

## Response Rules with Variable Number of Intermediate Stages

The family of **CHAIN** rules is adequate for dealing with cases in which the number of intermediate stages in the progress from  $p$  to  $q$  is bounded by a constant (5 for the case of BAKERY-2).

However, there are cases in which the number of intermediate stages cannot be bounded by a constant. Consider the trivial case of a sequential terminating loop.

```

local  $y$  natural
 $l_0$  : while  $y > 0$  do
       $l_1$  :  $y := y - 1$ 
 $l_2$  :
  
```

Termination of this program can be specified by the response formula

$$at\_l_0 \Rightarrow \diamond at\_l_2$$

How can we prove it?

Obviously, rule **CHAIN** cannot be used, because the number of intermediate stages depend on the initial value of variable  $y$ .

There are however, some principles which are retained from rule **CHAIN**. We would like to have some **measure of progress** in the journey from  $p$  to  $q$ . Every intermediate stage should be associated with a **helpful transition**  $t$  such that activation of  $t$  **decreases** the measured distance to  $q$ , and activation of any  $\tau \neq t$  at least does not increase the distance. In rule **CHAIN**, the distance was measured by the index of the assertion  $h_i$  holding at the current state. In more general rules, we will introduce an explicit **distance** (also called **ranking**) function.

## Possible Domains for the Ranking Function

In the example of the terminating loop, it is possible to take  $y$  as the ranking function. Its range is the natural numbers. This is not always adequate.

We define a **well-founded domain** to be a pair  $(\mathcal{A}, \succ)$  consisting of a domain  $\mathcal{A}$  and an ordering relation  $\succ$  over  $\mathcal{A}$  such that there does not exist an infinitely descending sequence

$$a_0 \succ a_1 \succ \dots$$

of  $\mathcal{A}$ -elements.

For example, the natural numbers with the  $>$  ordering forms a well-founded domain, denoted  $(\mathbb{N}, >)$ . When there is no danger of confusion, we refer to the well-founded domain  $(\mathcal{A}, \succ)$ , simply as  $\mathcal{A}$ . For elements  $a, b \in \mathcal{A}$ , we write  $a \succeq b$  if either  $a \succ b$  or  $a = b$ .

## Composite Well-Founded Domains

Given two well-founded domains  $(\mathcal{A}_1, \succ_1)$  and  $(\mathcal{A}_2, \succ_2)$ , we introduce two ways to construct a composite well-founded domain.

The **cross product**  $\mathcal{A}_1 \times \mathcal{A}_2$  is the well-founded domain  $(\mathcal{A}, \succ)$ , where  $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$  and

$$(a_1, a_2) \succ (b_1, b_2) \iff (a_1 \succ_1 b_1 \wedge a_2 \succeq_2 b_2) \vee (a_1 \succeq_1 b_1 \wedge a_2 \succ_2 b_2)$$

The **lexicographic product**  $\mathcal{A}_1 \times_{lex} \mathcal{A}_2$  is the well-founded domain  $(\mathcal{A}, \succ)$ , where  $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$  and

$$(a_1, a_2) \succ_{lex} (b_1, b_2) \iff (a_1 \succ_1 b_1) \vee (a_1 = b_1 \wedge a_2 \succ_2 b_2)$$

**Claim 12.** *If both  $(\mathcal{A}_1, \succ_1)$  and  $(\mathcal{A}_2, \succ_2)$  are well-founded, then so are  $\mathcal{A}_1 \times \mathcal{A}_2$  and  $\mathcal{A}_1 \times_{lex} \mathcal{A}_2$ .*

**Proof** It is sufficient to show that  $\mathcal{A}_1 \times_{lex} \mathcal{A}_2$  is well-founded.

Assume to the contrary, that there exists an infinitely descending sequence

$$(a_1, b_1) \succ_{lex} (a_2, b_2) \succ_{lex} \dots$$

From the definition of  $\succ_{lex}$  it follows that the sequence of first pair members satisfies  $a_1 \succeq_1 a_2 \succeq_1 \dots$ . Since  $\mathcal{A}_1$  is well founded, it follows that there exists some position  $k$  such that  $a_k = a_{k+1} = \dots$ . Therefore, the sequence  $b_k \succ_2 b_{k+1} \succ_2 \dots$  must be infinitely descending, contradicting the well-foundedness of  $\mathcal{A}_2$ . ▀

