# Modeling Hidden Events in Lower-level OR Holons

Sadegh Rahnamoon and W.M. Wonham

Systems Control Group, Electrical and Computer Engineering Department, University of Toronto

## 1 Introduction

This report aims to introduce a novel method of abstraction based on an arbitrary subset of events given as *events of interest* ($\Sigma_i \subseteq \Sigma$). In other words, given a holon $H = (X, \Sigma, \delta, X_0, X_m)$ (or a generator in general) and $\Sigma_i \subseteq \Sigma$, the present algorithm produces a high-level holon $H^{hi} = (X^{hi}, \Sigma_i, \delta^{hi}, X_0^{hi}, X_m^{hi})$ which only represents the occurrence of events inside $\Sigma_i$. $X^{hi}$ may have a number of superstates which hide other events ($\sigma \in \Sigma - \Sigma_i$) inside themselves.

Based on [1], a state tree structure is *well-formed* if it respects the **local coupling** condition.

**Definition 1.** [Local Coupling] Events of an inner transition structure can only be shared among those holons matched to the OR states that are AND-adjacent to the same AND superstate. Formally, for all superstates $a \neq b$ with matching holons $H^a, H^b \in \mathcal{H}$, we require

$$\Sigma_I^a \cap \Sigma_I^b \neq \emptyset \Rightarrow (\exists z) z <_\times x \quad \& \quad z <_\times b.$$

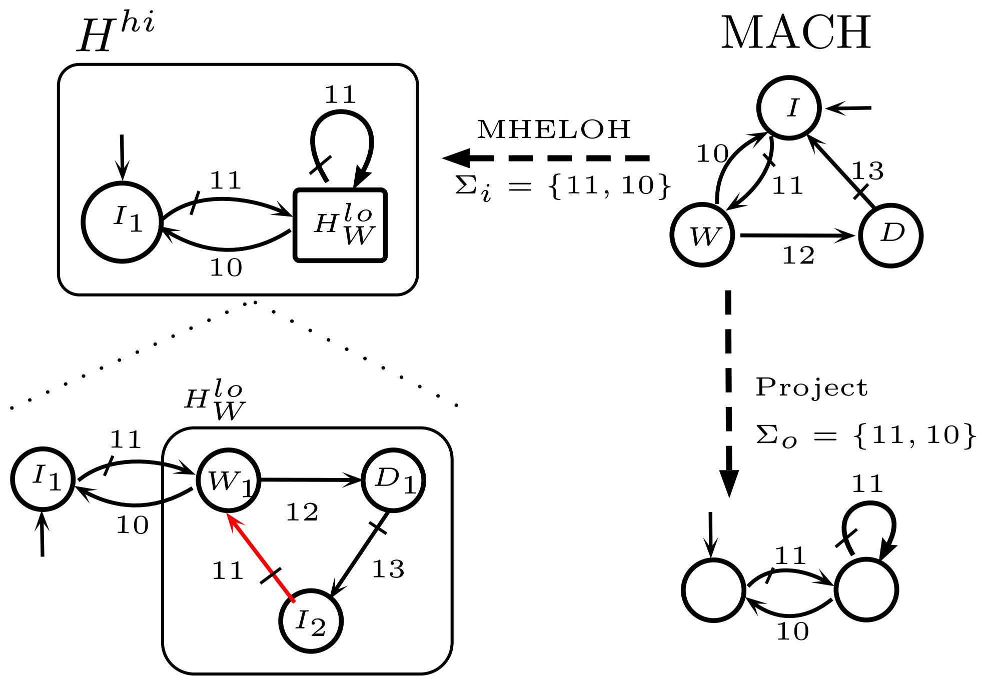
This author believes that this condition does not affect the fundamental definitions and functions of the state tree structures, so momentarily assumes that holons matched to the recently added superstates can also include $\sigma \in \Sigma_i$ as their internal events.

In this report a procedure called MHELOH is introduced. Given a subset of events as the events of interest ($\Sigma_i$), MHELOH produces another DES which hides events from the set $\Sigma - \Sigma_i$ inside its superstates. The behavior of the resulted DES without considering the details inside those superstates is isomorphic to the result of the familiar projection procedure while taking $\Sigma_o = \Sigma_i$ as the set of observable events.

