# Unbounded Orchestrations of Transducers for Manufacturing

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

Presented by E.Puglisi,M.Persiani,F.Frattolillo
Sapienza University

May 17,2019

# Overview


1. Introduction
2. Uni-Transducer
3. Orchestration Problem
4. Multi-Dimensional energy game
5. Orchestrator Synthesis
6. Example
7. Multi-Transducer
8. Orchestration Problem (Multi-Transducer)
9. Orchestration Synthesis (Multi-Transducer)

# Introduction

- Let's give some definitions:
  - **process plan:** specifies the specific *manufacturing resources* to be used for each manufacturing and assembly operations, and how materials and parts move between the various manufacturing resources
  - **manufacturing resources:** computer, robots etc.
  - **process recipe:** set of abstract manufacturing tasks.
- Reasons behind this study:
  - Process planning is traditionally carried out by manufacturing engineers and is largely a manual process.
  - Such process is uneconomic for the small batch sizes typical of the manufacturing as a service model.
  - There is an increasing interest in using synthesis to automate the synthesis of manufacturing process planning

# Introduction

- Sometimes the products to be manufactured are not known in advance and are often produced in small batches to tight timescales.
- We consider the problem of whether a product can be manufactured given sufficient resources.
- Only the types of available manufacturing resources are given, and we want to know whether is possible to manufature a product using only resources of those types, and, if so, how many resources of each type are needed.

# Uni-Transducers

A **Transducer** is a finite deterministic automaton with outputs.
A Uni-transducer take a single input and produce a single output in each state.
We consider tranducer which are *Mealy machine*.

## Uni-Transducer

A (uni-)transducer $T = (\Sigma, \Delta, S, s_0, f, g)$ is a deterministic transition system with inputs and outputs, where $\Sigma$ is the *input alphabet*, $\Delta$ is the *output alphabet*, S is the *set of states*, $s_0$ is the *initial state*, $f : Sx\Sigma \mapsto S$ is the *transition function* and $g : Sx\Sigma \mapsto \Delta$ is the *output function*.

We will consider both manufacturing resources and process recipes as uni-transducers.

# Orchestration Problem

- We have a set of *available transducer types* $\{T_0, ..., T_m\}$
- Where $T_j = (\Sigma, \Delta, Sj, s_{0j}, f_j, g_j)$ represents a manufacturing resource and all $T_j$'s are over the same input and output alphabet.
- T represents a *process recipe* which is a transducer with the same input and output as the $T_j$'s.

## Unbounded Orchestration

The unbounded orchestration problem is to combine some numbers $n_j$ of resources of each type $T_j$ to be able to mach the *behavior* of T.

# Orchestration Problem

## Behaviour of T

The behaviour of T on input $\omega = a^0, a^1, ...$ 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 $\omega$ is:

$\tau^o_{\omega,T} = T^o(a^0), ..., T^o(a^0, ..., a^i)...$

# Orchestration Problem

## Controller

Consider the transducer corresponding to the whole production facility:
$P = T_{1,1}x...xT_{1,n_1}x...xT_{m,1}x...xT_{m,n_m}$
where $n_j$ is the number of copies of every transducer type $T_j$.
A *controller* for P is a function
$C : \Sigma^+ \mapsto \{(1,1), ..., (m, n_m)\}$
That, for each finite input string, picks a transducer in P to make a transition. The sequence of states generated by the controller on inputs $\omega$ is:
$\tau^s_{\omega,T} = (S^0_{1,1}, ..., S^0_{m,n_j}), ..., (S^i_{1,1}, ..., S^i_{m,n_j}), ...$
where only one of the local state changes in each transition:
$s^{i+1}_h = f_h(s^i_h, a^i) \quad if \quad C(a^0, ..., a^i) = h$
$s^{i+1}_h = s^i_h \quad if \quad C(a^0, ..., a^i) \neq h$

**orchestration problem**: check if there are numbers of copies of each transducer type such that there is a controller C for P which realizes T.

