holds that $\mathscr{L}(\mathcal{A}^{safe}_{e_g}) = \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_g})$ and $\mathscr{L}(\mathcal{A}^{safe}_{e_{g'}}) = \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_{g'}})$, hence σ is a *synchronization* action (otherwise t would not be enabled). By definition of composition $\|$, the guard of t is $\gamma_{\mathcal{A}_{e_p}} \wedge \gamma_{\mathcal{A}_{e_r}}$ (there are no update functions by Definition 5). By Definition 5 it holds $t \in T_{\mathcal{A}^{\mathcal{K}}_{e_{g'}}}$. By Lemma 5 transition t is not enabled in $\mathcal{A}^{\mathcal{K}}_{e_{g'}}$ because $\gamma_{\mathcal{A}_{e_r}}$ evaluates to false under the boolean assignment $d_{0_{p'}}$ (i.e. $d_{0_{p'}} \not\models \gamma_{\mathcal{A}_{e_r}}$), whilst it evaluates to true or unknown under the boolean assignment d_{0_p} (i.e. $d_{0_p} \models \gamma_{\mathcal{A}_{e_r}}$). A contradiction is reached because by hypothesis $p \preceq p'$; i.e. p' is obtained from p by switching some feature from *false* to $\bullet$ while retaining satisfiability of φ (recall that in case of unknown value the transition is by default permitted by the mpc). □

An important consequence of Corollary 2 is that generally, for safety requirements (e.g. energy problems), given an FA, it is not necessary to compute the mpc for each of its valid products. Indeed, it suffices to compute some mpc of certain valid products defined below. The relation $\asymp$ introduced below is trivially an equivalence relation.

Definition 8 (Representative products). *Let $\mathcal{A}_f$ be an FA, let $p, p' \in [\![\varphi_{\mathcal{A}_f}]\!]$, let $ME(\mathcal{A}_f)$ be the set of maximal elements of $([\![\varphi]\!], \preceq)$, let*

$$p \asymp p' \quad \textit{iff} \quad \mathit{Forbidden}(p) = \mathit{Forbidden}(p')$$

and let the representative products *$p_c \in RP(\mathcal{A}_f)$ be the representatives of the equivalence classes of $ME(\mathcal{A}_f)/\asymp$.*

Intuitively, a representative product represents all the maximal elements in $\preceq$ that have the same set of forbidden features.

Example 1. Consider the feature constraint $\varphi_{\mathcal{A}_f} = (\bigvee_{i \in 1 \ldots 20} f_i) \wedge ((f_{21} \wedge \neg f_{22}) \vee (f_{22} \wedge \neg f_{21}))$, and its set of over one million products $[\![\varphi_{\mathcal{A}_f}]\!]$. The maximal elements of $([\![\varphi_{\mathcal{A}_f}]\!], \preceq)$ are 40. Formally, $ME(\mathcal{A}_f) = \{p \mid p(f_{22}) = true, p(f_{21}) = false, \exists i \in 1 \ldots 20. p(f_i) = true \text{ and } \forall j \neq i. p(f_j) = \bullet\} \cup \{p \mid p(f_{22}) = false, p(f_{21}) = true, \exists i \in 1 \ldots 20. p(f_i) = true \text{ and } \forall j \neq i. p(f_j) = \bullet\}$. There are two representative products: p_{c_1} and p_{c_2} such that $\mathit{Forbidden}(p_{c_1}) = \{f_{21}\}$ and $\mathit{Forbidden}(p_{c_2}) = \{f_{22}\}$.

A special automaton $\mathcal{K}$ is now defined as the union of the mpc of the representative products, where union is the standard operation on automata.

Definition 9. *Le $\mathcal{A}_f$ be a (P)FA, let $\mathcal{A}_{e_r}$ be an EFA requirement, for each $p \in [\![\varphi_{\mathcal{A}_f}]\!]$ let $\mathcal{A}_{e_p}$ be the corresponding EFA computed in Definition 5 and $\mathcal{A}^{\mathcal{K}}_{e_{g(p)}}$ the corresponding mpc for requirement $\mathcal{A}_{e_r}$. The special automaton $\mathcal{K}$ is defined as:*

$$\mathcal{K} = \bigcup_{p \in RP(\mathcal{A}_f)} \mathcal{A}^{\mathcal{K}}_{e_{g(p)}}$$

The representative products fully characterise the mpc of each valid product of $\mathcal{A}_f$ as refinement of $\mathcal{K}$, as guaranteed by the following theorem.

