Parallel Behavior Composition for Manufacturing

Paolo Felli
University of Nottingham, UK
paolo.felli@nottingham.ac.uk

Brian Logan
University of Nottingham, UK
bsl@cs.nott.ac.uk

Sebastian Sardina
RMIT University, Australia
sebastian.sardina@rmit.edu.au

Abstract

A key problem in the manufacture of highly-customized products is the synthesis of controllers able to manufacture any instance of a given product type on a given production or assembly line. In this paper, we extend classical AI behavior composition to manufacturing settings. We first introduce a novel solution concept for manufacturing composition, *target production processes*, that are able to manufacture multiple instances of a product simultaneously in a given production plant. We then propose a technique for synthesizing the largest target production process, together with an associated controller for the machines in the plant.

1 Introduction

Manufacturing companies are increasingly faced with demands for variable volumes of high quality customised products, produced rapidly and at low cost [Foresight, 2013]. One way of meeting such demands is through increased automation, and in particular allowing production control software greater autonomy in determining how products will be manufactured. The application of autonomous systems in manufacturing has the potential to increase productivity, flexibility and reliability, add value, and compensate for an ageing skilled workforce. However, the complexity of such "autonomous manufacturing systems" has so far prevented their widespread adoption. A key challenge in realising the potential of autonomous manufacturing is moving from human-authoring of the production control software that specifies how a particular product should be made, to the *automated synthesis of controllers that are able to manufacture any instance of a given product type on a particular production or assembly line*.

The problem of synthesising complex (virtual) systems from simple existing modules has been addressed in AI, where it is referred to as the *behavior composition problem* [De Giacomo *et al.*, 2013]. The composition task involves synthesizing a controller to coordinate a set of available behavior modules (e.g., automatic lights or blinds, TVs, microwaves, etc.) so as to implement a novel target behavior module (e.g., a smart house system). Standard behavior composition assumes a single sequential execution of the target behavior module. While adequate for domains such as web-services [Calvanese *et al.*, 2008], these approaches are not applicable to the manufacturing domain.

In manufacturing, the target represents a process specified by a *production recipe*, which lists the steps necessary to manufacture items of a particular product type, and the order in which those steps should be executed. The recipe encompasses all variations of the product type, e.g., variations in size, color or material, and decisions about how a particular item should be manufactured are made at run-time, based on order information associated with the item, e.g., whether the item is to be blue or green. The steps in a recipe are enacted by *production resources*, e.g., welding machines, painting machines, etc. For example, in garment manufacture, a production recipe may specify how to make shirts of various colors and sizes, and the production resources may be fabric cutting, sewing and pressing machines. In general, a recipe consists of multiple steps, and (depending on available production resources) it is often possible to start work on manufacturing another item before the items currently being manufactured have been completed. The aim is to construct a *target production process* that allows many instances of a product type (ideally as many as possible) to be produced at the same time on the available production resources. Critically, such a process must allow *all* variations of the product type specified by the recipe for each instance produced, as the specification of each item (e.g., whether it will be blue or green) is only known at run-time.

In this paper, we extend behavior composition [De Giacomo *et al.*, 2013; Stroeder and Pagnucco, 2009; Lustig and Vardi, 2009] to manufacturing settings. We formalise both production recipes and production resources, and propose a technique for synthesizing a target production process and controller capable of orchestrating the behaviors of the production resources to produce multiple instances of a product type in accordance with the recipe. To the best of our knowledge, our work is the first to consider the synthesis of controllers for multiple concurrent instances of a target process.

2 Production Recipes and Cycles

We begin by presenting the notions used to specify how items of a particular product type are manufactured. A production recipe captures the basic finite process required to manufac-

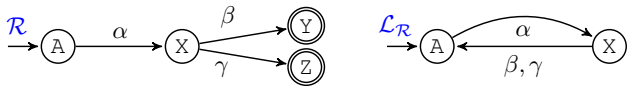

Figure 1: A simple product recipe $\mathcal{R}$ and its production cycle.

ture any instance (e.g., a blue shirt of size XL) of a particular product type (e.g., shirts).

Definition 1. A ***production recipe*** is a tuple $\mathcal{R} = (L, l_0, A, \lambda_\mathcal{R}, F)$ where *(i)* L is finite set of states and $l_0 \in L$ is the initial state; *(ii)* A is a finite set of actions; *(iii)* $\lambda_\mathcal{R} : L \times A \to L$ is an acyclic transition function with all final states being sink states; *(iii)* $F \subseteq L$ is the set of final states. ■

A production recipe may be arbitrarily large and include options for which action to perform in a state that encode the different ways in which a product can be manufactured (e.g., due to variations within a product type). For example, once the fabric of a shirt has been cut, it may be dyed green, red, or blue: which color should be used for a particular product item is decided at run-time and is outside the model (e.g., by a human operator, or determined by information carried by the item itself, e.g., in the form of a radio-frequency identification (RFID) tag specifying the color). If a recipe is action-deterministic, i.e., only one action is available in each state of $\mathcal{R}$, this indicates that there is only one possible way of manufacturing the product type.

We assume that recipes are acyclic and "terminating": the manufacture of every product item requires a bounded number of steps.[1] For example, producing an item of the product type specified by the recipe $\mathcal{R}$ depicted in Figure 1 requires performing operation α (e.g., cut fabric) followed by either action β (e.g., dye blue) or γ (e.g., dye green), after which the item is deemed manufactured (states Y and Z are final). Hence $\mathcal{R}$ specifies two different manufacturing alternatives or product variations. As explained above, whether β or γ will be performed for a particular item is decided at run-time, externally to the recipe.

While recipes specify how *one* item of a product type is to be manufactured to completion, a production plant aims to manufacture many instances of a product simultaneously, over and over, in principle forever. This continuous process is captured through the notion of production cycle.

Definition 2. Given a product recipe $\mathcal{R} = (L, l_0, A, \lambda_\mathcal{R}, F)$, the ***production cycle*** induced by $\mathcal{R}$ is a tuple $\mathcal{L}_\mathcal{R} = (L \setminus F, l_0, A, \lambda)$ such that $\lambda(l) = \lambda_\mathcal{R}(l)$ for all $l \in L \setminus F$, and $\lambda(l) = l_0$ for each $l \in F$. ■

Intuitively, a production cycle represents the repeated execution of a recipe. To represent the completion of one item and the beginning of the next item of the product type, we add a transition from the final states to the initial state of the recipe. When an item "returns" to the initial state (through a sequence of transitions within its production cycle), it is considered as "completed," and the cycle can be "restarted" representing a *new* item of the product type entering the production line. For instance, the production cycle $\mathcal{L}_\mathcal{R}$ in Figure 1 is induced by the recipe $\mathcal{R}$.

When $\mathcal{L}$ is a production cycle, we denote by $\mathcal{L}_-$ the extended cycle obtained from $\mathcal{L}$ by adding extra self-loops with the special action $- \notin A$ in every state. The $-$ ("no-op") transition is used to represent a "pause" step.

We will use the notion of ***traces*** in the usual way to represent legal runs of recipes and cycles. For example, a ***trace*** of a production cycle $\mathcal{R}$ is a finite sequence $\pi = l^0 \boldsymbol{a}^1 l^1 \boldsymbol{a}^2 \cdots \boldsymbol{a}^\ell l^\ell$ (or just $\pi = l^0 \xrightarrow{\boldsymbol{a}^1} l^1 \xrightarrow{\boldsymbol{a}^2} \cdots \xrightarrow{\boldsymbol{a}^\ell} l^\ell$), such that $\langle l^{j-1}, \boldsymbol{a}^j, l^j \rangle \in \lambda$ for all $1 \le j \le \ell$. When π is a trace as above, $last(\pi)$ denotes its last state t^ℓ. Finally, unless stated otherwise, we assume that traces start from the initial state (in this case, $l^0 = l_0$).