# Multi-Dimensional energy game

The unbounded orchestration problem can be reduced to a decidable problem on *multi-dimensional energy games*.

## multi-dimensional energy game

A multi-dimensional energy game is a tuple $G = (Q, R)$
**Q:** is a finite set of states
**R:** is a finite set of transitions of the form $(q_1, op, q_2)$
**C:** finite set of counters.
**op:** $\{0, -1, 1\}^n$ is an update vector for the value of the counters.
**counter evalutation:** $\nu : C \mapsto Z$ maps every counter to a set of integers.
**configuration:** $\gamma$ is a pair (q,$\nu$), where $q \in Q$
**transition:** between two configuration $\gamma_1 \mapsto^t \gamma_2$ $(t \in R)$is valid if $\nu_2 = \nu_1 + op$.
**history:** is a finite sequence of configurations
**strategy:** for a Player $i \in \{0, 1\}$ is a function $\sigma_i$ that assigns to each history ending in a configuration $\gamma$ a possible transition.

# Multi-Dimensional energy game

## Winning strategy

Player 0 **wins the game** in $\gamma$ if it wins every play starting on $\gamma$.
**Winning game:** a configuration for a player is winning if $v_k(c) \geq 0$ for all $k \geq 1$ and $c \in C$.
**Winning set:** the set $W(G, 0)$ is the set of configurations $\gamma$ in which the player 0 wins.
**Pareto frontier:** is $min(W(G, 0))$.
The Pareto frontier is computable in **doubly exponential time** and **pseudo-polynomial** assuming the number of counters is fixed. The maximum norm of the Pareto frontier are bounded by an exponential value in the number of counter $n$ and polynomial in the number of states and transitions.

# Pareto Frontier

## Pareto Efficiency

State of allocation of resources in which it's impossible to re-allocate so to improve any individual evaluation criterion without worsening at least another one. (i.e. an optimal configuration)

## Pareto Frontier

The set of all the *pareto efficient* allocations for a given set of resources.

In our case the resources to allocate are the transducer type counters stored in vector $\nu$.

# Orchestrator Synthesis

Let's see how to reduce the **orchestration problem to a multi-dimensional energy game**.

We are given $m$ resource transducer types:
$T_i = (\Sigma, \Delta, P_i, p_0^i, \alpha_i, \beta_i)$
and a target process recipe:
$T = (\Sigma, \Delta, S, s_0, f, g)$
The realizability problem for $\prod T_i^{n_i} = T_{1,1}x...xT_{1,n_1}x...xT_{m,1}x...xT_{m,n_m}$ can be formulated as a game between Controller(player 0) and Adversary(player 1). The game starts with adversary choosing a symbol from the input alphabet $\Sigma$ and the controller must choose a component $T_{i,j}$ where $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$.
Controller must be able to keep playing forever without causing a mismatch of outputs between T and $\prod T_i^{n_i}$.

# Orchestrator Synthesis

We can describe $\prod T_i^{n_i}$ as an integer vector
$v = (v_0^1, ..., v_{k_1-1}^1, ..., v_0^m, ..., v_{k_5-1}^m)$
where $k_i$ is the cardinality of the set of states of $T_i$ and $v_0^i + ... + v_{k_i-1}^i = n_i$
Where $v_q^i$ represents the number of copies of $T_i$ in the state $p_q^i$
The game starts with all the copies of all the transducers in the state $s_0$, hence for each transducer $T_i$ the corresponding vector $v_i$ is $(n_i, 0)_i$