Theorem 3 (Refinement of product line). *Given $\mathcal{K}$ from Definition 9, it holds that:*

$$\forall\, p \in [\![\varphi_{\mathcal{A}_f}]\!] : \mathscr{L}(\mathcal{A}^{\mathcal{K}}_{e_{g(p)}}) \subseteq \mathscr{L}(\mathcal{K})$$

Proof. The statement follows from Definition 9 and Corollary 2. □

An immediate application of Theorem 3 to the energy problem is stated in the following corollary.

Corollary 3. *Given $\mathcal{K}$ from Definition 9 and $\mathcal{A}_{e_r}$ from Definition 6. The following hold*

1. *if the* safety *energy problem has a solution on $\mathcal{K}$ then it has solution on all $proj_p(\mathcal{A}_f), p \in [\![\varphi_{\mathcal{A}_f}]\!]$.*
2. *if the* reachability *energy problem has a solution for a given $proj_p(\mathcal{A}_f), p \in [\![\varphi_{\mathcal{A}_f}]\!]$, then it has a solution for all $proj_{p'}(\mathcal{A}_f), p' \in [\![\varphi_{\mathcal{A}_f}]\!], p \preceq p'$.*

Proof. First point follows from Theorem 3 and Theorem 2, whilst the second follows from Theorem 2 and Lemma 5. □

Example 2. Continuing Example 1, if the safety energy problem has a solution on the corresponding $\mathcal{K}$, then it will have a solution for all valid products. Hence, to decide if the whole product line is safe it suffices to synthesise the mpc for only two representative products, instead of synthesising the mpc of all $\approx 10^6$ valid products.

Note that Corollary 3 can be applied to other safety or reachability problems, provided they can be converted into an equivalent mpc synthesis problem. Moreover, in case Corollary 3.1 is not satisfied, it is still possible to check which products are safe without computing all products' controllers. Indeed, a simple algorithm that exploits the partial order is sketched. The algorithm visits iteratively in a top-down fashion the DAG induced by the partial order of products, starting from the maximal elements. In case the mpc of a product (i.e. a node of the DAG) is safe, then all its sub-products are detected to be safe, without computing their controllers. Thus this node can be pruned by the algorithm.

7 Related Work and Conclusion

Some recent work in the application of SCT to SPLE is discussed below and compared to the results presented in this paper. Final remarks and future work follow.

7.1 Related Work

SCT and SPLE have been applied to Contract automata (CA) [20]. CA are a formalism for specifying, composing and synthesising a safe orchestration of service contracts. Safety is guaranteed for specific properties of contract agreement. The orchestration synthesis of CA is rendered as the synthesis of the mpc in SCT. Contracts can declare both optional and necessary requirements, interpreted as controllable and uncontrollable actions from SCT [21], thus adding a first layer of behavioural variability.

Featured Modal Contract Automata (FMCA) add an additional layer of structural variability to CA. The associated toolkit FMCAT is used for modelling and analysing contract-based dynamic service product lines [22–24]. FMCAT imports feature models and their valid products from FeatureIDE [25]. An FMCA defines a feature constraint (cf. [26, 27]) over service actions, and declares urgent, greedy and lazy necessary service requests, reflecting a decreasing level of criticality. Features are identified as service actions and each FMCA represents a behavioural product line of services equipped with feature constraints. The mpc synthesis deals with both feature constraints and the different necessary requests.

In this paper, such results are more general and consider any possible quantitative requirement expressed as EFA whilst the product line is described as a PFA. FA and EFA are widely adopted formalisms endowed with off-the-shelf tools that have been applied to industrial case studies.

Timed Service Contract Automata (TSCA) have been studied in [28], and are an extension of CA with real-time constraints. Similarly to this paper, the mpc (orchestration) synthesis has been rendered as a strategy synthesis on a timed game solving both reachability and safety problems. However, only the orchestration can be synthesised while here the synthesis problem is considered more generally.

SCT has been applied to SPLE in [29] through an implementation in the tool `CIF3` [19]. The `CIF3` toolset is used to synthesise all the valid products of a family composed of behavioural components and behavioural requirements rendered as FSA. Additional constraints are generated from an attributed feature model and other behavioural requirements (e.g. guards on events, state invariants). Product line aspects are expressed as behavioural requirements. Whilst [29] focusses on aspects of implementation, in this paper a rigorous approach has been proposed and the translation from FA to EFA has been proved to be correct. Moreover, instead of synthesising the mpc for each product separately, a technique has been proposed for exploiting common aspects different products share, leading to a potential improvement in verification performance.