## Rule WELL

### Rule WELL

For a well-founded domain  $(\mathcal{A}, \succ)$

For just statements  $t_1, \dots, t_m,$

assertions  $p, q = h_0, h_1, \dots, h_m,$

and ranking functions  $\delta_1, \dots, \delta_m : \Sigma \mapsto \mathcal{A}$

$$\text{W1. } p \Rightarrow \bigvee_{j=0}^m h_j$$

For  $i = 1, \dots, m$

$$\text{W2. } h_i \wedge \rho_t \Rightarrow (h'_i \wedge \delta_i = \delta'_i) \vee \bigvee_{j=0}^m (h'_j \wedge \delta_i \succ \delta'_j) \quad \text{For every } t \neq t_i$$

$$\text{W3. } h_i \wedge \rho_{t_i} \Rightarrow \bigvee_{j=0}^m (h'_j \wedge \delta_i \succ \delta'_j)$$

$$\text{W4. } h_i \Rightarrow \text{En}(t_i)$$

---


$$p \Rightarrow \diamond q$$

## Soundness of Rule WELL

**Claim 13.** Rule WELL is sound for proving the response property  $p \Rightarrow \diamond q$ .

**Proof** Assume that the premises of rule WELL are valid. Let  $\sigma : s_0, s_1, \dots$  be a computation of  $\mathcal{D}$  and let  $p$  hold at position  $j$ . We have to show that there exists a position  $k \geq j$  such that  $q$  holds at position  $k$ .

Assume to the contrary, that no position beyond  $j$  satisfies  $q$ . By premise W1, state  $s_j$  must satisfy  $h_i$ , for some  $i \geq 0$ . Since  $q$  never holds beyond  $j$ , we must have  $i > 0$ . Let us denote by  $i_j$  the index  $i > 0$  such that  $h_i$  holds at state  $s_j$ , and by  $d_j \in \mathcal{A}$  the value of  $\delta_{i_j}$  at state  $s_j$ . By premise W2, the successor state  $s_{j+1}$  must also satisfy  $h_i$ , for some  $i$ . Denote this index by  $i_{j+1}$ . By argument similar to the above,  $i_{j+1} > 0$ . In this way we proceed to establish an infinite sequence of indices  $i_j, i_{j+1}, \dots$  where, for each  $k \geq j$ ,  $i_k > 0$  and  $s_k \models h_{i_k}$ . Let us denote by  $d_j, d_{j+1}, \dots$  the sequence of values of the corresponding ranking functions at the respective states. By premises W2 and W3, the sequence  $d_j \succeq d_{j+1} \succeq \dots$  is non-increasing. Since this is an infinite non-increasing sequence over a well-founded domain, there must exist an index  $n$ , such that  $d_n = d_{n+1} = \dots$ , and consequently (due to W2)  $i_n = i_{n+1} = \dots$ .

By premise W3, we can have  $i_n = i_{n+1} = \dots = i$  only if statement  $t_i$  is never executed beyond position  $n$ . On the other hand, due to W4, statement  $t_i$  is continuously enabled beyond  $n$ . Thus,  $\sigma$  violates the justice requirement associated with statement  $t_i$ , and therefore is not a computation, contrary to our original assumption.

We conclude that there must exist a position  $k \geq j$  satisfying  $q$ . ▀

# Application to Program UP-DOWN

$x, y$ : **natural initially**  $x = y = 0$

$$P_1 :: \left[ \begin{array}{l} l_0 : \text{while } x = 0 \text{ do} \\ \quad [l_1 : y := y + 1] \\ l_2 : \text{while } y > 0 \text{ do} \\ \quad [l_3 : y := y - 1] \\ l_4 : \end{array} \right] \quad \parallel \quad P_2 :: \left[ \begin{array}{l} m_0 : x := 1 \\ m_1 : \end{array} \right]$$

We wish to prove, using rule **WELL**, the response property

$$at\_l_0 \wedge at\_m_0 \Rightarrow \diamond (at\_l_4 \wedge at\_m_1)$$

As a well-founded domain, we choose  $\mathcal{A} = \mathbf{N} \times_{lex} \mathbf{N} \times_{lex} \mathbf{N}$ . The other constructs are given by:

| $i$ | $t_i$ | $h_i$                                   | $\delta_i$  |
|-----|-------|---|-------------|
| 0   | —     | $at\_l_4 \wedge at\_m_1$                | $(0, 0, 0)$ |
| 1   | $l_3$ | $at\_l_3 \wedge at\_m_1 \wedge y > 0$   | $(0, y, 1)$ |
| 2   | $l_2$ | $at\_l_2 \wedge at\_m_1$                | $(0, y, 2)$ |
| 3   | $l_1$ | $at\_l_1 \wedge at\_m_1 \wedge (x = 1)$ | $(2, 0, 0)$ |
| 4   | $l_0$ | $at\_l_0 \wedge at\_m_1 \wedge (x = 1)$ | $(1, 0, 0)$ |
| 5   | $m_0$ | $at\_l_{0,1} \wedge at\_m_0$            | $(3, 0, 0)$ |

## A Rule with Distributed Ranking

In many cases of parameterized systems  $P_1 \parallel \dots \parallel P_n$ , it is possible to identify a global ranking which can be presented as the cross-product  $\delta_1 \times \dots \times \delta_n$ . This leads to the following rule **DISTR-RANK**.