3 Production Plants

Production recipes are enacted by a *production plant*, consisting of production resources and their associated capabilities (e.g., cutting, sewing, knitting, and pressing machines in a clothing factory). Following [De Giacomo *et al.*, 2013], we model those capabilities as ***available behaviors*** of the form $\mathcal{B} = (B, b_0, A, R)$ where B is finite set of states, $b_0 \in B$ is the initial state, A is the set of actions (we assume $- \notin A$), and $R \subseteq B \times A \times B$ is $\mathcal{B}$'s transition relation. Behaviors may be nondeterministic, and so are only *partially controllable*, i.e., once a behavior is instructed to execute an action, its evolution cannot be controlled. For example, a painting machine can non-deterministically evolve to a state signaling out-of-paint after painting an item.

A production plant is composed of $m \ge 1$ available behaviors, and is formally captured as the synchronous product of those behaviors. (We assume that the production plant is fixed, as it represents the capabilities of the available production resources.) From now on, we denote by indx the set of vectors $\boldsymbol{k} \in (\{1, \ldots, m\})^m$ in which the components $\boldsymbol{k}_i$ are pairwise distinct. As is customary, when X is a set, $n \ge 1$, $X^k = X \times \cdots \times X$ denotes the k cross-product of X; and when t is a tuple of size k, we use t_i to denote its i-th component, for each $1 \le i \le k$. When A is a set of actions, we use $A_-^k = (A \cup \{-\})^k \setminus (\{-\})^k$ to denote the set of vectors of k actions extended to include the distinguished no-op $-$ action.

Definition 3. Given a set of $m \ge 1$ behaviors $\{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$, with $\mathcal{B}_i = (B_i, b_{0i}, A, R_i)$ for each $1 \le i \le m$, the ***production plant system*** is a tuple $\mathcal{S} = (S, s_0, A_-^m, \rho)$ where:

- $S = B_1 \times \cdots \times B_m$ is the set of states;
- $s_0 = \langle b_{01}, \ldots, b_{0m} \rangle$ is $\mathcal{S}$'s initial state;
- $\rho \subseteq S \times A_-^m \times \mathsf{indx} \times S$ is $\mathcal{S}$'s transition relation such that $\langle b_1, \ldots, b_m \rangle \xrightarrow{\boldsymbol{a}\ \boldsymbol{k}} \langle b'_1, \ldots, b'_m \rangle$ in ρ iff for each $1 \le i \le m$, if $\boldsymbol{a}_i = -$, then $b'_i = b_i$; otherwise $b'_{\boldsymbol{k}_i} \in R_k(b_k, \boldsymbol{a}_{\boldsymbol{k}_i})$. ■

Intuitively, a transition $\langle b_1, \ldots, b_m \rangle \xrightarrow{\boldsymbol{a}\ \boldsymbol{k}} \langle b'_1, \ldots, b'_m \rangle$ in $\mathcal{S}$ captures the delegation of each action $\boldsymbol{a}_j$ (including possibly no-op actions) to the available behavior $\mathcal{B}_{\boldsymbol{k}_j}$, which evolves

[1] It is straightforward to add constructs specifying the bounded repetition of a subprocess, e.g., to model the re-trying of a process (clean subprocess) until success (product sensed clean), and exiting after some number of repetitions (discard product).

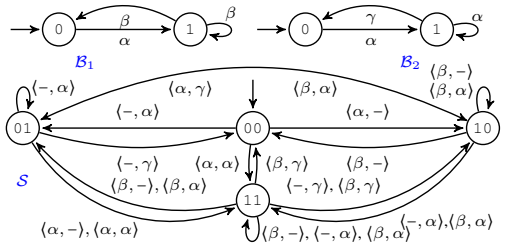

Figure 2: A plant $\mathcal{S} = \langle \mathcal{B}_1, \mathcal{B}_2 \rangle$. For compactness, $\langle \alpha, \gamma \rangle$ in $\mathcal{S}$ stands for both $\langle \alpha, \gamma \rangle\langle 1, 2 \rangle$ and $\langle \gamma, \alpha \rangle\langle 2, 1 \rangle$.

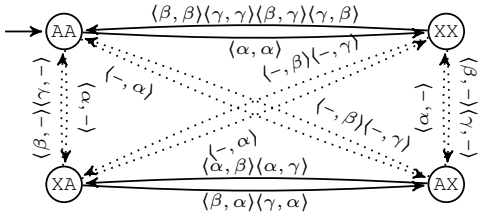

Figure 3: The 2-pcycles $\mathcal{L}^2$ (solid edges only) and $\mathcal{L}^2_-$ (solid and dotted edges) for production cycle $\mathcal{L}_\mathcal{R}$ from Figure 1.

from current state $b'_{\boldsymbol{k}_j}$ to successor state $b'_{\boldsymbol{k}_j}$. As an example, Figure 2 shows two available behaviors and the resulting production plant system $\mathcal{S}$.

As for production cycles, traces of production plant systems are sequences of the form $s^0 \xrightarrow{\boldsymbol{a}^1} \cdots \xrightarrow{\boldsymbol{a}^\ell} s^\ell$, which we refer to as ***histories***. We use $\mathcal{H}$ to denote the set of histories of $\mathcal{S}$.

3.1 Plant Controllers

Intuitively, a production plant is operated as follows. At any point in time, the "user" or "operator" of the plant—which may be human, software, or a hybrid of the two—requests the execution of up to m domain actions, where m is the number of available behaviors in the plant. A *controller* then delegates each domain action to an available behavior in the plant that is capable of handling the action in the current situation. The designated behaviors then execute their assigned actions and evolve as specified by their transition relation, yielding the new state of the plant, from where a new request is issued and so on. The objective is to build a controller that *always* fulfills the user's requests by appropriate delegation of domain actions.

Definition 4. Given a production plant system $\mathcal{S}$ with $m \geq 1$ available behaviors, a ***plant controller*** (or ***controller***) is a function

$$P : \mathcal{H} \times A_-^m \to \mathsf{indx}$$

which delegates a set of actions $\boldsymbol{a} \in A_-^m$ to available behaviors in $\mathcal{S}$: if $P(h, \boldsymbol{a}) = \boldsymbol{k}$, then P delegates action $\boldsymbol{a}_i$ to available behavior $\mathcal{B}_{\boldsymbol{k}_i}$, for $i \in \{1, \ldots, m\}$. ■

Note that our notion of controller extends that of De Giacomo *et al.* (2013) to multiple actions, as we assume a production plant can perform more than one operation at a time, due to the concurrent operation of production resources. However, since each resource can carry out one operation at a time, in a plant with m resources, one can execute up to m actions simultaneously.

