Synthesis of Orchestrations of Transducers for Manufacturing

SAPIENZA
UNIVERSITÀ DI ROMA

Dmytro Horpynchenko
1807584

Based on the paper:

Synthesis of Orchestrations of Transducers for Manufacturing

Giuseppe De Giacomo, Moshe Y. Vardi, Paolo Felli, Natasha Alechina, Brian Logan

2018

A bit of background

1. Manufacturing as a service
2. Adopting industry 4.0 is essential to maintain competitiveness.
3. Process planning item Manual $\rightarrow$ Automated

Previous work on the automated synthesis of process planning

De Silva et al. 2016 and Felli et al. 2017

- Techniques based on AI behaviour composition
- In : a process recipe and a production topology
- Out : a process plan controller

These problems are :

- polynomial in the size of the topology, (exponential in the number of resources and polynomial in their size), and exponential in the size of the process recipe and number of resources
- Difficult to relate synthesis of controllers for manufacturing to the existing literature and tools on reactive synthesis

Uni-Transducer

What is a transducer? A uni-transducer?

A transducer is a finite deterministic automaton with outputs, we consider here Mealy machines. Uni-transducer is a machine that takes a single input and produces a single output in each state. Model manufacturing resources and process.

Definition 1

A (uni-)transducer $T = (\Sigma, \Delta, S, s_0, f, g)$ is a deterministic transition system with inputs and outputs, where Σ is the input alphabet, Δ is the output alphabet, S is the set of states , s_0 the initial state, $f : S \times \Sigma \to S$ is the transition function (which takes a state and an input symbol and returns the successor state) and $g : S \times \Sigma \to \Delta$ is the output function (which returns the output of the transition).

Uni-transducer example

$T = (\Sigma, \Delta, S, s_0, f, g)$

$\Sigma = \{cleansq, cleanrd, greensq, yellowrd\}$

Input alphabet encodes operations

$\Delta = \{sqclean, rdclean, sqgreen, rdyellow, err\}$

Output alphabet encodes results of operations

$S = s_0, s_1, s_2, s_e$

$f(s_0, cleansq) = s_1 \ g(s_0, cleansq) = sqclean$

Uni-transducer example

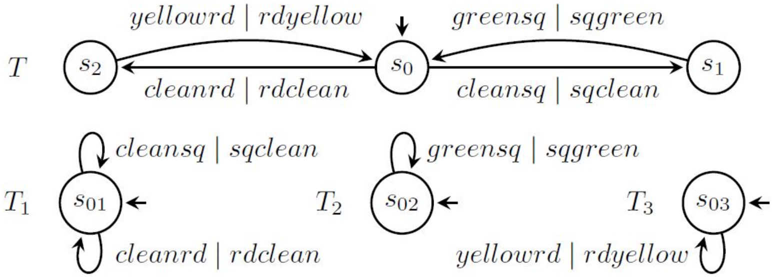

Figure: Process recipe T and manufacturing resources T_1, T_2 and T_3

Uni-Transducer: Orchestration

Definition

Given a set of transducers $T_1, \ldots, T_m$ and a production recipe transducer T, the orchestration problem is the question whether there is a controller C for $P = T_1 \times \cdots \times T_m$ which realizes T.

- Output sequence of T on input w: $\tau^o_{w,T} = T^o(a^0), \ldots, T^o(a^0 \ldots a^i) \ldots$
- $C : \Sigma^+ \longrightarrow \{1, \ldots, \mathrm{m}\}$
- The output of C on P over the input w is $\tau^o_{w,C} = b^0, \ldots, b^i, \ldots,$ where $b^i = g_h(s^i_h, a^i)$
- C realizes T if $\tau^o_{w,T} = \tau^o_{w,T}$ for all w.

Task of the controller

Decide, at each cycle, to which resource the input should be assigned.

Uni-Transducer: Orchestrator Synthesis

- DFA Games
- $\mathrm{G} = (\mathcal{X} \times \mathcal{Y}, \mathrm{Q}, \mathrm{q}_0, \delta, \mathrm{F})$
- Win by playing forever
- $\mathrm{PreC}(\mathcal{E}) = \{q \in \mathrm{Q} \mid$ for all $\mathcal{X} \in \mathcal{Y}$ exists $\mathrm{Y} \in \mathcal{Y}$ such that $\delta(q, (\mathrm{X}, \mathrm{Y})) \in \mathcal{E}\ \}$
- Define $\mathrm{Win}(\mathrm{G})$ by greatest-fixpoint
- $\mathcal{T}_G = (\mathrm{X} \times \mathrm{Y}, \mathrm{Q}, \mathrm{q}_0, \varrho, \gamma)$

Theorem

Checking whether there exists a controller C for P that realizes T can be done by solving the safety game G defined above.