### Rule **DISTR-RANK**

For a well-founded domain  $(\mathcal{A}, \succ)$

For just statements  $t_1, \dots, t_m$ ,

assertions  $p, q = h_0, h_1, \dots, h_m$ ,

and ranking functions  $\delta_1, \dots, \delta_m : \Sigma \mapsto \mathcal{A}$

$$\text{W1. } p \Rightarrow \bigvee_{j=0}^m h_j$$

For  $i = 1, \dots, m$

$$\text{W2. } h_i \wedge \rho_t \Rightarrow h'_i \vee \left( \left( \bigvee_{j=0}^m h'_j \right) \wedge \left( \bigvee_{j=1}^m (\delta_j \succ \delta'_j) \right) \right) \quad \text{For every } t \neq t_i$$

$$\text{W3. } h_i \wedge \rho_{t_i} \Rightarrow \left( \bigvee_{j=0}^m h'_j \right) \wedge \left( \bigvee_{j=1}^m (\delta_j \succ \delta'_j) \right)$$

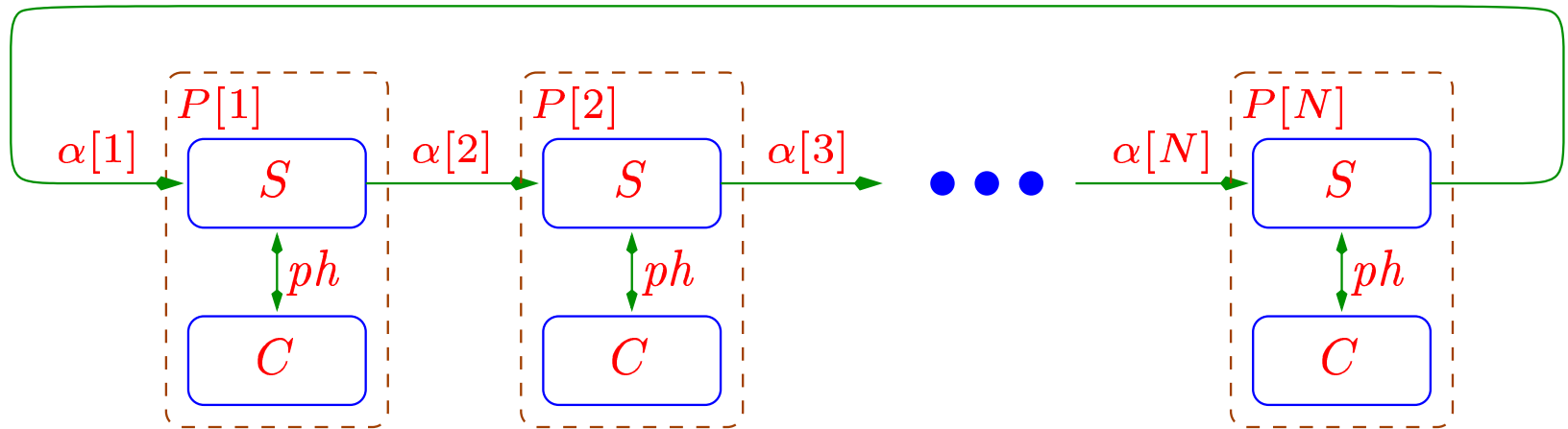
$$\text{W4. } h_i \wedge \rho \Rightarrow \bigwedge_{j=1}^m (\delta_j \succeq \delta'_j)$$

$$\text{W5. } h_i \Rightarrow \text{En}(t_i)$$

---


$$p \Rightarrow \diamond q$$

## Example: Mutual Exclusion by Token Passing



**local**  $\alpha : \text{array}[1..N]$  of boolean where  $\alpha[1] = 1, \alpha[2] = \dots = \alpha[N] = 0$

$$\begin{array}{l} \parallel_{i=1}^N P[i] :: \\ \left[ \begin{array}{l} S :: \left[ \begin{array}{l} l_0 : \text{loop forever do} \\ \left[ \begin{array}{l} l_1 : \text{request } \alpha[i] \\ l_2 : \text{if } at\_m_2[i] \text{ then} \\ \left[ l_3 : \text{await } at\_m_4[i] \right] \\ l_4 : \text{release } \alpha[i \oplus 1] \end{array} \right] \end{array} \right] \\ \\ C :: \left[ \begin{array}{l} m_0 : \text{loop forever do} \\ \left[ \begin{array}{l} m_1 : \text{Non-critical} \\ m_2 : \text{await } at\_l_3[i] \\ m_3 : \text{Critical} \\ m_4 : \text{await } \neg at\_l_3[i] \end{array} \right] \end{array} \right] \end{array} \right] \end{array}$$

