Synthesis of Orchestrations of Transducers for Manufacturing

Dmytro Horpynchenko
1807584

SAPIENZA
UNIVERSITÀ DI ROMA

Multi-Port Transducer Setting

Based on the paper:

Synthesis of Orchestrations of Transducers for Manufacturing

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

2018

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

Multi-transducers

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

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

,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

Multi-transducers: Ports

All interaction between environment and transducers are done through out ports.
Depending on 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.

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

Multi-transducers: Ports

A set c of port bindings, henceforth called *binding set*, must be consistent with a set of binding constraints $\mathcal{B}$, specified as boolean combinations of atoms of the form $(out_{x',y'}, in_{x,y})$; a binding set c is said to be *legal* iff $c \models \mathcal{B}$.
We use the set $\mathcal{B}$ to impose three kinds of requirements:

- for all x, y (with $x \in \{0, 1, ..., m\}$ and $y \in \{1, ..., k_x\}$) there exists at most one x', y' such that $(out_{x',y'}, in_{x,y}) \in c$ (if, for some $z \in \{1, ..., k_x\}$,$in_{x,z}$ does not appear in c, its value is assumed to be empty, i.e., $val(in_{x,z}) = \epsilon$);
- all physical output ports $out_{x',y'}$ occur in at most one binding $(out_{x',y'}, in_{x,y}) \in c$
- arbitrary requirements specifying, e.g., the possible physical connections between machines on the shop floor, or the set of virtual connections determined by the possible communication routes between resources.

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.

Note

Opposite to uni-transducers case, the controller binds ports, and all transducers $T_1, ..., T_m$ get input (possibly empty) and move at every step.

Orchestration

The sequence of (global) states and outputs generated by the controller on $w = a^0 a^1 ...$ is, respectively,

$$\tau_{w,C} = (s_{01}..s_{0m})...(s_{i1}...s_{im})...$$

$$\tau^o_{w,C} = b^0...b^i...$$

where
$C(a^0...a^i) = c^i$ and c^i is legal
$val^i(in_{x,y}) = val^i(out_{x',y'})$ for $(out_{x',y'}, in_{x,y}) \in c^i$
$s_x^{i+1} = f_x(s_x^i, val^i(in_x))$ for $x \in \{1, ..., m\}$
$val^i(out_x) = g_x(s_x^i, val^i(in_x))$ for $x \in \{1, ..., m\}$
$val^i(out_0) = a^i$ (Environment is represented as transducer T_0)
$val^i(out_{x,y}) = b^i$ if $(out_{x,y} in_{0,y'}) \in c^i$

Orchestration

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)

Orchestrator Synthesis: Transition function

States transition function δ is defined as follows:
$\delta((s, s_1, ..., s_m), a, c) = (s', s'_1, ..., s'_m)$ if the following conditions hold:

- $s_0 = f(s, a)$
- c is legal;
- $val(in_{x,y}) = val(out_{x',y'})$for $(out_{x',y'}, in_{x,y}) \in c$;
- $s'_x = f_x(s_x, val(in_x))$ for $x \in \{1, ..., m\}$;
- $val(out_x) = g_x(s_x, val(in_x))$ for $x \in \{1, ..., m\}$;
- $val(out_0) = a$;
- $val(out_{x,y}) = g(s, val(out_0))_{y'}$ if $(out_{x,y}, in_{0,y'}) \in c$, (the value $b_{y'}$ of the output port y' of the target is the same of the value of the output port y of the available transducer x, when c binds y to y'.

$\delta((s, s_1, ..., s_m), a, c) = q_{err}$, otherwise.

Multi-transducers: Example

The recipe is as follows: take a square and a round part, clean them both, paint the square part green and round part yellow, and glue them together. On completion of the recipe, a final transition simply requests the resulting product. Recipe transducer:

$$T = (\Sigma, S, s_0 f, g, 4, 3)$$

where
Σ= {fsq, rd, sqrd, clean, green, yel, glue, glued, err, sqclean, rdclean, sqgreen, rdyellowg},
$S = \{s_0, s_1, s_2, s_3, s_e\}$

Multi-transducers: Example

Transition and output definitions:

$f(s_0, (sq, rd, clean, clean)) = s_1$;
$g(s_0, (sq, rd, clean, clean)) = (sqclean, rdclean, \epsilon)$;

$f(s_1, (\epsilon, \epsilon, green, yel)) = s_2$;
$g(s_1, (\epsilon, \epsilon, green, yel)) = (sqgreen, rdyellow, \epsilon)$;