4 Target Production Processes

With the notions of production recipe and production plant system in hand, we are now ready to present one of the main contributions of the paper, namely *target production processes*, as a solution concept for manufacturing composition. Since production plants aim to manufacture not one, but several items of a given product type simultaneously, the natural question that arises is: *what is the actual "target" process that a production plant needs to support?*

Unlike in AI behavior composition (e.g., [De Giacomo *et al.*, 2013]), the desired manufacturing process to be "implemented" in the production plant is neither arbitrary nor given as part of the input. Intuitively, the desired process involves the production of *multiple items* of a given product type, all conforming to the production cycle induced by the production recipe specifying how to manufacture items of the product type. We shall call such a process the *target production process* (TPP), or just the *target*. However it is not clear exactly what this process ought to be, let alone how to compute it, so we first spell out the requirements. The first requirements is:

R1. A TPP should allow the manufacture of *multiple* product items *simultaneously*. When a TPP allows the manufacture of exactly n items in the plant at a given time, we say it is an n-TPP.

For example, while a shirt is being dyed by one production resource, the fabric of another shirt—another product instance—is being cut by another resource. One way of meeting **R1** would be to define the production process to be the synchronous execution of multiple copies of the product production cycle, formally captured as the synchronous execution of n production cycles $\mathcal{L}$ or $\mathcal{L}_-$ (see Figure 3):

Definition 5. Given a production cycle $\mathcal{L} = (L, l_0, A, \lambda)$ and $n \geq 1$, the ***n-pcycle*** of $\mathcal{L}$ is a tuple $\mathcal{L}^n = (T, t_0, A^n, \delta)$ where:

- $T = L^n$ is the set of states of $\mathcal{L}^n$. When $t = \langle l_1, \cdots, l_n \rangle \in T$, we use $st_i(t) = l_i$ to denote the i-th component of t;
- $t_0 = \langle l_0, \cdots, l_0 \rangle$ is the initial state of $\mathcal{L}^n$; and
- $\delta : T \times A^n \to T$ is such that $t' = \delta(t, \boldsymbol{a})$ (or $t \xrightarrow{\boldsymbol{a}} t'$) iff $st_i(t') = \lambda(st_i(t), \boldsymbol{a}_i)$ for each $i \in \{1, \ldots, n\}$. ■

However, taking $\mathcal{L}^n$ (or $\mathcal{L}^n_-$) as the production process is *too demanding*: they embed *all* possible ways in which n product items may be manufactured (including no-op transitions), whereas the production plant may only be able to produce n items concurrently in a particular way. To address this, one could consider extracting the *maximal realizable target* of $\mathcal{L}^n_-$ [Yadav and Sardina, 2012; Yadav *et al.*, 2013] to obtain the aspects of $\mathcal{L}^n_-$ that can be realized in the plant. However the maximal realizable target may restrict some of the production cycles, as the resulting fragments may not account for the complete production cycle. This leads to our second requirement for TPPs:

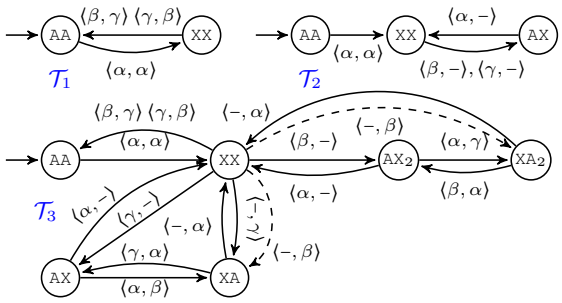

Figure 4: $\mathcal{T}_1$ and $\mathcal{T}_2$ are not 2-targets for $\mathcal{L}$ as in Figure 1, while $\mathcal{T}_3$ is. Dashed edges represent transitions in δ_{UN}.

R2. For each product item in a TPP, the complete product recipe should be available, no matter how the other product items are processed.

That is, when manufacturing each item, it should be possible to produce any variant of the product type (e.g., color of a shirt). In general, which variant a particular item corresponds to is not known at the outset (e.g., whether action β or γ will be performed is known until we reach state XX in $\mathcal{R}$; Figure 1). As a result, $\mathcal{T}_1$ in Figure 4 is *not* a 2-TPP for the recipe $\mathcal{R}$, as it is unable to manufacture two "β items" only (or similarly two γ items): there is no transition $\langle\beta,\beta\rangle$ from state XX. Note that a naive projection from $\mathcal{T}_1$ on either item instance would indeed yield the production cycle of the product—*more is needed* to capture this requirement.

To ensure that TPPs are both *general* and *flexible*, we stipulate the following additional requirements:

R3. The possible evolutions of the TPP may depend on how (nondeterministic) behaviors in the plant happen to evolve, as long as **R2** is met.

This means that unlike in AI behavior composition, we shall accept target production processes that are "nondeterministic."

R4. A TPP may pause certain items and account for different ways in which all of them can be concurrently produced (the decision how exactly left to the user at run-time).

That is, a TPP may include several (though not all) alternative ways of manufacturing n items concurrently, including "pausing" certain items at some point, where this is necessary due to limitations in plant resources. However we restrict the ways that – ("pause") actions may be used:

R5. A TPP should never allow the starvation of any item: every item must eventually be completed.

For example, $\mathcal{T}_2$ in Figure 4 is also not a 2-TPP for $\mathcal{R}$, because the second item instance is commenced but never completely manufactured.

With the five requirements above, we are now ready to introduce the technical machinery to capture TPPs. We start by defining a notion of fragments of an n-pcycle which is extended to incorporate memory and which differentiates between controllable and uncontrollable transitions.

Definition 6. Tuple $\mathcal{T} = (G, g_0, A_{-}^{n}, \delta', \delta'_{\text{UN}})$ is an ***n*-product** of a production cycle $\mathcal{L}$ if (recall $\mathcal{L}_{-}^{n} = (T, t_0, A_{-}^{n}, \delta)$):

- $G = T \times M$ is the set of $\mathcal{T}$'s states, for some arbitrary finite set M, and $g_0 = \langle t_0, m\rangle$ with $m \in M$ is $\mathcal{T}$'s initial state. The set M will be used to encode additional memory;
- $\delta' \subseteq G \times A_{-}^{n} \times G$ is an arbitrary transition relation such that if $\langle t, c\rangle \xrightarrow{\boldsymbol{a}} \langle t', c'\rangle$ in δ', then $t \xrightarrow{\boldsymbol{a}} t'$ in δ; and
- $\delta'_{\text{UN}} \subseteq \delta'$ is arbitrary subset of transitions deemed "uncontrollable" and with the only requirement that if $\langle t, c\rangle \xrightarrow{\boldsymbol{a}} \langle t', c'\rangle \in \delta'_{\text{UN}}$, then there exists another transition $\langle t, c\rangle \xrightarrow{\boldsymbol{a}} \langle t', c''\rangle \in \delta'_{\text{UN}}$ such that $c' \neq c''$. ■

An n-product can therefore account for only some transitions of the corresponding n-pcycle (which means not every interleaving of the underlying production cycles are allowed), while extending it with additional memory (the set M) and distinguishing a subset of transitions as "uncontrollable".

n-products constrained to adhere to requirements **R1**-**R5** form then the basis of our definition of TPPs. In particular, we need to make sure that by pausing a product item we do not violate either **R2** or **R5**. Intuitively, this amounts to checking whether, whenever an item instance i is paused at a state g of TPP $\mathcal{T}$, all actions available for i in g will be eventually available, i.e., manufacture of the item can be fully resumed. We do so in two steps. First, for any action a and instance i, we define all possible acyclic traces of $\mathcal{T}$ in which we either *(i)* resume the item (after a – sequence) by performing exactly a first; or *(ii)* not resume the item at all. This step is captured by the definition of a *closure tree*. Second, we check whether we are able to enforce one of the traces for *(i)*, irrespective of the uncontrollability represented by transitions in δ_{UN}. This step is captured by the definition of *controllable subsets*.