## First Some Invariants

**local**  $\alpha : \text{array}[1..N]$  **of boolean** where  $\alpha[1] = 1, \alpha[2] = \dots = \alpha[N] = 0$

|  |   |   |       |                        |   |
|--|---|---|-------|------------------------|---|
|  |   |   | $l_0$ | <b>loop forever do</b> |   |
|  | [ |   |       |                        |   |
|  |   | S | ::    |                        |   |
|  |   |   |       | [                      |   |
|  |   |   |       |                        | l <sub>1</sub> : request $\alpha[i]$          |
|  |   |   |       |                        | l <sub>2</sub> : if $at\_m_2[i]$ then         |
|  |   |   |       |                        | [ l <sub>3</sub> : await $at\_m_4[i]$ ]       |
|  |   |   |       |                        | l <sub>4</sub> : release $\alpha[i \oplus 1]$ |
|  |   |   |       |                        | ]   |
|  |   |   |       | ]                      |   |
|  |   |   |       |                        |   |
|  |   |   |       |                        |   |
|  |   |   |       |                        |   |
|  |   | C | ::    |                        |   |
|  |   |   |       | [                      |   |
|  |   |   |       |                        | m <sub>0</sub> : loop forever do              |
|  |   |   |       |                        | [ m <sub>1</sub> : Non-critical ]             |
|  |   |   |       |                        | m <sub>2</sub> : await $at\_l_3[i]$           |
|  |   |   |       |                        | m <sub>3</sub> : Critical                     |
|  |   |   |       |                        | m <sub>4</sub> : await $\neg at\_l_3[i]$      |
|  |   |   |       |                        | ]   |
|  |   |   |       | ]                      |   |

$\prod_{i=1}^N P[i]$

The following are invariants of TOKEN-RING:

$$\varphi_1 : \sum_{i=1}^N (\alpha[i] + at\_l_{2..4}[i]) = 1$$

$$\varphi_2 : at\_m_3[i] \rightarrow at\_l_3[i]$$

Together they imply **mutual exclusion!**

## Now to Liveness

Accessibility can be specified by

$$at\_m_2[z] \Rightarrow \diamond at\_m_3[z]$$

for some process  $z : [1..N]$ .

We define the **cyclic distance** between  $j$  and  $z$  as

$$\Delta(j, z) = (z - j) \bmod N$$

This is the number of times  $j$  is incremented by 1 modulo  $N$  until it reaches  $z$ .

The choice of helpful transitions and ranking functions for use in rule **WELL** is given by the following table:

| Trans. $t$ | $h(t)$   | $\delta(t)$                  | Successors                         |
|------------|--|------------------------------|------------------------------------|
| $l_0[i]$   | $at\_m_2[z] \wedge at\_l_0[i] \wedge \alpha[i]$  | $(\Delta(i, z), 6)$          | $l_1[i]$                           |
| $l_1[i]$   | $at\_m_2[z] \wedge at\_l_1[i] \wedge \alpha[i]$  | $(\Delta(i, z), 5)$          | $l_2[i]$                           |
| $l_2[i]$   | $at\_m_2[z] \wedge at\_l_2[i]$                   | $(\Delta(i, z), 4)$          | $l_4[i], m_2[i]$                   |
| $m_2[i]$   | $at\_m_2[z] \wedge at\_m_2[i] \wedge at\_l_3[i]$ | $(\Delta(i, z), 3)$          | $m_3[i], at\_m_3[z]$               |
| $m_3[i]$   | $at\_m_2[z] \wedge at\_m_3[i]$                   | $(\Delta(i, z), 2)$          | $l_3[i]$                           |
| $l_3[i]$   | $at\_m_2[z] \wedge at\_l_3[i] \wedge at\_m_4[i]$ | $(\Delta(i, z), 1)$          | $l_4[i]$                           |
| $l_4[i]$   | $at\_m_2[z] \wedge at\_l_4[i]$                   | $(\Delta(i \oplus 1, z), 7)$ | $l_0[i \oplus 1], l_1[i \oplus 1]$ |

# The BAKERY Algorithm

$N$  : natural where  $N > 0$   
 $y$  : array[1.. $N$ ] of natural where  $y = 0$

$\prod_{i=1}^N P[i] ::$

$l_0$ : loop forever do

$l_1$ : Non-critical  
 $l_2$ :  $y[i] := \max(y[1], \dots, y[N]) + 1$   
 $l_3$ : await  $\forall j \neq i : y[j] = 0 \vee y[i] < y[j]$   
 $l_4$ : Critical  
 $l_5$ :  $y[i] := 0$

Program BAKERY: the Bakery Algorithm.