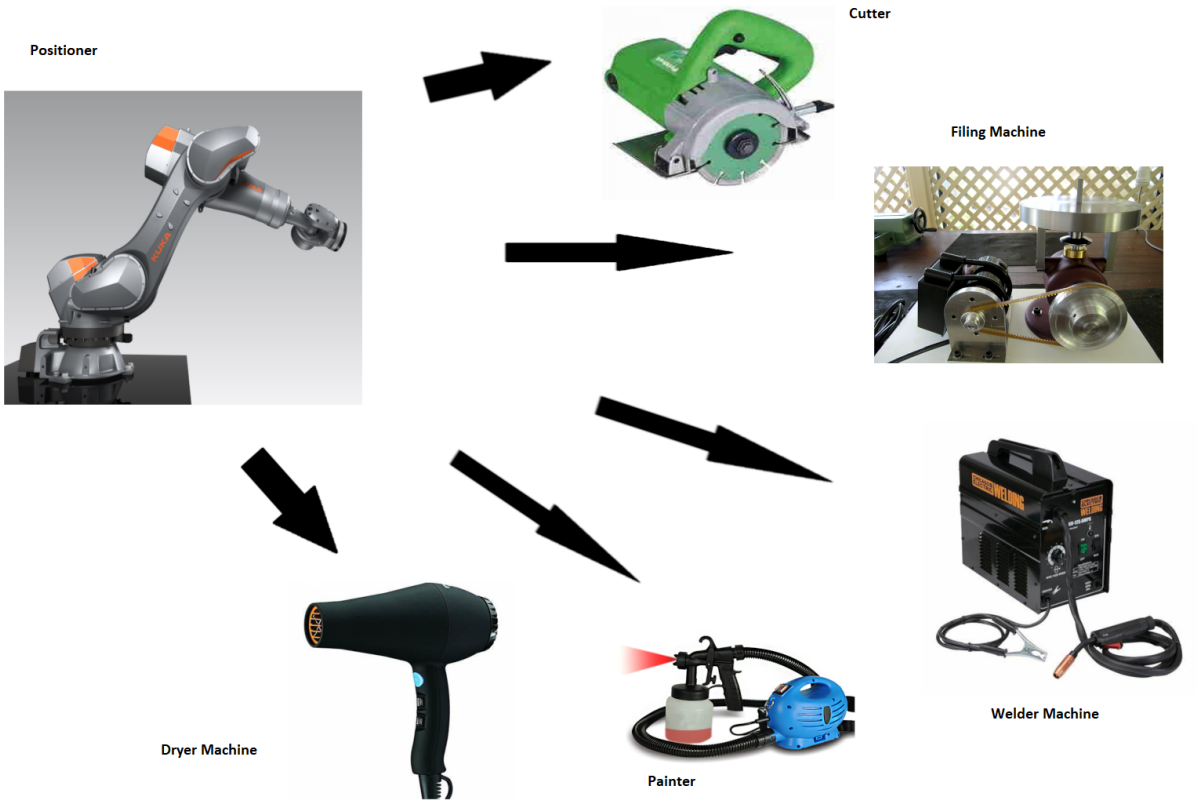
The vector v representing the counter evaluation is updated by adding the update vector $u = (u_0, ..., u_{k_m-1}^m)$ where $u_q^i = -1$ and $u_q^i = 1$ and all other components are 0. This operation represents the transition between two states of one of the available resource transducers.

# Orchestrator Synthesis

Now we can define the following multi-dimensional energy game $G_{T_1,\ldots,T_m,T}$:

- The state set of $G_{T_1,\ldots,T_m,T}$ is $(SxΣ) \cup S$ where $(SxΣ)$ represent all the combination of input symbol and states of T (including the error states). $Q_0 = (SxΣ)$ are the states of player 0 and $Q_1 = S$ are the states of player 1.
- For each transition $(s, a, b, t)$(initial state, input symbol, output, new state) of T, we have in $G_{T_1,\ldots,T_m,T}$ the transition $((s, a), u, t)$ for all $u \in U_{a,b}$ (player 0's moves) and a transition $(s, 0, (s, a))$ (player 1's moves).