Given a n-product $\mathcal{T}$ as above and a state $g \in G$, the ***closure tree*** in $\mathcal{T}$ for a component $i \in \{1, \ldots, n\}$ and an action $a \in A$, denoted $out_{\mathcal{T}}(g, i, a)$, is the set of all traces $\pi = g^0\boldsymbol{a}^1g^1\boldsymbol{a}^2\cdots\boldsymbol{a}^\ell g^\ell$ of $\mathcal{T}$, with $t^0 = t$, such that:

1. $\boldsymbol{a}_i^j = -$ for all $1 \leq j < \ell$, and either $\boldsymbol{a}_i^\ell = -$ or $\boldsymbol{a}_i^\ell = a$;
2. no transition is taken more than once, i.e., for each g^j and g^q, if $g^j = g^q$ then $\boldsymbol{a}^{j+1} \neq \boldsymbol{a}^{q+1}$ for $1 \leq j, q < \ell$;
3. for each $1 \leq j < \ell$, if there exists a transition $\langle g^j, \boldsymbol{a}, g'\rangle \in \delta$ for some g' and $\boldsymbol{a}$ with $\boldsymbol{a}_i = a$, then $\boldsymbol{a}_i^{j+1} = a$.

Informally, the closure tree $out_{\mathcal{T}}(g, i, a)$ contains all possible finite traces of $\mathcal{T}$, starting from g, such that *(i)* either it is never possible to execute a on the i-th component or only no-op actions are allowed until a is executed; *(ii)* they are acyclic; and *(iii)* whenever a is executable on the i-th item instance at some step j of π, then the next action vector $\boldsymbol{a}^{j+1}$ has a on the i-th component.

Definition 7. Given a set of traces Π of an n-product $\mathcal{T}$, the set of ***controllable subsets*** of Π is the set $contr(\Pi)$ of subsets $\Pi' \subseteq \Pi$ where, for each trace $\pi = g^0\boldsymbol{a}^1\cdots\boldsymbol{a}^\ell g^\ell$ in Π', if for some $1 \leq j \leq \ell$ we have $\langle g^{j-1}, \boldsymbol{a}^j, g^j\rangle \in \delta_{\text{UN}}$, then there is in Π' also a trace $\pi' = g^0\boldsymbol{a}^1\cdots\boldsymbol{a}^j g'^j\cdots$ for every $g'^j \neq g^j$ with $\langle g^{j-1}, \boldsymbol{a}^j, g'^j\rangle \in \delta_{\text{UN}}$ (at least one exists by definition of δ_{UN}).■

Informally, a set of traces $\Pi' \in contr(\Pi)$ is "closed" under uncontrollability of transitions: there is no trace with an uncontrollable transition at some step, unless all possible evolutions also appear in some trace.

Putting it all together, we use controllable subsets of the closure tree to verify that items paused in an n-product $\mathcal{T}$ can always be resumed, so satisfying **R2** and **R5**. For any i, g and a, given the (unique) set of traces $out_{\mathcal{T}}(g, i, a)$, then an element of $contr(out_{\mathcal{T}}(g, i, a))$ is a subset of $out_{\mathcal{T}}(g, i, a)$ such that, by controlling only controllable transitions, we can force the resulting path to be in the subset, irrespective of the nondeterminism of δ_{UN}.

Finally, we say that a vector $\boldsymbol{b} \in A_-^n$ is ***subsumed*** by an action vector $\boldsymbol{a} \in A^n$ iff $\boldsymbol{b}_i = \boldsymbol{a}_i$ or $\boldsymbol{b}_i = -$, for each $1 \leq i \leq n$.

We now have all the elements necessary to define a target production process, i.e., the process we would like to deploy in the production plant.

Definition 8. An n-product $\mathcal{T} = (G, g_0, A_-^n, \delta', \delta'_{\text{UN}})$ is an n ***target production process*** (n-TPP) for a production cycle $\mathcal{L}$ (of a given product recipe $\mathcal{R}$) if: (here $\mathcal{L}^n = (T, t_0, A^n, \delta)$)

- $\mathcal{T}$ is an n-product of $\mathcal{L}$; and
- $\mathcal{T}$ is *complete* and *fair* wrt $\mathcal{L}^n$, that is, for any state $\langle t, m\rangle \in G$ and each transition $t \xrightarrow{\boldsymbol{a}} t'$ in $\mathcal{L}^n$:
  1. $\langle t, m\rangle \xrightarrow{\boldsymbol{b}} \langle t'', m'\rangle$ exists with $\boldsymbol{b}$ subsumed by $\boldsymbol{a}$;
  2. for each $i \in \{1, \ldots, n\}$, if $\boldsymbol{b}_i = -$ then there exists $\Pi \in contr(out_{\mathcal{T}}(\langle t'', m'\rangle, i, \boldsymbol{a}_i))$ such that, for any $g^0 = \langle t'', m'\rangle$ as above and for any trace $\pi = g^0 \boldsymbol{c}^1 g^1 \boldsymbol{c}^2 \cdots \boldsymbol{c}^\ell g^\ell$ in Π we have $\boldsymbol{c}_i^\ell = \boldsymbol{a}_i$: whenever an action $\boldsymbol{a}_i$ is not replicated ($-$ is executed instead), then it is still possible to enforce a trace in $\mathcal{T}$ in which $\boldsymbol{a}_i$ can be finally executed. ■

Hence, an n-TPP is a system whose behavior is a subset of the behaviors of $\mathcal{L}_-^n$, but with possibly additional (bounded) memory. Note we do not require an n-TPP to be the "largest" such system (see later), only that it satisfies **R1**-**R5**.

As an example, consider $\mathcal{T}_3$ in Figure 4. It is easy to verify that it is indeed a 2-TPP of $\mathcal{L}$, since for each trace $\pi \in \mathcal{L}^n$ there exists a trace performing the same actions on both product instances, and which can be extended to account for each possible future evolution of π. For instance, we can perform β (resp. γ) actions on both instances, by pausing one at a time. Of course, other 2-TPPs for $\mathcal{L}$ exist.

4.1 Compositions and Production Controllers

Clearly, not *any* TPP is satisfactory for a given production plant. For instance, TPPs that cannot be "realized" (i.e., implemented) in the plant or that never take advantage of parallelism implicit in the plant's production resources, are not desired. Moreover, the TPP is not a problem input.

From now on, we will restrict ourselves to m-TPPs, where m is the number of behaviors (i.e., available machines) in $\mathcal{S}$, as TPPs with more than m concurrent items will obviously never be implementable in a plant with m machines.