Some useful invariants:

$$\varphi_1 : y[i] > 0 \iff at_{l_{3..5}}[i]$$

$$\varphi_2 : at_{l_{4,5}}[i] \rightarrow \underbrace{\forall j \neq i : y[j] = 0 \vee y[i] < y[j]}_{\mu(i)}$$

Together, they imply mutual exclusion.

## Verifying Accessibility

Next, let us verify accessibility, specifiable by

$$at\_l_2[z] \Rightarrow \diamond at\_l_4[z]$$

We intend to use rule **DISTR-RANK**. For a transition  $t$ , we will define

- $\delta(t) = 1$  If  $t$  is currently enabled. This is also the case that  $t$  is helpful.
- $\delta(t) = 2$  If  $t$  is currently disabled, but may become helpful on the way from  $p$  to  $q$ .
- $\delta(t) = 0$  If  $t$  is disabled and can never become helpful before  $q$  is achieved.

The following table identifies for all transitions  $t$  when they are helpful (and therefore  $\delta(t) = 1$ ):

| $t$      | $h(t)$                                       |
|----------|--|
| $l_5[i]$ | $at\_l_3[z] \wedge at\_l_5[i]$               |
| $l_4[i]$ | $at\_l_3[z] \wedge at\_l_4[i]$               |
| $l_3[i]$ | $at\_l_3[z] \wedge at\_l_3[i] \wedge \mu(i)$ |
| $l_2[i]$ | $i = z \wedge at\_l_2[i]$                    |

The next table identifies for all transitions  $t$  when they have the distributed rank  $\delta(t) = 2$ , and may therefore become helpful in the future:

| $t$      | $\delta(t) = 2$  |
|----------|--|
| $l_5[i]$ | $at\_l_2[z] \vee at\_l_3[z] \wedge at\_l_{3,4}[i] \wedge y[i] < y[z]$                            |
| $l_4[i]$ | $at\_l_2[z] \vee at\_l_3[z] \wedge at\_l_3[i] \wedge y[i] < y[z]$                                |
| $l_3[i]$ | $at\_l_2[z] \vee at\_l_3[z] \wedge at\_l_3[i] \wedge \neg\mu(i) \wedge (i = z \vee y[i] < y[z])$ |

For all other transitions and all other cases,  $\delta(t) = 0$ .

## Alternately, Using Rule WELL

We can also use rule **WELL** for proving the accessibility property

$$at\_l_2[z] \Rightarrow \diamond at\_l_4[z]$$

for the **BAKERY** algorithm.

Define a ranking function

$$\Delta = |\{i \mid 0 < y[i] \leq y[z]\}|$$

which counts the number of processes with positive tickets whose values do not exceed the value of  $y[z]$ .

The following table summarizes the helpful transitions and their rankings as required by rule **WELL**:

| $t$      | $h(t)$                                       | $\delta(t)$      | Successors           |
|----------|--|------------------|----------------------|
| $l_2[z]$ | $at\_l_2[z]$                                 | $(1, 0, 0)$      | $\{l_{3,4,5}[j]\}$   |
| $l_3[i]$ | $at\_l_3[z] \wedge at\_l_3[i] \wedge \mu(i)$ | $(0, \Delta, 2)$ | $l_4[i], at\_l_4[z]$ |
| $l_4[i]$ | $at\_l_3[z] \wedge at\_l_4[i]$               | $(0, \Delta, 1)$ | $l_5[i]$             |
| $l_5[i]$ | $at\_l_3[z] \wedge at\_l_5[i]$               | $(0, \Delta, 0)$ | $\{l_3[j]\}$         |

## Verification Diagrams

Up to now, we have presented the constituents of a proof by rules **CHAIN**, **WELL**, or **DISTR-RANK** by tables. An alternate presentation is provided by **verification diagrams**. A verification diagram is a directed graph such that:

- Nodes contain labeled assertions, identifying **helpful situations**.
- There exists a single node with no successors, called the **terminal node**, and labeled by the **goal assertion**  $q$ .
- Every node has a distinguished edge departing from it, and labeled by a transition which is **helpful** for this node. A node may have additional multiple unhelpful (indifferent) edges departing from it.

Diagrams differ by the rule they represent.

## Chain Diagrams

It is required that

- The terminal node is labeled by  $h_0 : q$ .
- If there is an edge connecting node  $h_i$  to node  $h_j$ , then  $i > j$ .