# Example

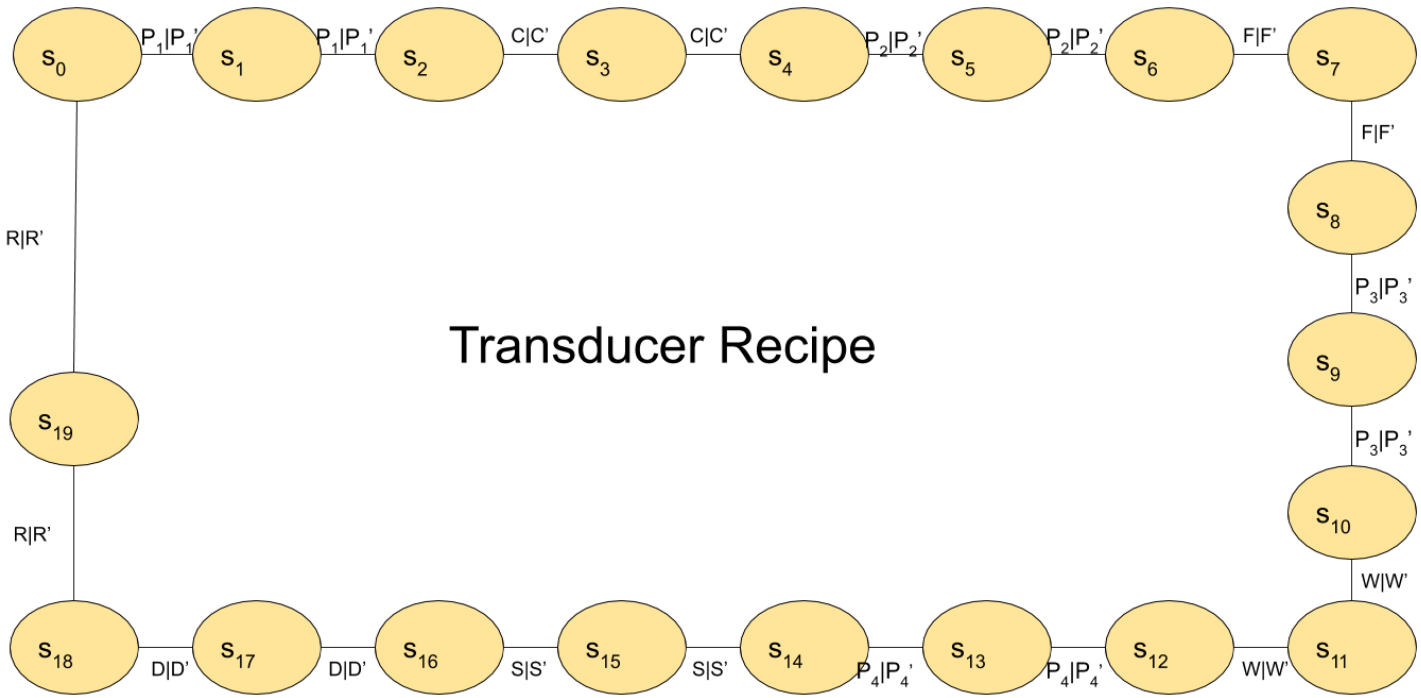

Positioner
Cutter
Filing Machine
Welder Machine
Dryer Machine
Painter

# Example

Given two metal objects, the manufacturing recipe require to:

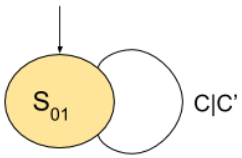
- Position both the objects on the **cutting machine** and cut them.
- Position the objects on the **Filing machine** and file them.
- Position the objects on the **Welder machine** and weld them.
- Position the objects on the **Paint machine** and both paint and dry it.
- Release the object.