Following [De Giacomo and Sardina, 2007; De Giacomo *et al.*, 2013] we first formally define what it means for a plant controller P to ***realize*** a given trace π of an m-TPP $\mathcal{T}$ in a production plant system $\mathcal{S}$: P is able to delegate each action in the vector (including "$-$" actions), at each step, to a suitable machine in the plant. For example, consider the trace $\pi = \texttt{AA} \xrightarrow{\langle\alpha,\alpha\rangle} \texttt{XX} \xrightarrow{\langle -,\beta\rangle} \texttt{XA}$ of the TPP $\mathcal{T}_3$ from Figure 4 and plant $\mathcal{S}$ from Figure 2. Then π is realized in $\mathcal{S}$ by any controller P such that $P(\texttt{00}, \langle\alpha,\alpha\rangle) = \langle 1, 2\rangle$ and $P(\texttt{00} \xrightarrow{\langle\alpha,\alpha\rangle} \texttt{11} \xrightarrow{\langle -,\beta\rangle}) = \langle 2, 1\rangle$, corresponding to one of the two traces $\texttt{00} \xrightarrow{\langle\alpha,\alpha\rangle\langle 1,1\rangle} \texttt{11} \xrightarrow{\langle -,\beta\rangle\langle 2,1\rangle} \texttt{01}$ and $\texttt{00} \xrightarrow{\langle\alpha,\alpha\rangle\langle 1,1\rangle} \texttt{11} \xrightarrow{\langle -,\beta\rangle\langle 2,1\rangle} \texttt{11}$ in $\mathcal{S}$. Note, however, that while these traces can be extended in $\mathcal{T}_3$ with the action vector $\langle\beta,\alpha\rangle$, this action vector can be replicated in $\mathcal{S}$ only from 11 and not from 01, so any trace of the form $\pi \xrightarrow{\langle\beta,\alpha\rangle} \cdots$ is not realizable.

Then, again as in classical behavior composition, we say that P is a ***plant composition*** of a TPP $\mathcal{T}$ in $\mathcal{S}$ *iff* P realizes all the traces of $\mathcal{T}$ in $\mathcal{S}$. When a composition for a TPP $\mathcal{T}$ exists on a plant $\mathcal{S}$, we say that $\mathcal{T}$ is ***realizable*** in $\mathcal{S}$.

Finally, because, unlike in behavior composition, the target production process is *not* an input to our parallel composition problem, we define the *solution concept* as a pair $\langle\mathcal{T}, P\rangle$, such that $\mathcal{T}$ is an m-TTP for $\mathcal{L}$ and P a composition for $\mathcal{T}$ in $\mathcal{S}$. We refer to these pairs as ***production controllers*** for $\mathcal{L}$ in $\mathcal{S}$.

5 Largest Target Production Process

Production controllers may be evaluated against various metrics (such as average throughput, machine utilisation, load balancing, etc.) to isolate the most efficient ones. As metrics are highly dependent on the application and production plant, in this section we focus on computing production controllers $\langle\mathcal{T}, P\rangle$ where $\mathcal{T}$ is the *largest realizable production process*, that is, which embody (i.e., can "mimic") *all* realizable production processes. It turns out such controllers can be defined in terms of the standard notion of simulation between transition systems [Milner, 1971], and in particular its nondeterministic variant [De Giacomo *et al.*, 2013]: a target production process $\mathcal{T}$ simulates another target production process $\mathcal{T}'$, denoted $\mathcal{T} \succeq \mathcal{T}'$, iff $\mathcal{T}$ can "replicate" every action of $\mathcal{T}'$, step by step. The largest realizable production process for a given production cycle $\mathcal{L}$ and a production plant system $\mathcal{S}$ is threfore the m-TPP $\mathcal{T}$ for $\mathcal{L}$ that *(i)* is realizable in $\mathcal{S}$; and *(ii)* can simulate any other realizable m-TPP $\mathcal{T}'$.

To capture property *(i)*, we shall define our own notion of simulation between the production plant $\mathcal{S}$ and the m-pcycle $\mathcal{L}^m$, which expresses what it means for $\mathcal{S}$ to be able to "mimic" exactly m copies of the production cycle. Moreover, in order to *synthesize* the largest realizable m-TTP, we will also need to compute, as required by Definition 6, the set M encoding arbitrary memory and the set of uncontrollable transitions δ_{UN}. Crucially, both of these can be extracted from the so-called *enacted system*, capturing the joint execution of $\mathcal{L}_-^m$ with the system $\mathcal{S}$.

Given a production cycle $\mathcal{L}$, we define the ***enacted system*** $\mathcal{S}_{\mathcal{L}}^*$ as the synchronous product of $\mathcal{L}_-^m$ and the production plant system $\mathcal{S}$, so that a state of $\mathcal{S}_{\mathcal{L}}^*$ is a pair $\langle t, s\rangle$, where t a state of $\mathcal{L}_-^m$ and s is a state of $\mathcal{S}$. If ρ^* is the transition relation of $\mathcal{S}_{\mathcal{L}}^*$, then we use ρ_{ND}^* to denote the set of nondeterministic transitions of ρ: $s^* \xrightarrow{\boldsymbol{a},\boldsymbol{k}} s^{*\prime}$ is in ρ_{ND}^* *iff* $s^* \xrightarrow{\boldsymbol{a},\boldsymbol{k}} s^{*\prime}$ and $s^* \xrightarrow{\boldsymbol{a},\boldsymbol{k}} s^{*\prime\prime}$ are in ρ^*, for some $s^{*\prime} \neq s^{*\prime\prime}$. We also de-

fine the notion of ***closure tree*** $out_{\mathcal{S}^*_{\mathcal{L}}}(s^*, i, a)$ ***for*** $\mathcal{S}^*_{\mathcal{L}}$ as for n-product (see Section 4), and $contr(out_{\mathcal{S}^*_{\mathcal{L}}}(s^*, i, a))$ is as in Definition 7, but this time with respect to transitions in ρ^*_{ND}. A subset in $contr(out_{\mathcal{S}^*_{\mathcal{L}}}(s^*, i, a))$ is thus a set of traces of the enacted system $\mathcal{S}^*_{\mathcal{L}}$ which can be enforced irrespective of the nondeterminism of the production plant system.

Definition 9. Given an m-pcycle $\mathcal{L}^m = (T, t_0, A^m, \delta)$ for some production cycle $\mathcal{L}$, a ***lazy simulation*** relation of $\mathcal{L}^m$ by $\mathcal{S} = (S, s_0, A^m_-, \rho)$ is a relation $\sigma_+ \subseteq S \times T$ such that $\langle s, t\rangle \in \sigma_+$ implies that for each $\boldsymbol{a}$, if $t \xrightarrow{\boldsymbol{a}} t'$ is in $\mathcal{L}^m$ then $\boldsymbol{b}, \boldsymbol{k}$ exist such that:

(1) there exists a transition $s \xrightarrow{\boldsymbol{b}\,\boldsymbol{k}} s' \in \rho$ and $\boldsymbol{b}$ is subsumed by $\boldsymbol{a}$ — each $\boldsymbol{a}_i$ is either replicated in $\boldsymbol{b}_i$ or replaced by $-$;

(2) for each such s' we have $\langle s', t''\rangle \in \sigma_+$ with $t \xrightarrow{\boldsymbol{b}} t''$ in $\mathcal{L}^m_-$ — any possible new system state s' is in the relation with the state t'', which is unique; and

(3) given t'', s' and $\boldsymbol{b}$ as above, for each index $1 \leq i \leq n$, if $\boldsymbol{b}_i = -$ then there exists a set of traces $\Pi \in contr(out_{\mathcal{S}^*_{\mathcal{L}}}(s^*, i, \boldsymbol{a}_i))$, for $s^* = \langle t'', s'\rangle$, such that for every trace $s^* \xrightarrow{\boldsymbol{c}^1\boldsymbol{k}^1} \langle t^1, s^1\rangle \cdots \xrightarrow{\boldsymbol{c}^\ell\boldsymbol{k}^\ell} \langle t^\ell, s^\ell\rangle$ of $\mathcal{S}^*_{\mathcal{L}}$ in Π, we have *(i)* $\boldsymbol{c}^\ell_i = \boldsymbol{a}_i$ and *(ii)* $\langle s^j, t^j\rangle \in \sigma_+$ for $j \leq \ell$ — the action $\boldsymbol{a}_i$ can always eventually be performed in at least one controllable subset of traces in the enacted system, by visiting only states preserving σ^+. ■