A product line is expressed through a set of Modal Sequence Diagrams (MSD) endowed with a feature model in [30], in the so-called Scenario-based Product Line Specification. The absence of inconsistencies in the different variants is checked by expressing the MSD as a featured game, similar to FA. An algorithm to synthesise a winning strategy is proposed, by exploiting commonalities between the different variants. Instead of checking inconsistencies in games, this

paper formally proves an encoding such that it is possible to synthesise an mpc for a generic requirement. Moreover, it has been showed how the controller synthesis solves both reachability and safety games.

Similarly to the results presented in this paper, in [31, 32] quantitative properties of product lines are studied. In particular, in [31] the algorithm for computing limit average cost of Weighted Transition Systems (WTS) is adapted to a product line framework, whilst in [32] the problem of a dynamically changing product line with quantitative requirements is studied using statistical model checking techniques. However, SCT has not been considered, which is one of the main aspects addressed in this paper.

7.2 Conclusion

In this paper a formal mapping from (P)FA to EFA has been proved. The main outcome of this translation is the synthesis of an mpc enforcing a generic requirement on an SPL. It has been proved that both reachability and safety energy problems can be solved at once with one mpc synthesis. The synthesis is not computed for all products, potentially exponential in number. Instead, a partial order relation among products and automata is exploited. A potentially exponential gain in performance can thus be obtained. Indeed, an example shows how to check such safety requirements by only considering a small fraction of products (i.e. 2 in the best case) instead of the initial $\approx 10^6$ products.

Future work concerns implementing the proposed translation as a plugin to allow tools for (P)FA and FTS to interact with tools for SCT of EFA, as for example `CIF3`.

Previously, in the CA framework [20] the mpc synthesis enforcing a specific quantitative requirement (called weak agreement) has been formalised as a network flow problem, to which optimisation algorithms have been applied. Moreover, semi-controllable transitions have been introduced in [22] to define particular transitions controllable under specific global conditions. Such semi-controllable transitions are useful when the plant represents a set of entities exchanging resources, as for the case of CA. Further research needs to be pursued to generalise both results on network flow problems and semi-controllable transitions to synthesise an mpc for a generic requirement.

Acknowledgements The author would like to thank Maurice ter Beek for many useful discussions and the unknown reviewers for the useful comments.

References

1. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
2. D. L. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, SE-2(1):1–9, March 1976.
3. D. Benavides, S. Segura, and A. Ruiz-Cortes. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615 – 636, 2010.

4. A. Classen, P. Heymans, P.Y. Schobbens, A. Legay, and J.F. Raskin. Model checking lots of systems: Efficient verification of temporal properties in software product lines. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 335–344, New York, NY, USA, 2010. ACM.
5. M.H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints. *J. Log. Algebr. Meth. Program.*, 85(2):287–315, 2016.
6. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, New York, NY, USA, 2006.
7. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller Synthesis for Timed Automata. *IFAC Proceedings Volumes*, 31(18):447–452, 1998.
8. O. Malerand A. Pnueli and J. Sifakis. On the Synthesis of Discrete Controllers for Timed Systems. In E.W. Mayr and C. Puech, editors, *Proceedings 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.
9. Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *Formal Modeling and Analysis of Timed Systems*, pages 33–47, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
10. Krishnendu Chatterjee and Laurent Doyen. Energy parity games. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 599–610, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
11. A. Classen, M. Cordy, P. Y. Schobbens, P. Heymans, A. Legay, and J. F. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to ltl model checking. *IEEE Transactions on Software Engineering*, 39(8):1069–1089, Aug 2013.
12. U. Fahrenberg and A. Legay. Featured weighted automata. In *5th IEEE/ACM International FME Workshop on Formal Methods in Software Engineering, FormaliSE@ICSE 2017, Buenos Aires, Argentina, May 27, 2017*, pages 51–57, 2017.
13. M. H. ter Beek, E. P. de Vink, and T. A. C. Willemse. Family-based model checking with mCRL2. In *International Conference on Fundamental Approaches to Software Engineering*, pages 387–405. Springer, 2017.
14. M. Cordy, A. Classen, P. Heymans, P.Y. Schobbens, and A. Legay. Provelines: a product line of verifiers for software product lines. In *Proceedings of the 17th International Software Product Line Conference co-located workshops*, pages 141–146. ACM, 2013.
15. A. Classen, P. Heymans, P.Y. Schobbens, and A. Legay. Symbolic model checking of software product lines. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 321–330, New York, NY, USA, 2011. ACM.
16. Y.L. Chen and F. Lin. Modeling of discrete event systems using finite state machines with parameters. In *Proceedings of the 2000 IEEE International Conference on Control Applications.*, pages 941–946. IEEE, 2000.
17. M. Skoldstam, K. Akesson, and M. Fabian. Modeling of discrete event systems using finite automata with variables. In *46th IEEE Conference on Decision and Control, 2007*, pages 3387–3392. IEEE, 2007.
18. L. Ouedraogo, R. Kumar, R. Malik, and K. Akesson. Nonblocking and safe control of discrete-event systems modeled as extended finite automata. *IEEE Transactions on Automation Science and Engineering*, 8(3):560–569, 2011.