# Example

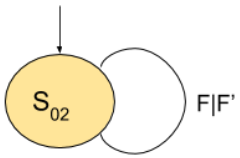

Transducer Recipe
S0
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
P1|P1'
P1|P1'
C|C'
C|C'
P2|P2'
P2|P2'
F|F'
F|F'
P3|P3'
P3|P3'
W|W'
W|W'
P4|P4'
P4|P4'
S|S'
S|S'
D|D'
D|D'
R|R'
R|R'

# Example

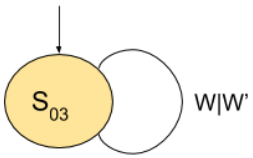
T1:Cutting Machine



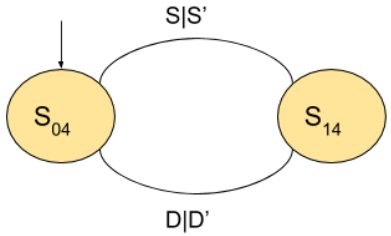

T2:Filing Machine



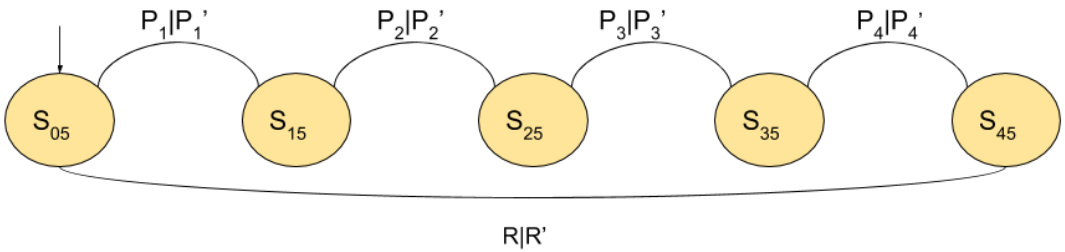

T2:Welding Machine




T2:Painting Machine




T1:Positioning Machine



# Example

Our process recipe is defined as a quintuple $T = (\Sigma, \Delta, S, s_0, f, g)$ where:

- $\Sigma = \{P_1, P_2, P_3, P_4, C, F, S, D, R, W\}$
- $\Delta = \{P_1', P_2', P_3', P_4', C', F', S', D', R', W'\}$
- $S = \{s_0, s_1, ..., s_{19}, s_{err}\}$
- $f = \{f(s_0, P_1) = s_1, ..., f(s_{19}, R) = s_0\}$
- $g = \{g(s_0, P_1) = P_1', ..., g(s_{19}, R) = R'\}$

Let's define the transducer corresponding to the whole production facility as:
$P = \prod_i^{n_i} T_i = T_{1,1} x ... x T_{1,n_1} x ... x T_{5,1} x ... x T_{5,n_5}$
where $n_j$ is the number of copies of the transducer $T_j$.

# Example

Our controller will be a function $C : \Sigma^+ \mapsto \{(1, n_1), ..., (5, n_5)\}$
Now let's reformulate our problem as a multi-dimensional energy game.
We are given:

- 5 resource transducers **types** $T_i = (\Sigma, \Delta, P_i, p_0^i, \alpha_i, \beta_i)$
- The process recipe as defined above.
- P = the transducer corresponding to the whole production facility.

This game is played by two player:

- **Player 0:** is the Controller.
- **Player 1:** is the Adversary.

# Example

We can describe $\prod_i^{n_i} T$ as an integer vector
$v = (v_0^1, ..., v_{k_1-1}^1, ..., v_0^5, ..., v_{k_5-1}^5)$
where $k_i$ is the cardinality of the set of states of $T_i$ and $v_0^i + ... + v_{k_i-1}^i = n_i$
Where $v_q^i$ represents the number of copies of $T_i$ in the state $p_q^i$
The game starts with all the copies of all the transducers in the state $s_0$, hence for each transducer $T_i$ the corresponding vector $v_i$ is $(n_i, 0)_i$