Assume that non-terminal node  $h_i$  has the helpful transition  $t_i$  which connects it to node  $h_j$  and the unhelpful successors  $h_{k_1}, \dots, h_{k_n}$ . This implies the following verification conditions:

$$\begin{aligned} \text{C2. } h_i \wedge \rho_t &\Rightarrow h'_i \vee h'_{k_1} \vee \dots \vee h'_{k_n} && \text{For every } t \neq t_i \\ \text{C3. } h_i \wedge \rho_{t_i} &\Rightarrow h'_j \\ \text{C4. } h_i &\Rightarrow \text{En}(t_i) \end{aligned}$$

A **CHAIN** diagram is defined to be  **$\mathcal{D}$ -valid** if all the verification conditions associated with its nodes are  **$\mathcal{D}$ -valid**.

**Claim 14.** *If a verification diagram with nodes  $h_0, \dots, h_n$  is  $\mathcal{D}$ -valid then so is the temporal formula*

$$\bigvee_{i=0}^n h_i \Rightarrow \diamond h_0$$

**Corollary 15.** *If, in addition, we establish the  $\mathcal{D}$ -validity of*

$$p \Rightarrow \bigvee_{i=0}^n h_i \quad \text{and} \quad h_0 \Rightarrow q$$

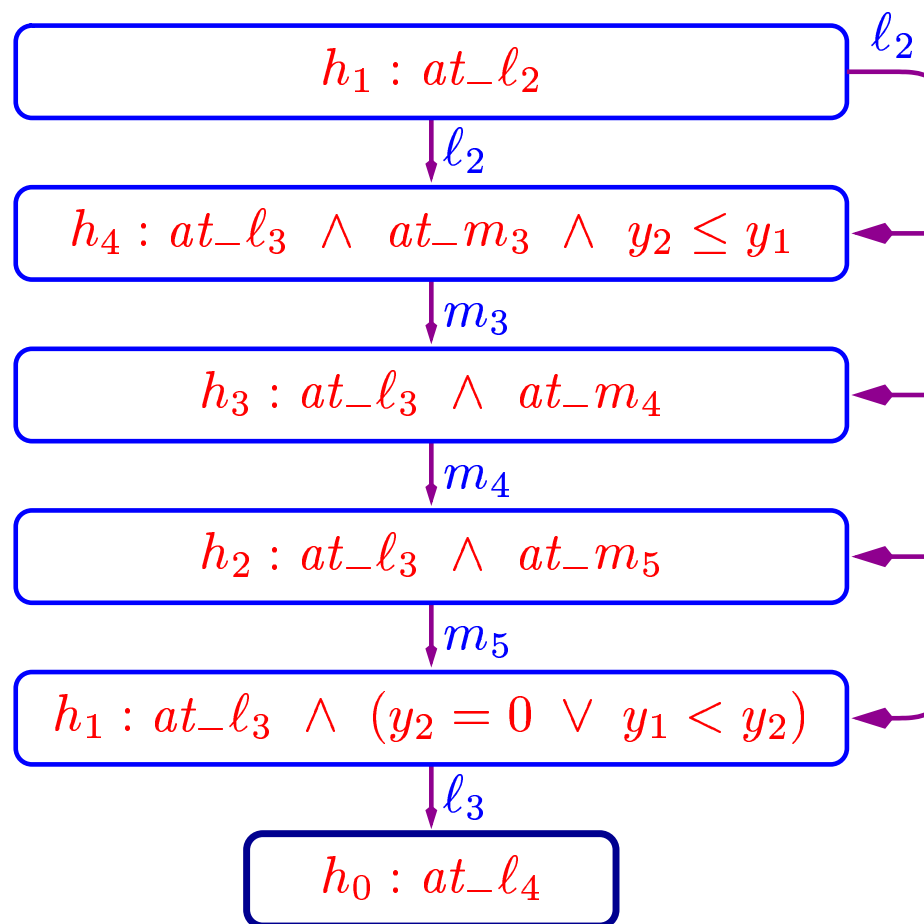
*then we can conclude*