We say that a state s of $\mathcal{S}$ ***lazily simulates*** a state t of $\mathcal{L}^m$, denoted $s \rightsquigarrow t$, iff there exists a lazy simulation relation σ_+ of $\mathcal{L}^m$ by $\mathcal{S}$ with $\langle s, t\rangle \in \sigma_+$. We say that $\mathcal{S}$ lazily simulates $\mathcal{L}^m$, denoted $\mathcal{S} \rightsquigarrow \mathcal{L}^m$, iff $s_0 \rightsquigarrow t_0$. Then $\mathcal{S} \rightsquigarrow \mathcal{L}^m$ captures the fact that $\mathcal{S}$ can always replicate all actions vectors of $\mathcal{L}^m$, but not in a step-by-step fashion, as some product instances may be paused. However, it guarantees the requirements discussed in Section 4.

Once the lazy simulation relation $\rightsquigarrow$ is computed, if $\mathcal{S} \rightsquigarrow \mathcal{L}^m$, we can then build a finite state program, called the ***maximal controller generator*** (CG), that returns, at each step, the set of all possible action vectors that can be delegated to behaviors.

Definition 10. Given $\mathcal{S}$, $\mathcal{L}^m$ and $\mathcal{L}^m_-$ as before, if $\mathcal{S} \rightsquigarrow \mathcal{L}^m$ then the maximal CG for $\mathcal{L}^m$ and $\mathcal{S}$ is the tuple $\mathcal{C} = (W, w_0, A^m_-, \gamma, \omega)$ such that

- $W = \{\langle t, s\rangle \in T \times S \mid s \rightsquigarrow t\}$ and $w_0 = \langle t_0, s_0\rangle$;
- $\gamma \subseteq W \times \mathcal{A}^m \times \mathsf{indx} \times W$ is such that $\langle t, s\rangle \xrightarrow{\boldsymbol{a}\,\boldsymbol{k}} \langle t', s'\rangle \in \gamma$ iff $t \xrightarrow{\boldsymbol{a}} t'$ is in $\mathcal{L}^m_-$, $s \xrightarrow{\boldsymbol{a}\,\boldsymbol{k}} s'$ is in $\mathcal{S}$; and
- $\omega : W \to 2^{(A^m_- \times \mathsf{indx})}$ is the controller function, defined as $\omega(w) = \{\langle \boldsymbol{a}, \boldsymbol{k}\rangle \mid w \xrightarrow{\boldsymbol{a}\,\boldsymbol{k}} w' \text{ is in } \gamma \text{ for some } w'\}$. ■

Given the current state t of $\mathcal{L}^m$ and the current state s of the production plant system, the maximal controller generator returns the set of possible delegations, such that each member $\langle \boldsymbol{a}, \boldsymbol{k}\rangle$ represents the delegation of each action $\boldsymbol{a}_i$ to the behavior specified by $\boldsymbol{k}_i$. A production controller $\langle \mathcal{T}, P\rangle$ is *generated* by $\mathcal{C}$ as follows:

- $\mathcal{T} = (W, w_0, A^m_-, \gamma', \gamma'_{\text{UN}})$ is an m-TTP, and its transition relation $\gamma' : W \times A^m_- \times W$ is such that $w \xrightarrow{\boldsymbol{a}} w'$ is in γ' iff $\langle \boldsymbol{a}, \boldsymbol{k}\rangle \in \omega(w)$;

  γ'_{UN} is the subset of γ' defined as the set of transitions $w \xrightarrow{\boldsymbol{a}} w'$ in γ' such that $w \xrightarrow{\boldsymbol{a}} w''$ is in γ' for some $w' \neq w''$: multiple transitions deriving from the existence of different available delegations of actions to behaviors (i.e. same $\boldsymbol{a}$, different $\boldsymbol{k}$) constitute controllable transitions, whereas multiple transitions deriving from the production plant system's nondeterminism (i.e. same $\boldsymbol{a}$ and $\boldsymbol{k}$) will constitute uncontrollable transitions;

- P is such that for any system history $h = s^0 \xrightarrow{\boldsymbol{a}^1\boldsymbol{k}^0} \cdots \xrightarrow{\boldsymbol{a}^\ell\boldsymbol{k}^\ell} s^\ell$ (with $s^0 = s_0$) and action vector $\boldsymbol{a}$, if $\langle s_0, t_0\rangle \xrightarrow{\boldsymbol{a}^1\boldsymbol{k}^1} \langle s_1, t_1\rangle \xrightarrow{\boldsymbol{a}^2\boldsymbol{k}^2} \cdots \xrightarrow{\boldsymbol{a}^\ell\boldsymbol{k}^\ell} \langle s^\ell, t^\ell\rangle$ is in $\mathcal{C}$ for some $t^1 \cdots t^\ell$, then $P(h, \boldsymbol{a}) = \boldsymbol{k}$ only if $\langle \boldsymbol{a}, \boldsymbol{k}\rangle \in \omega(\langle s^\ell, t^\ell\rangle)$.

$\mathcal{T}$ is obtained by projecting out delegation indexes from transitions, while the corresponding composition P is obtained by selecting, at each step, for a given action $\boldsymbol{a}$, a vector $\boldsymbol{k}$ such that a transition from the current state exists with label $\boldsymbol{a}, \boldsymbol{k}$. Note that this means that the second component in $W \subseteq T \times S$ plays the same role of the arbitrary set M in Definition 6: for the same state in $\mathcal{L}^m$, there may be more than one possible state in the production plant system $\mathcal{S}$, as this is nondeterministic (e.g., states $\langle \texttt{11}, \texttt{AX}\rangle$ and $\langle \texttt{01}, \texttt{AX}\rangle$ in Figure 5).

Theorem 1. Let $\mathcal{C}$ be the maximal CG for $\mathcal{L}^m$ and $\mathcal{S}$. Then *(i)* any P is a composition for $\mathcal{T}$ in $\mathcal{S}$ iff $\langle \mathcal{T}, P\rangle$ is generated by $\mathcal{C}$ and *(ii)* $\mathcal{T}$ is the largest realizable m-TPP for $\mathcal{L}$ in $\mathcal{S}$.

Proof. (Sketch) Let $\mathcal{H}_{\pi,P}$ be the set of histories of $\mathcal{S}$ that may be obtained by running the controller P in $\mathcal{S}$ to match all the actions in a given trace π of an m-TTP $\mathcal{T}$ (see [De Giacomo and Sardina, 2007; De Giacomo *et al.*, 2013]).