# Example

- The game starts with adversary choosing an input symbol $a \in \Sigma$ yielding output g(p,a) and changing the state of T to f(s,a).
- The controller must choose a copy of $T_i$ in some state $p_q^i$ and replacing it by a copy in state $p_r^i = \alpha_i(p_q^i, a)$. The output corresponding to this transition is $b = \beta_i(p_q^i, a)$.
- The previous transition is denoted by $(p_q^i, a, b, p_r^i)$
- The vector v representing the counter evaluation is updated by adding the *update vector* $u = (u_0, ..., u_{k_m-1}^m)$ where $u_q^i = -1$ and $u_q^i = 1$ and all other components are 0. $U_{a,b}$ is the set of such update vectors for transition with input $a \in \Sigma$ and output $b \in \Delta$.
- Controller must be able to play forever without causing a mismatch of outputs between T and $\prod_i^{n_i} T$ . If controller has a **winning strategy** then $\prod_i^{n_i} T$ realizes T.

# Multi-Transducer

Multi-transducer are used for modeling manufacturing resources that can take multiple input and output ports.

- A multi-transducer is a tuple $T = (\Sigma, S, s_0, f, g, k, l)$ where k is the number of input ports and l is the number of outputs port
  - $f : Sx\Sigma^k \mapsto S$ transition function that takes n inputs and a state and return the successor state.
  - $g : Sx\Sigma^k \mapsto \Sigma^l$ is the output function that returns the output associated to a transition

Ports can be *physical* (which accept physical object) or *virtual* (which accept signals). A physical output port can be bounded only to one physical port, while a virtual output port can be bounded to multiple virtual output port. In general an input port should not be bounded to more than one output ports.

# Orchestration Problem for multi-transducers

- We have a set of multi-transducer types $T_1, ..., T_m$ where each $T_j = (\Sigma, S_j, s_{0,j}, f_j, g_j, k_j, l_j)$ representing manufacturing resources. We also have a recipe T that is a multi-transducer with the same alphabet as $T_j s$.
- The behaviour of T on input $\omega = a^0, a^1, ...$ (where $a^i \in \Sigma^k$) has the same definition as the uni-trasducer case.
- Define P as the multi-transducer corresponding to the whole production facility which is a composition of $n_j$ copies of $T_j$.
- In addition to picking a transducer $x \in \{(1,1), ..., (m, n_m)\}$, the controller is also in charge of changing the port binding .

# Orchestration Problem for multi-transducers

- We denote $in_{x,y}$ the input port y of multi-transducer $T_x$, and by $out_{x,y}$ the output port y of $T_x$. We extend the notation by using index x=0 to represent inputs and outputs of the environment.
- **Port binding c:** is a pair $(out_{x',y'}, in_{x,y})$ which represent a connection between output port y' of multi-transducer x' and the input port y of multi-transducer x.
- **Binding constraint B:** is a boolean combinations of atoms of the form $(out_{x',y'}, in_{x,y})$, B is used to impose three kinds of requirements:
  - for all x,y there exists at most one x',y' such that $(out_{x',y'}, in_{x,y}) \in c$(if for some outputs z, $in_{x,z}$ does not appear in c, its value is assumed empty: $val(in_{x,z}) = \epsilon$
  - all physical output ports $out_{x',y'}$ occur in at most one binding $(out_{x',y'}, in_{x,y}) \in c$
  - Others requirements specifying physical and virtual connection.

# Orchestration Problem for multi-transducers

## Controller

A **controller C** for P is a strategy $C : (\Sigma^k)^+ \mapsto Cntl$ (where Cntl is the set of all possible port binding set).

The sequence of (global) states and outputs generated by the controller on $\omega = a^0 ... a^i ...$ is respectively:
$\tau_{\omega,C} = (s^0_{1,1}, ..., s^0_{m,n_m}), ..., (s^i_{1,1}, ..., s^i_{m,n_m}) ...$
$\tau^o_{\omega,C} = b^0, ..., b^i, ...$
where:
$C(a^0, ..., a^i) = c^i$ and $c^i$ is legal, where legal means that $c \models B$.
$val^i(in_{x,y}) = val^i(out_{x',y'}) \in c^i$.
$s^{i+1}_x = f_x(s^{i+1}_x, val^i(in_x))$.
$val^i(out_x) = g_x(s^i_x, val_i(in_x))$.