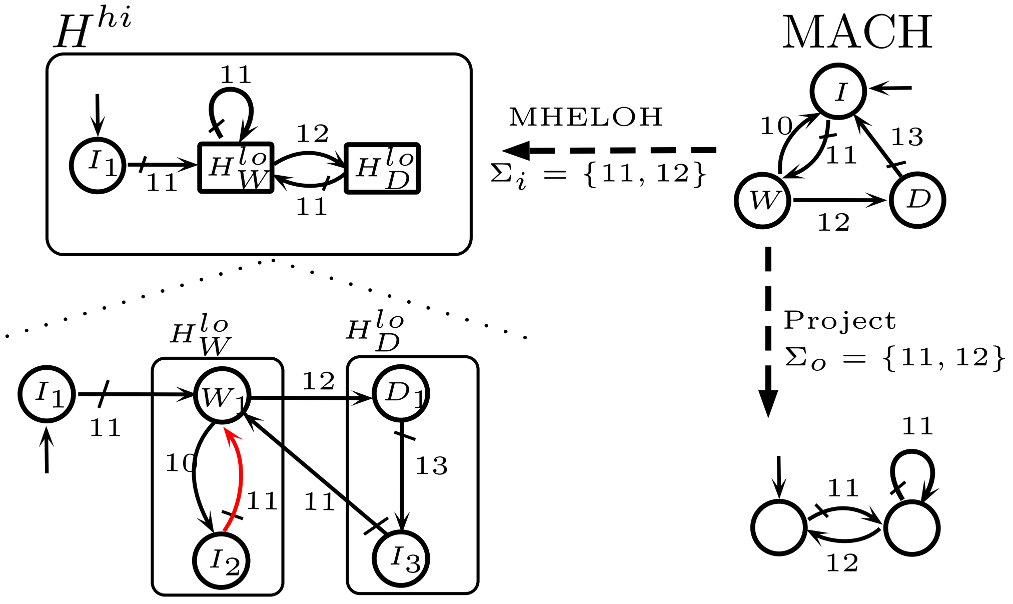
## 2 Examles

Here we take the MACH model with the event set $\Sigma = \{10, 11, 12, 13\}$, and will represent the results of abstraction given different subsets of events as $\Sigma_i$.

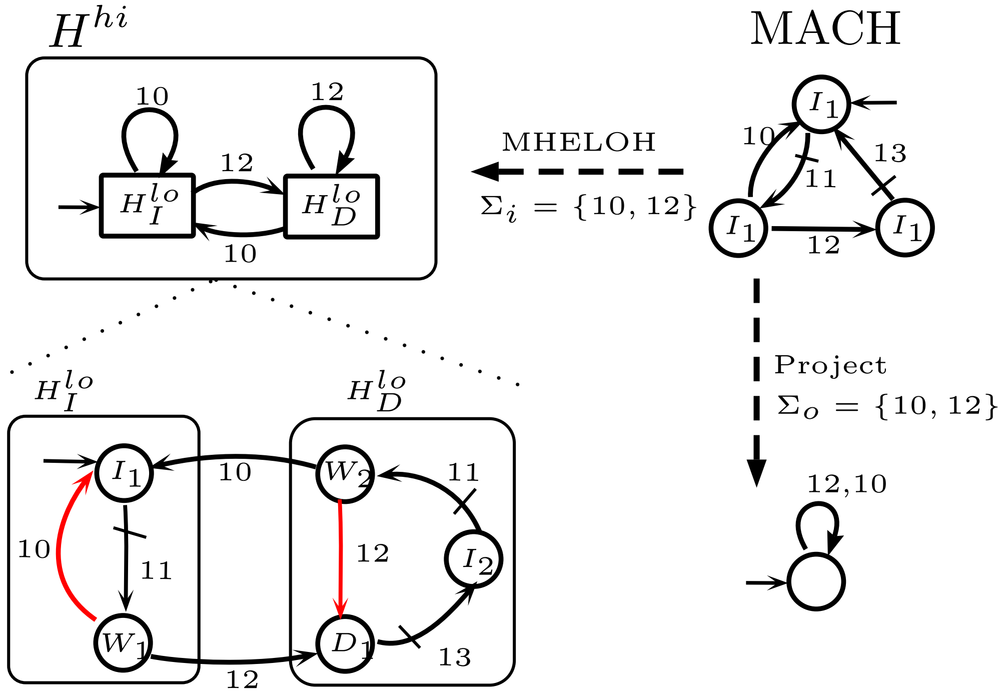
- $\Sigma_i = \{10, 11\}$



- $\Sigma_i = \{11, 12\}$


$H^{hi}$
MACH
MHELOH
$\Sigma_i = \{11, 12\}$
Project
$\Sigma_o = \{11, 12\}$
$H^{lo}_W$
$H^{lo}_D$
$I_1$
$W_1$
$D_1$
$I_2$
$I_3$
$I$
$W$
$D$
10
11
12
13