19. D.A. van Beek, W.J. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J.M. van de Mortel-Fronczak, and M.A. Reniers. CIF 3: Model-Based Engineering of Supervisory Controllers. In E. Ábrahám and K. Havelund, editors, *Proceedings 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *LNCS*, pages 575–580. Springer, 2014.
20. D. Basile, P. Degano, and G.L. Ferrari. Automata for Specifying and Orchestrating Service Contracts. *Log. Meth. Comput. Sci.*, 12(4:6):1–51, 2016.
21. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, 1987.
22. D. Basile, M.H. ter Beek, F. Di Giandomenico, and S. Gnesi. Orchestration of Dynamic Service Product Lines with Featured Modal Contract Automata. In *SPLC*, pages 117–122. ACM, 2017.
23. D. Basile, F. Di Giandomenico, and S. Gnesi. FMCAT: Supporting Dynamic Service-based Product Lines. In *SPLC*, pages 3–8. ACM, 2017.
24. Davide Basile, Maurice H. ter Beek, and Stefania Gnesi. Modelling and analysis with featured modal contract automata. In *Proceeedings of the 22Nd International Systems and Software Product Line Conference - Volume 2*, SPLC '18, pages 11–16, New York, NY, USA, 2018. ACM.
25. J. Meinicke, T. Thüm, R Schröter, F. Benduhn, T. Leich, and G. Saake. *Mastering Software Variability with FeatureIDE*. Springer, Cham, Switzerland, 2017.
26. M. Mannion. Using First-Order Logic for Product Line Model Validation. In G.J. Chastek, editor, *Proceedings 2nd International Software Product Lines Conference (SPLC'02)*, volume 2379 of *LNCS*, pages 176–187. Springer, 2002.
27. D.S. Batory. Feature Models, Grammars, and Propositional Formulas. In J.H. Obbink and K. Pohl, editors, *Proceedings 9th International Software Product Lines Conference (SPLC'05)*, volume 3714 of *LNCS*, pages 7–20. Springer, 2005.
28. Davide Basile, Maurice H. ter Beek, Axel Legay, and Louis-Marie Traonouez. Orchestration synthesis for real-time service contracts. In *Verification and Evaluation of Computer and Communication Systems - 12th International Conference, VECoS 2018, Grenoble, France, September 26-28, 2018, Proceedings*, pages 31–47, 2018.
29. M.H. ter Beek, M.A. Reniers, and E.P. de Vink. Supervisory Controller Synthesis for Product Lines Using CIF 3. In T. Margaria and B. Steffen, editors, *Proceedings 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*, volume 9952 of *LNCS*, pages 856–873. Springer, 2016.
30. M. Cordy, J.M. Davril, J. Greenyer, E. Gressi, and P. Heymans. All-at-once-synthesis of controllers from scenario-based product line specifications. In *Proceedings of the 19th International Conference on Software Product Line*, pages 26–35. ACM, 2015.
31. Rafael Olaechea, Uli Fahrenberg, Joanne M. Atlee, and Axel Legay. Long-term average cost in featured transition systems. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, pages 109–118, New York, NY, USA, 2016. ACM.
32. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. A framework for quantitative modeling and analysis of highly (re)configurable systems. *IEEE Transactions on Software Engineering*, 2018.