# Orchestration Problem for multi-transducers

C realizes T if $\tau^o(\omega, T) = \tau^o(\omega, C)$ for all $\omega$
While in the uni-transducer case, the controller select one transducer to make a move, in the multi-transducer case, the controller binds ports, and all transducers $T_{1,1}, ..., T_{m,n_m}$ get input(possibly empty) and move at every step.

# Orchestrator Synthesis for multi-transducers

## Unbounded Multi-Transducer

In the unbounded multi-transducer case, this problem is undecidable. The main reason for undecidability is the combination of multiple input and output ports and no bound on the number of transducers.

# Undecidability example with Turing Machine

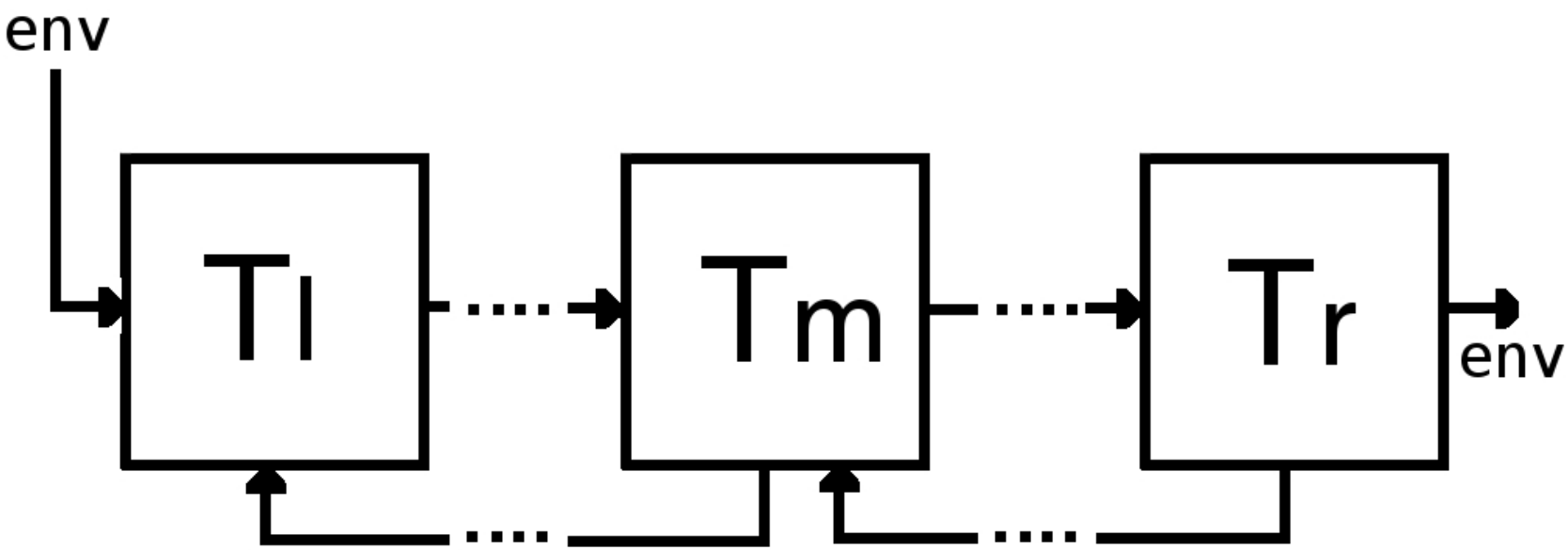
**Proof:**

Let's prove that the problem is undecidable by exibiting an example where such a controller exists if and only if a Turing machine M halts on empty input.