(i) ($\Leftarrow$) P is a composition for $\mathcal{T}$, i.e. it realises any trace $\pi = t^0 \xrightarrow{\boldsymbol{a}^1} \cdots$ (recall $t^0 = t_0$). First, we show that for any history $h^\ell = s^0 \xrightarrow{\boldsymbol{a}^0\boldsymbol{k}^0} \cdots \xrightarrow{\boldsymbol{a}^\ell\boldsymbol{k}^\ell} s^\ell$ in $\mathcal{H}_{\pi,P}$ ($s^0 = s_0$) we have $s^j \rightsquigarrow t^j$, for any $1 \leq j \leq |\pi|$ (for $j = 0$ this follows by Definition 10). By induction on ℓ, $h^{\ell+1} \in \mathcal{H}^{\ell+1}_{\pi,P}$ is such that $s^\ell \xrightarrow{\boldsymbol{a}^{\ell+1}\boldsymbol{k}^{\ell+1}} s^{\ell+1}$ is in $\mathcal{S}$ by definition, so $\langle t^\ell, s^\ell\rangle \xrightarrow{\boldsymbol{a}^{\ell+1}\boldsymbol{k}^{\ell+1}} \langle t^{\ell+1}, s^{\ell+1}\rangle$ is in $\mathcal{C}$, $P(h^\ell, \boldsymbol{a}) = \boldsymbol{k}^{\ell+1}$, and from $s^\ell \rightsquigarrow t^\ell$ it follows that $s^{\ell+1} \rightsquigarrow t^{\ell+1}$. Then, for any π and $h \in \mathcal{H}_{\pi,P}$ as above with $|h| < |\pi|$, by the definition of $\rightsquigarrow$ for any action $\boldsymbol{a}$ with $t^\ell \xrightarrow{\boldsymbol{a}^{\ell+1}} t^{\ell+1}$ in $\mathcal{T}$, we have $s^\ell \xrightarrow{\boldsymbol{a}^{\ell+1}\boldsymbol{k}} s^{\ell+1}$ in $\mathcal{S}$. Further, assume $\mathcal{T}$ is not a m-TPP for $\mathcal{L}$. Then by Definition 8, for some trace $\langle t_0, s_0\rangle \xrightarrow{\boldsymbol{b}^1\boldsymbol{k}^1} \cdots \xrightarrow{\boldsymbol{b}^\ell\boldsymbol{k}^\ell} \langle t^\ell, s^\ell\rangle$ we have $t^\ell \xrightarrow{\boldsymbol{a}} t'$ in $\mathcal{L}^m$ such that either (1) no $\langle t^\ell, s^\ell\rangle \xrightarrow{\boldsymbol{b}^{\ell+1}} \langle t^{\ell+1}, s^{\ell+1}\rangle$ exists s.t. $\boldsymbol{b}^{\ell+1}$ is subsumed by $\boldsymbol{a}$, or (2) there exists i s.t. $\boldsymbol{b}_i = -$ and in every set Π in $contr(out_{\mathcal{T}}(\langle t^{\ell+1}, s^{\ell+1}\rangle, i, \boldsymbol{a}_i))$ there is a trace $\langle t^{\ell+1}, s^{\ell+1}\rangle \xrightarrow{\boldsymbol{b}^{\ell+2}} \cdots \xrightarrow{\boldsymbol{b}^{\ell+q}} \langle t^{\ell+q}, s^{\ell+q}\rangle$ and $\boldsymbol{b}^{\ell+q} \neq \boldsymbol{a}_i$. Then it is easy to see how this implies the same for every set Π' in $contr(out_{\mathcal{S}^*_{\mathcal{L}}}(\langle t^{\ell+1}, s^{\ell+1}\rangle, i, \boldsymbol{a}_i))$. Thus we exclude both (1) and (2) as they contradict $s^\ell \rightsquigarrow t^\ell$.

(i) ($\Rightarrow$) We prove that for any trace $t^0 \xrightarrow{\boldsymbol{a}^1} \cdots \xrightarrow{\boldsymbol{a}^\ell} t^\ell$ of $\mathcal{T}$ and history $h = s^0 \xrightarrow{\boldsymbol{a}^1\boldsymbol{k}^1} \cdots \xrightarrow{\boldsymbol{a}^\ell\boldsymbol{k}^\ell} s^\ell \in \mathcal{H}^\ell_{\pi,P}$, if P is a composition for $\mathcal{T}$ then $\langle t^0, s^0\rangle \xrightarrow{\boldsymbol{b}^1\boldsymbol{k}^1} \cdots \xrightarrow{\boldsymbol{b}^\ell\boldsymbol{k}^\ell} \langle t^\ell, s^\ell\rangle \xrightarrow{\boldsymbol{a}\,\boldsymbol{k}} \langle t, s\rangle$ is in $\mathcal{C}$ for $\boldsymbol{k} = P(h, \boldsymbol{a})$. If not, then $\langle \boldsymbol{a}, \boldsymbol{k}\rangle \notin \omega(\langle t^\ell, s^\ell\rangle)$, and, by definition, either there is no $s^\ell \xrightarrow{\boldsymbol{a}\,\boldsymbol{k}} s^{\ell+1}$ in $\mathcal{S}$, no $t \xrightarrow{\boldsymbol{a}^\ell} t'$ in $\mathcal{T}$,

