Parallel Behavior Composition for Manufacturing

Paolo Felli[1] Brian Logan [1] Sebastian Sardina [2]

[1]University of Nottingham, UK

[2]RMIT University, Australia

IJCAI 2016 New York

The problem

- manufacturers increasingly need to produce variable volumes of highly customised products, rapidly and at low cost
- one way to do this is to allow production control software greater autonomy in determining how products will be manufactured
- requires the *automated synthesis of controllers that are able to manufacture any instance of a given product type on a particular production or assembly line*
- **problem:** standard AI behavior composition, e.g., (De Giacomo et al 2013), assumes a single sequential execution of the target behavior module

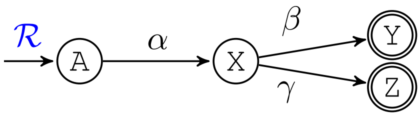
want to manufacture multiple instances of a product in parallel

Key contributions

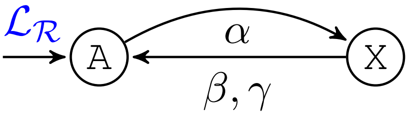
- we extend classical AI behavior composition to manufacturing settings
- we introduce a novel solution concept for manufacturing composition, **target production processes**, that are able to manufacture multiple instances of a product simultaneously in a given production plant
- we propose a technique for synthesizing the **largest target production process**, together with an associated **controller** for the production plant

Production recipes & cycles

- a ***production recipe*** is the (finite) sequence of steps required to manufacture any instance (e.g., a blue shirt of size XL) of a particular product type (e.g., shirts)

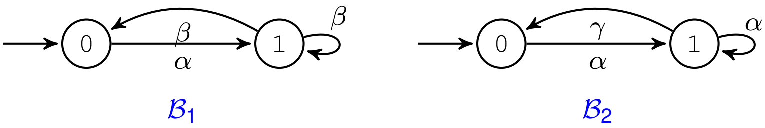


- the ***production cycle*** $L_{\mathcal{R}}$ induced by a recipe $\mathcal{R}$ represents the repeated execution of $\mathcal{R}$



Production plants & controllers

- production recipes are enacted by a production plant, consisting of *production resources* and their associated *capabilities*
- the capabilities of each resource are modelled as an ***available behavior*** — a (possibly nondeterministic) transition system



- a ***production plant system*** $\mathcal{S}$ consisting of *m* resources is the synchronous product of *m* available behaviors (plus "no-op" '$-$' actions)
- a ***plant controller*** is a function which delegates $k \leq m$ actions to *m* available behaviors in the plant

Target production process

- production cycle $\mathcal{L}_{\mathcal{R}}$ tells us how to make items, and the available behaviors tell us what the production plant can do
- but the ***target production process*** specifying how to make multiple items of a product type simultaneously is not given
  - note that the target production process is *not* just the synchronous execution of multiple copies of $\mathcal{L}_{\mathcal{R}}$
  - the production plant may be able to make *n* instances of an item in parallel, but in *only one* (or a small number of) way(s)
- we need a new solution concept for manufacturing composition

TPP requirements

R1. a TPP should allow the manufacture of multiple product items simultaneously

R2. for each product item in a TPP the complete product recipe should be available

R3. the evolution of a TPP may depend on how (nondeterministic) behaviors in the plant happen to evolve

R4. a TPP may "pause" certain items (using "no-op" actions)

R5. a TPP should never allow the starvation of any item

Production controllers

***n*-pcycle** $\mathcal{L}^n$ the **synchronous execution** of *n* copies of the production cycle $\mathcal{L}$

achieves **R1**, but is *too demanding*: represents *all possible ways* in which *n* items can be manufactured

***n*-product** $\mathcal{T}$ fragment of a *n*-pcycle with **memory** & **uncontrollable transitions**

captures that not every interleaving of a *n*-pcycle may be possible in the production plant

***n*-TPP** an *n*-product that is **complete** and **fair** w.r.t. the *n*-pcycle

Given a recipe $\mathcal{R}$ and production plant system $\mathcal{S}$

A **production controller** is a pair $\langle \mathcal{T}, P \rangle$, such that $\mathcal{T}$ **is an *m*-TTP** for *m*-pcycle $\mathcal{L}^m_{\mathcal{R}}$ and *P* **is a plant composition** for $\mathcal{T}$ in $\mathcal{S}$

Largest TPP

- we can compute a production controller $\langle \mathcal{T}, P \rangle$ where $\mathcal{T}$ is the **largest realizable production process**, i.e., which can "mimic" *all* realizable production processes
- define a special notion of **"simulation"** (lazy simulation) between the production plant $\mathcal{S}$ and the *m*-pcycle $\mathcal{L}^m_{\mathcal{R}}$, s.t.:
  - $\mathcal{S}$ "mimics" *m* copies of the production cycle $\mathcal{L}_{\mathcal{R}}$
  - $\mathcal{S}$ can delegate all action vectors of $\mathcal{L}^m_{\mathcal{R}}$, but *not necessarily in a step-by-step fashion*: product instances may be paused
- compute the set encoding memory and the set of uncontrollable transitions in the *m*-product
- both can be extracted from the ***enacted system***: joint execution of *m*-pcycle $\mathcal{L}^m_{\mathcal{R}}$ with production plant $\mathcal{S}$

Main result

- 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 $s_0 \rightsquigarrow t_0$
- if $\mathcal{S} \rightsquigarrow \mathcal{L}^m$ we can build a ***maximal controller generator*** (CG), that returns, at each step, the set of all possible action vectors that can be delegated to behaviors

Theorem

Let $\mathcal{C}$ be the maximal CG for $\mathcal{L}^m$ and $\mathcal{S}$. Then (i) any P is a plant 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 $\mathcal{S}$.

Summary & future work

- we have defined a notion of “adequacy” for solutions in the form of TPPs that respect requirements **R1**-**R5**
- in future work we plan to *refine* *m*-TPPs to give “efficient” manufacturing processes, e.g., 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

For more details, please see poster ??