- Our target transducer is T which on input a outputs a. $T = (\Sigma, S, s_0, f, g, 1, 1)$ where $\Sigma = \{a\}$, $S = s_0$, $f(s_0, a) = s_0$, $g(s_0, a) = a$.
- We consider M as a single tape of TM with instruction of the form $qc \mapsto c'dq'$. Where $d \in \{-1, 1\}$ which means , if in state q reading symbol c, write symbol c' and goes to left (d=-1) or to right(d=1).
- Between every two instructions M performs 2 actions:
  - It adds a cell to the right and write $\omega$ on the rightmost non blank cell.
  - it goes all the way left and then return on to the last state before the real instruction.

# Undecidability example with Turing Machine

We have three type of transducer resources $T_l, T_m, T_r$ representing leftmost, middle and rightmost cell. The alphabet of the resource transducer is $\Sigma' = q_0, ..., q_n, a, err$ where $q_0, ..., q_n$ are the states of M.



## left cell transducer:

It has two input ports, one receiving **a** from the environment and the other receiving the output of the cell on the right. It has one output to communicate with the cell to the right. $T_l$ starts in state $s_{init}$ (corresponding to containing $\alpha$ and being in state $q_0$) and on input a on input port 1 and blank on 2($(a, \epsilon)$), it mimics the instruction of M: writes $\alpha$ again, and outputs the symbol $q_i$ which is the state M goes into upon empty input. $T_l$ then goes in state $s_\alpha$. If $T_l$ receives input $q_j$ from the right ($(\epsilon, q_j)$), in $s_\alpha$, it again does what the instruction of M requires, and outputs the new state q'. On all other inputs it outputs *err* and goes in state $s_{err}$.
States of $S_l$ are $\{s_{init}, s_\alpha, s_{err}\}$

## middle cell transducer:

It has two outputs and two inputs connected to the cell on the left and the cell on the right. Input 1 and output 1 connect to the cell on the left, and input 2 and output 2 connect to the cell on the right. Input $(q, \epsilon)$ means receiving q from the left, and input $(\epsilon, q)$ means receiving q from the right. Similarly for outputs: the output $(q', \epsilon)$ corresponds to moving to the cell to the left in state q', and $(\epsilon, q')$ corresponds to moving to the cell to the right in state q'.

The states of $T_m$ are $s_\epsilon$ (being blank, initial state), $s_0$ (containing 0), $s_1$ (containing 1), $s_\omega, s_{h_1}, ..., s_{h_t}, s_{err}$.

On input q when in state $s_c$ the trasducer will go into state $s_{c'}$ and output q' on the left output port (if d=-1) or on the right (if d=1) output port.

On all other inputs, for example input from the right when in state $s_\epsilon$, it goes into $s_{err}$ and outputs *err*.

## right cell transducer:

It has one input port from the left cell and 2 output ports, one connected to the left cell and the other connected to the environment and can be used to output **a**. This only happens when M would have gone into state $q_n$ (so instead of outputting $q_n$, $T_r$ outputs **a**). Otherwhise $T_r$ mimics M's transitions: when it is in state $s_c$ and receives q on the left, it goes into state $s_{c'}$ and outputs q' to the left. In all other cases, in particular when getting as input a state symbol which corresponds to moving one cell to the right to write $\omega$ in it, $T_r$ goes into $s_{err}$ and outputs *err*.
The states of $T_r$ are $S_r = \{s_{err}, s_\omega, s_0, s_1, s_{h_1}, .., s_{h_t}, s_{err}\}$.

# Undecidability Proof

Let's prove that a controller for a correctly wired composition of length k realizing T exists if and only if M halts after using k cells.
Suppose we have a correctly wired composition of length k. It has a free input port ($T_I$) and all the controller can do is to pass a to that port. On input a, $T_I$ starts the imitation of a Turing machine M, passing symbols corresponding to the state of M back and forth along the composition, with cell transducers changing states to represent a new symbol on the tape of M. If M halts in the kth cell of the tape, eventually $T_r$ gets input $q_n$, upon which it outputs a, and T is realized. If M does not halt, then $T_r$ will get an input to the right to write $\omega$ in it, and $T_r$ outputs err.

*Thank you!*