- $\Sigma_i = \{10, 12\}$


$H^{hi}$
MACH
MHELOH
$\Sigma_i = \{10, 12\}$
Project
$\Sigma_o = \{10, 12\}$
$H^{lo}_I$
$H^{lo}_D$
$I_1$
$W_1$
$W_2$
$D_1$
$I_2$
10
11
12
13
12,10

## 3 Algorithm

First, we endow each state of the original DES with three main attributes: color, reference state, and list of representatives inside the output DES.

```
state{
        color
        ref_state
        reps: list<state>
}
```

The **color** attribute is used to distinguish an already visited state. Each state is initially colored white, then it is changed to gray whenever that state is visited by the algorithm for the first time.

Each state inside the produced DES represents one and only one state of the original DES which is stored in the **ref_state** attribute. Respectively, each state of the original DES has one or several representatives in the produced DES which are kept inside the **reps** ( list of states) attribute. Obviously, reps[0] is the first representative of that state inside the result DES.

**Algorithm 1** Modeling hidden events in lower-level OR holons

```
1: procedure MHELOH($G, \Sigma_i$)
2:     $H \leftarrow$ new superstate
3:     for all $x \in X$ do          $\rightarrow$ $X$ is the state set of $G$.
4:         if $x$.color == white then
5:             Procedure1($H$, _, $x$, _)
6:         end if
7:     end for
8:     return $H$          $\rightarrow$ This is the result.
9: end procedure
```

Procedure1 has 4 input arguments. The first argument $H$ is the high-level holon of the output DES. The second argument $s$ is the most recently added state inside $H$. The fourth element $s'$ is the state inside the original DES reachable by the reference of $s$ over the event $\sigma$ which is the third argument.

This procedure checks if $s'$ is already visited. If so, connects $s$ with the first representative of $s'$ inside $H$. Otherwise if all outgoing transitions defined from $s'$ are inside the subset $\Sigma_i$, then a new simple state as the representative of $s'$ is added to $H$. If not, a superstate is created to hide the events in $\Sigma - \Sigma_i$ inside itself. Procedure2 completes inside that superstate.

**Algorithm 2** Procedure1

```
procedure PROCEDURE1($H, s, e, s'$)
    if $s'$.color == gray then
        $H \leftarrow$ new transition $t(s, e, s'')$          $\rightarrow s'' = s'$.rep[0]
    else
        $s'$.color = gray
        $s'' \leftarrow$ new simple state
        $s''$.ref = $s'$
        $s'$.rep.insert($s''$)
        if $(\forall\sigma \in \Sigma)\delta(s', \sigma)! \Rightarrow \sigma \in \Sigma_i$ then
            $H.X = H.X \cup \{s''\}$
            for all $s''' \in adj[s']$ do          $\rightarrow (\exists\sigma' \in \Sigma)\delta(s', \sigma')! = s'''$
                Procedure1($H, s'', \sigma', s'''$)
            end for
        else
            $H' \leftarrow$ new superstate
            $H'.X = H'.X \cup \{s''\}$
            Procedure2($H, H', s'', s'$)
        end if
    end if
end procedure
```

**Algorithm 3** Procedure2

```
procedure PROCEDURE2($H, H', s, s'$)
    for all $s'' \in adj[s']$ do          $\rightarrow (\exists\sigma \in \Sigma)\delta(s', \sigma)! = s''$
        if $\sigma \in \Sigma_i$ then
            Procedure1($H, s, \sigma, s''$)
        else if $s''$.color == gray && $(\exists s''' \in H'.x)s''' \in s'$.rep then
            $H' \leftarrow$ new transition $t(s, \sigma, s''')$
        else
            $s'$.color = gray
            $s''' \leftarrow$ new simple state
            $s'''$.ref = $s''$
            $s''$.rep.insert($s'''$)
            Procedure2($H, H', s''', s''$)
        end if
    end for
end procedure
```

## 4 Formalism

**Theorem 1.** L(DES) = L(MDESLO).

*Proof.* □

**Theorem 2.** L(PDES) = L(MDESHI).

*Proof.* □

## References

[1] Chuan Ma and W. Murray Wonham. *Nonblocking Supervisory Control of State Tree Structures (Lecture Notes in Control and Information Sciences)*. Springer, 2008.