Multi-transducers:Problem overview

Uni-transducers are suitable only for single pipeline processes. Meanwhile it is essential to:

- Process entities in parallel
- Split raw materials apart
- Combine or assemble multiple items to produce new entity type

Definition

Deterministic transition system with multiple input/output ports is called multi-port transducer or *multi-transducer*

Multi-transducers: Ports

Unlike uni-transducers, all interaction between controller and transducers are done through out ports and depends on the nature of things sent within the port they are separated to:

- *Physical* - accept/output physical objects such as parts. Physical output port can be bound only to one (physical) input port
- *Virtual* - accept/output signals such as messages specifying that a particular operation should be performed. Virtual output port can be bound to multiple (virtual) input ports

Any type of input ports should not be bound to more than one output port.

Note

All input ports must have input value and all output ports must provide an output value. In case of absence there are special empty value ϵ.

Multi-transducers: Ports

$in_{x,y}$ - the input port y of multi-transducer T_x
$out_{x,y}$ - the output port y of multi-transducer T_x
$val(in_{x,y}), val(out_{x,y})$ - the value at the input/output port y of transducer T_x
$(out_{x',y'}, in_{x,y})$ - *port bindings*, a connection between the output port y' of multi-transducer x' and input port y of multi-transducer x

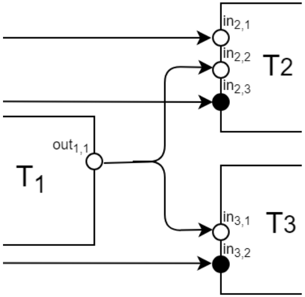

Figure: Connection example: white/black ports are virtual/physical ports respectively.

Multi-transducers: Environment and routing

Interaction with set of manufacturing resources (represented as transducers) is done using *binding sets* and through an object so-called *"Environment"*. Binding set is a set of port bindings that must be *"legal"*: it must be consistent with requirements described above.

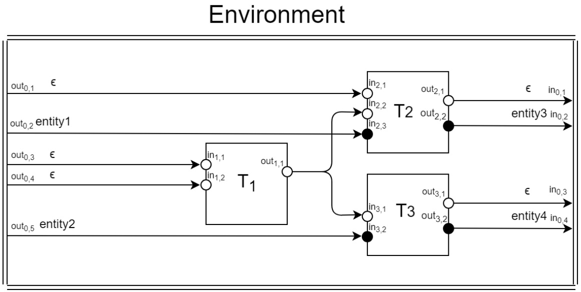

Figure: Example of multi-transducers binding set. Inputs and outputs that have no value use spesial ϵ value

Multi-transducers

Considering defined above, multi-transducer is defined as follows:

$$T = (\Sigma, S, s_0, f, g, k, l)$$

,where
Σ is the alphabet (of both inputs and outputs)
S is the set of states
s_0 is the initial state
$f : S \times \Sigma^k \to S$ is the transition function
$g : S \times \Sigma^k \to \Sigma^l$ is the output function
k is the number of T's input ports
l is the number of T's output ports

Orchestration

The manufacturing resources are represented as a set of multi-transducers $T_1, ..., T_m$

$$T_j = (\Sigma, S_j, s_{0j}, f_j, g_j, k_j, l_j)$$

The behavior of T on input $w = a^0 a^1 ...$, where $a_i \in \Sigma_k$, is described by the following sequence of states and outputs:
$T^s(a^0) = f(s_0, a^0)$
$T^o(a^0) = g(s_0, a^0)$

. . .
$T^s(a^0 ... a^i) = f(T^s(a^0 ... a^{i-1}), a^i)$
$T^o(a^0 ... a^i) = g(T^s(a^0 ... a^{i-1}), a^i)$
The observable output sequence of T on input w is

$$\tau^o(w, T) = T^o(a^0) ... T^o(a^0 ... a^i) ...$$

Orchestration

Considering a production P:

$$P = T_1 \times ... \times T_m$$

a controller C for P is defined as:

$$C : (\Sigma_k)^+ \rightarrow Cntl$$

,where *Cntl* is denoting the set of all possible port binding sets.
Identically to uni-transducers case,

Theorem

C realizes T if $\tau^o(w, T) = \tau^o(w, C)$ *for all w*

Orchestrator Synthesis

Synthesizing a controller for multi-transducer setting has the same principle as in uni-transducers case: by adopting automata-theoretic techniques and resort to solving a safety game.
Given the target transducer T and the available transducers $\{T_1, ..., T_m\}$, we build the safety game:

$$G_{multi} = (\Sigma^k, Cntl, Q, q_0, \delta)$$

where
Σ^k is the input alphabet of the target T;
$Q = S \times S1 \times ... \times S_m \cup \{q_{err}\}$ is the cartesian product of the states;
$q0 = (s_{0,} s_{01}, ..., s_{0m})$
δ is the states transition function (see next slide for definition)

Example: Implementation of orchestration of uni-transducer setting

As the practical part of paper analysis orchestration of uni-transducers setting was implemented in Python.

Description of the target transducer and manufacturing resources must be provided as text files using special notation:
state input newstate->output

Read more

Full description,python environment setup instruction, this slides and domains examples can be found on GitHub repository
`https://github.com/dhorpynchenko/Orchestration-of-Transducers-for-Manufacturing`