$f(s_2, (\epsilon, \epsilon, glue, \epsilon)) = s_3$;
$g(s_2, (\epsilon, \epsilon, glue, \epsilon)) = (glued, \epsilon, \epsilon)$;

$f(s_3, (\epsilon, \epsilon, \epsilon, \epsilon)) = s_0$;
$g(s_3, (\epsilon, \epsilon, \epsilon, \epsilon)) = (\epsilon, \epsilon, sqrd)$

Any other input results in a transition to s_e with output (err, err, err)

Multi-transducers: Example

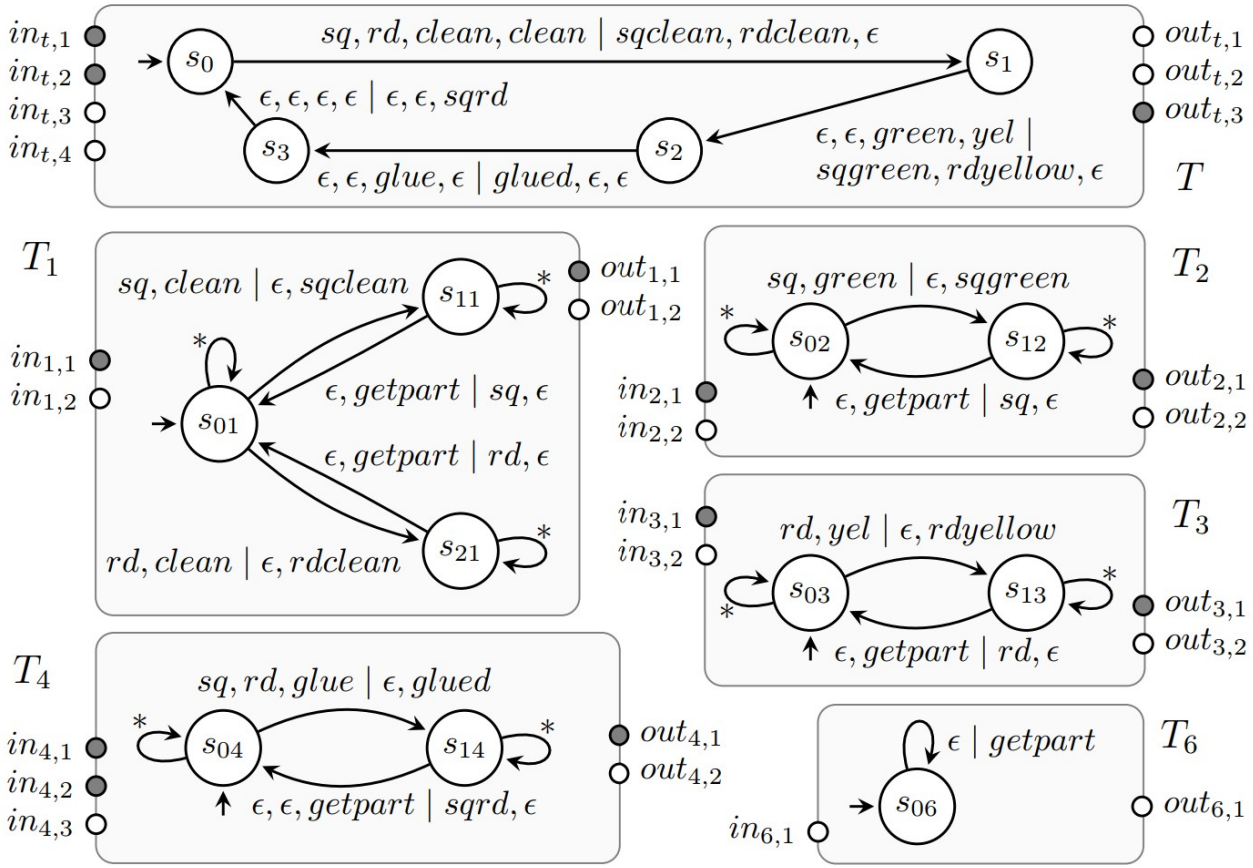

Figure: Target transducer T and resource transducers $T_1, ..., T_6$ (T_5 is identical to T_1). Physical ports are shown greyed, and error states s_{ej}, are not shown. $(*)$ on self-loops stands for $\epsilon, ..., \epsilon, ... | \epsilon, \epsilon$