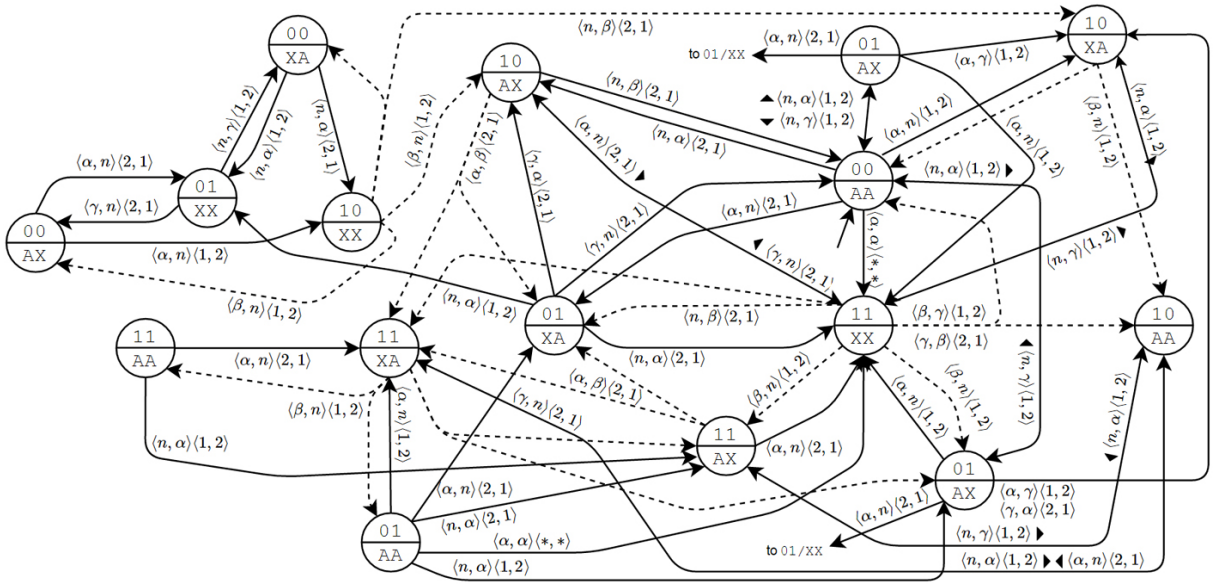

Figure 5: A graphical representation of the maximal CG for the production cycle $\mathcal{L}_{\mathcal{R}}$ in Figure 1 and the production plant system $\mathcal{S}$ in Figure 2. A generated production controller $\langle \mathcal{T}, P \rangle$ is as follows: $\mathcal{T}$ is obtained by removing indexes $\boldsymbol{k}$ from transitions (it is thus unique); given a system history h and an action vector $\boldsymbol{a}$, $P(h, \boldsymbol{a}) = \boldsymbol{k}$ only if a transition labelled with $\boldsymbol{a}, \boldsymbol{k}$ exists from $last(h)$ (hence many P exist). Solid and dashed lines represent controllable and uncontrollable transitions of $\mathcal{T}$, respectively, and $\boldsymbol{n}$ denotes no-op – actions. For instance, by delegating the action vector $\langle \beta, - \rangle$ to behaviors $\langle 1, 2 \rangle$ from state $\langle \texttt{11}, \texttt{XX} \rangle$ (namely β to $\mathcal{B}_1$ and – to $\mathcal{B}_2$), the next state can either be $\langle \texttt{11}, \texttt{AX} \rangle$ or $\langle \texttt{01}, \texttt{AX} \rangle$, depending on the successor state reached by $\mathcal{B}_1$ (see Figure 2). On the other hand, the two transitions labelled with $\langle -, \alpha \rangle$ from $\langle \texttt{01}, \texttt{AA} \rangle$ are controllable: they represent two distinct control choices, to delegate action α to $\mathcal{B}_2$ and to pause $\mathcal{B}_1$ or vice-versa, respectively.

or $s^{\ell+1} \not\rightarrow t^{\ell+1}$. In the first two cases P is not a composition. If the latter holds, it must be the case that in any trace $\langle s^{\ell+1}, t^{\ell+1} \rangle \xrightarrow{\boldsymbol{a}^{\ell+1}\boldsymbol{k}^{\ell+1}} \cdots$ condition (3) of Definition 9 does not hold, hence $\langle \mathcal{T}, P \rangle$ is not a production controller generated by $\mathcal{C}$.

(ii) If $\mathcal{T}' \succ \mathcal{T}$ exists, then there exists a trace π such that $last(\pi) \xrightarrow{\boldsymbol{b}} t$ is in $\mathcal{T}'$ for some t, but $last(\pi) \xrightarrow{\boldsymbol{b}} t'$ is not in $\mathcal{T}$ for any t'. Since $\mathcal{T}'$ is realizable, there exists a composition P' s.t. $h \in \mathcal{H}_{\pi,P'}$ where $P'(h, \boldsymbol{b})$ is defined, and $P(h, \boldsymbol{b})$ is not. So there is no trace $\langle t^0, s^0 \rangle \xrightarrow{\boldsymbol{b}^1\boldsymbol{k}^1} \cdots \xrightarrow{\boldsymbol{b}^\ell\boldsymbol{k}^\ell} \langle t^\ell, s^\ell \rangle$ in $\mathcal{C}$, with $s^0 \xrightarrow{\boldsymbol{b}^1\boldsymbol{k}^1} \cdots \xrightarrow{\boldsymbol{b}^\ell\boldsymbol{k}^\ell} s^\ell = h$ and $t^0 \xrightarrow{\boldsymbol{b}^1} \cdots \xrightarrow{\boldsymbol{b}^\ell} t^\ell = \pi$, s.t. $\langle t^\ell, s^\ell \rangle \xrightarrow{\boldsymbol{b}\,\boldsymbol{k}} \langle t, s \rangle$ for some s and $\boldsymbol{k}$, and t as above. Then by construction either $s^\ell \not\rightarrow t^\ell$ or $s \not\rightarrow t$. □

Computing the lazy simulation relation $\rightsquigarrow$ between $\mathcal{S}$ and $\mathcal{L}^m$ can be done by following the algorithm for the standard simulation relation $\geq$ in [De Giacomo *et al.*, 2013], but checking the controllable closure tree at each step (Definition 9). An algorithm can be derived accordingly. Performance can be improved by reordering actions $\boldsymbol{a}$ according to indexes $\boldsymbol{k}$ in $\mathcal{S}$ so as to substantially reduce the number of transitions.

6 Conclusions

In this paper we consider the parallel behavior composition problem in a manufacturing setting, where many instances of a product are to be manufactured on a production line. We introduced a novel solution concept, *target production processes*, and showed how to generate the largest realizable TPP for a given production plant.

There are subtle conceptual and technical differences between the parallel behavior problem and classical behavior composition in the AI literature. In particular, *multiple* concurrent actions must be delegated to the available behaviors in the plant, rather than just one. We note that this is not the same as *multiple* behavior composition [Sardina and De Giacomo, 2008], in which several target modules are realized in the same shared available system. Multiple behavior composition is equivalent to realizing $\mathcal{L}^m_-$, which we argue is overly demanding. Moreover, the target desired module is *not* given as an input to the problem, but is part of the solution constructed from the specified production recipe and plant.

Our work is just the first step in manufacturing composition. We have defined the problem, and provided a notion of "adequacy" for solutions in the form of TPPs that respect requirements **R1**-**R5**. Further work is needed in order to *refine* m-TPPs to "efficient" manufacturing processes, for example, with respect to average throughput, machine utilization, load balancing, etc. Such optimizations can be done after the TPP has been built, or possibly during synthesis by discriminating between controllers from individual lazy simulation relations σ^j_+. Another avenue we are interested in pursuing, is linking our parallel composition problem with the composition of high-level programs [Sardina and De Giacomo, 2009], at least from a representational perspective.

Acknowledgments
We thank the anonymous reviewers for their helpful feedback. We acknowledge the support of the Australian Research Council (under DP120100332) and the RMIT Foundation (under an International Visiting Fellowship for the second author to visit RMIT University).

References

[Calvanese *et al.*, 2008] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, and Fabio Patrizi. Automatic service composition and synthesis: The Roman Model. *IEEE Data Engineering Bulletin*, 31(3):18–22, 2008.

[De Giacomo and Sardina, 2007] Giuseppe De Giacomo and Sebastian Sardina. Automatic synthesis of new behaviors from a library of available behaviors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1866–1871, 2007.

[De Giacomo *et al.*, 2013] Giuseppe De Giacomo, Fabio Patrizi, and Sebastian Sardina. Automatic behavior composition synthesis. *Artificial Intelligence*, 196:106–142, 2013.

[Foresight, 2013] The future of manufacturing: A new era of opportunity and challenge for the UK. The Government Office for Science, London, 2013. Ref: BIS/13/810.

[Lustig and Vardi, 2009] Yoad Lustig and Moshe Y. Vardi. Synthesis from component libraries. In *Proceedings of the International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 395–409, 2009.

[Milner, 1971] Robin Milner. An algebraic definition of simulation between programs. Technical report, Stanford University, Stanford, CA, USA, 1971.

[Sardina and De Giacomo, 2008] Sebastian Sardina and Giuseppe De Giacomo. Realizing multiple autonomous agents through scheduling of shared devices. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 304–312, 2008.

[Sardina and De Giacomo, 2009] Sebastian Sardina and Giuseppe De Giacomo. Composition of ConGolog programs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 904–910, 2009.

[Stroeder and Pagnucco, 2009] Thomas Stroeder and Maurice Pagnucco. Realising deterministic behaviour from multiple non-deterministic behaviours. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 936–941, 2009.

[Yadav and Sardina, 2012] Nitin Yadav and Sebastian Sardina. Qualitative approximate behavior composition. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA)*, volume 7519 of *Lecture Notes in Computer Science (LNCS)*, pages 450–462. Springer, 2012.

[Yadav *et al.*, 2013] Nitin Yadav, Paolo Felli, Giuseppe De Giacomo, and Sebastian Sardiña. Supremal realizability of behaviors with uncontrollable exogenous events. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1176–1182, 2013.