$$p \Rightarrow \diamond q$$

## Example: BAKERY-2

local  $y_1, y_2$  : natural initially  $y_1 = y_2 = 0$

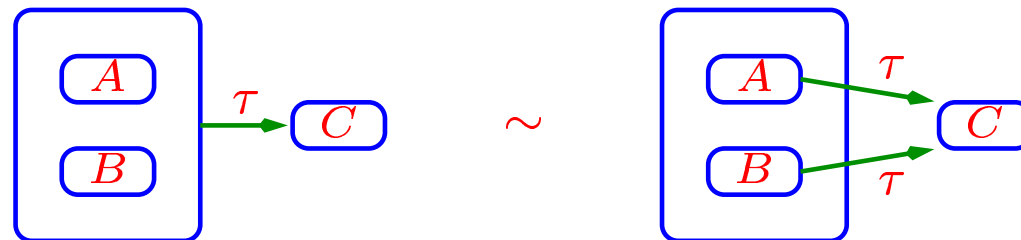
$$P_1 :: \left[ \begin{array}{l} l_0 : \text{loop forever do} \\ \left[ \begin{array}{l} l_1 : \text{Non-Critical} \\ l_2 : y_1 := y_2 + 1 \\ l_3 : \text{await } y_2 = 0 \vee y_1 < y_2 \\ l_4 : \text{Critical} \\ l_5 : y_1 := 0 \end{array} \right] \end{array} \right] \parallel P_2 :: \left[ \begin{array}{l} m_0 : \text{loop forever do} \\ \left[ \begin{array}{l} m_1 : \text{Non-Critical} \\ m_2 : y_2 := y_1 + 1 \\ m_3 : \text{await } y_1 = 0 \vee y_2 \leq y_1 \\ m_4 : \text{Critical} \\ m_5 : y_2 := 0 \end{array} \right] \end{array} \right]$$



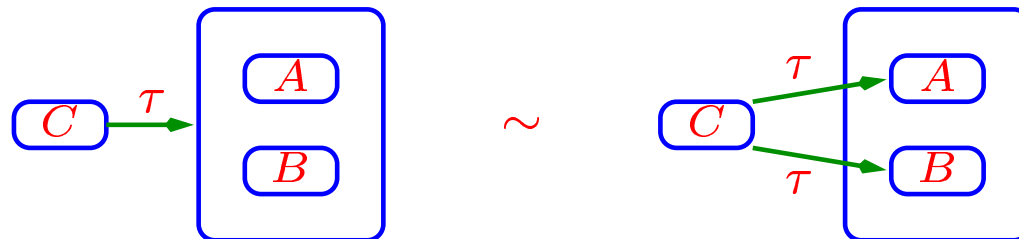
## Encapsulation (Statecharts) Conventions

There are several conventions which make visual presentation more effective. We introduce **compound nodes** which may contains several internal nodes. The following graphical equivalences explain the conventions:

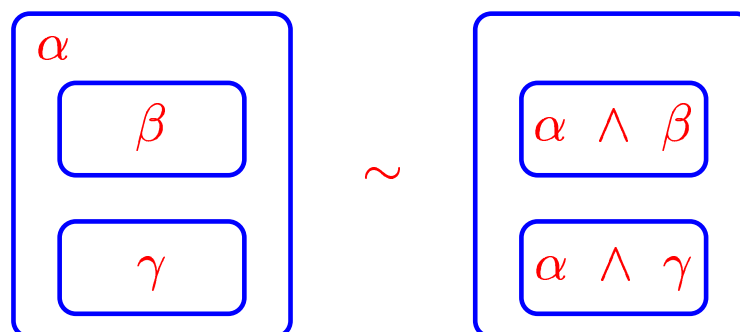
### Departing Edges:



### Arriving Edges:

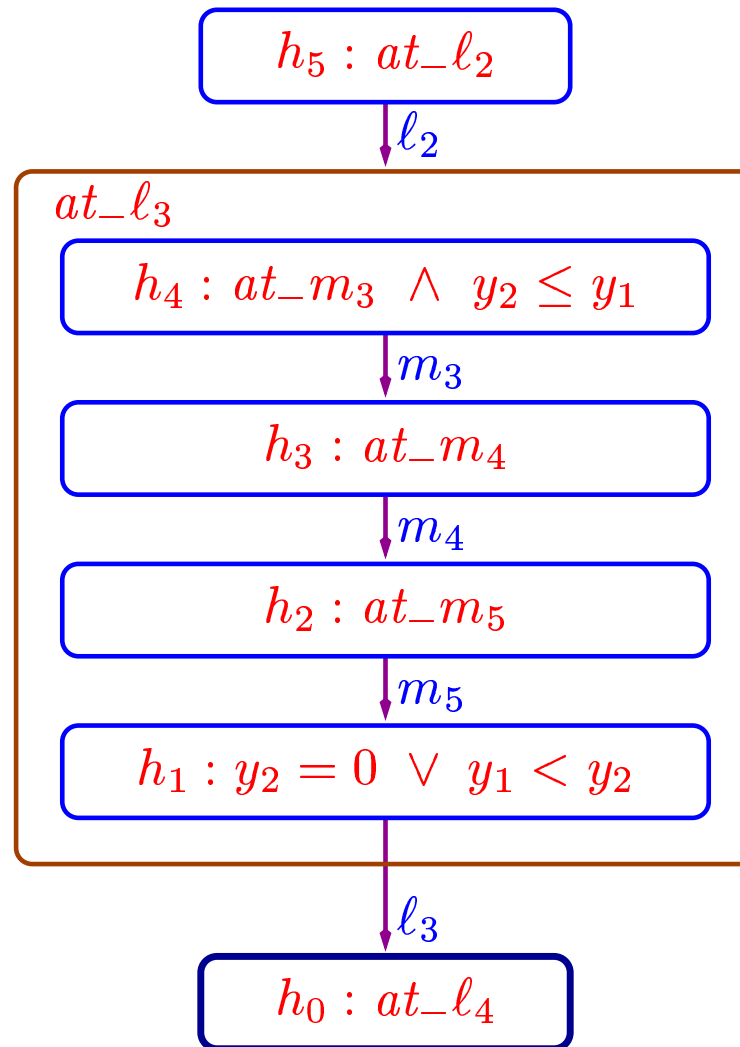
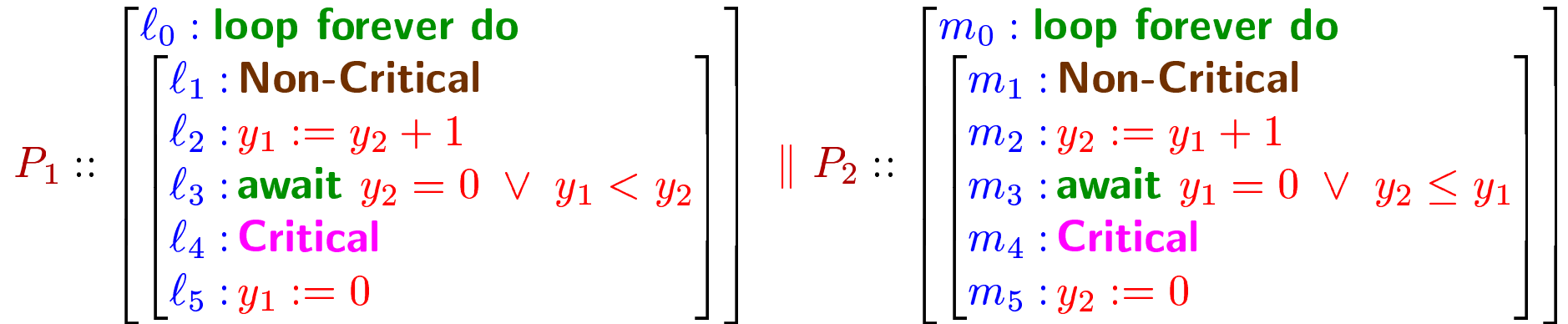


### Common Factors:



# Encapsulated Verification Diagram for BAKERY-2

**local**  $y_1, y_2$  : **natural initially**  $y_1 = y_2 = 0$



## WELL Diagrams

Node  $h_i$  contains also a ranking function  $\delta_i$ . It is required that  $\delta_0 = 0$ .

Assume that non-terminal node  $h_i$  has the helpful transition  $t_i$  which connects it to node  $h_j$  and the unhelpful successors  $h_{k_1}, \dots, h_{k_n}$ . This implies the following verification conditions:

$$\begin{aligned} \text{W2. } h_i \wedge \rho_t &\Rightarrow (h'_i \wedge \delta_i \succeq \delta'_i) \vee \\ &\quad (h'_{k_1} \wedge \delta_i \succ \delta'_{k_1}) \vee \dots \vee (h'_{k_n} \wedge \delta_i \succ \delta'_{k_n}) \quad \text{For every } t \neq t_i \\ \text{W3. } h_i \wedge \rho_{t_i} &\Rightarrow h'_j \wedge \delta_i \succ \delta'_j \\ \text{W4. } h_i &\Rightarrow \text{En}(t_i) \end{aligned}$$

A **WELL** diagram is defined to be  **$\mathcal{D}$ -valid** if all the verification conditions associated with its nodes are  **$\mathcal{D}$ -valid**.

**Claim 16.** *If a **WELL** verification diagram with nodes  $h_0, \dots, h_n$  is  **$\mathcal{D}$ -valid** then so is the temporal formula*

$$\bigvee_{i=0}^n h_i \Rightarrow \diamond h_0$$

**Corollary 17.** *If, in addition, we establish the  **$\mathcal{D}$ -validity** of*

$$p \Rightarrow \bigvee_{i=0}^n h_i \quad \text{and} \quad h_0 \Rightarrow q$$

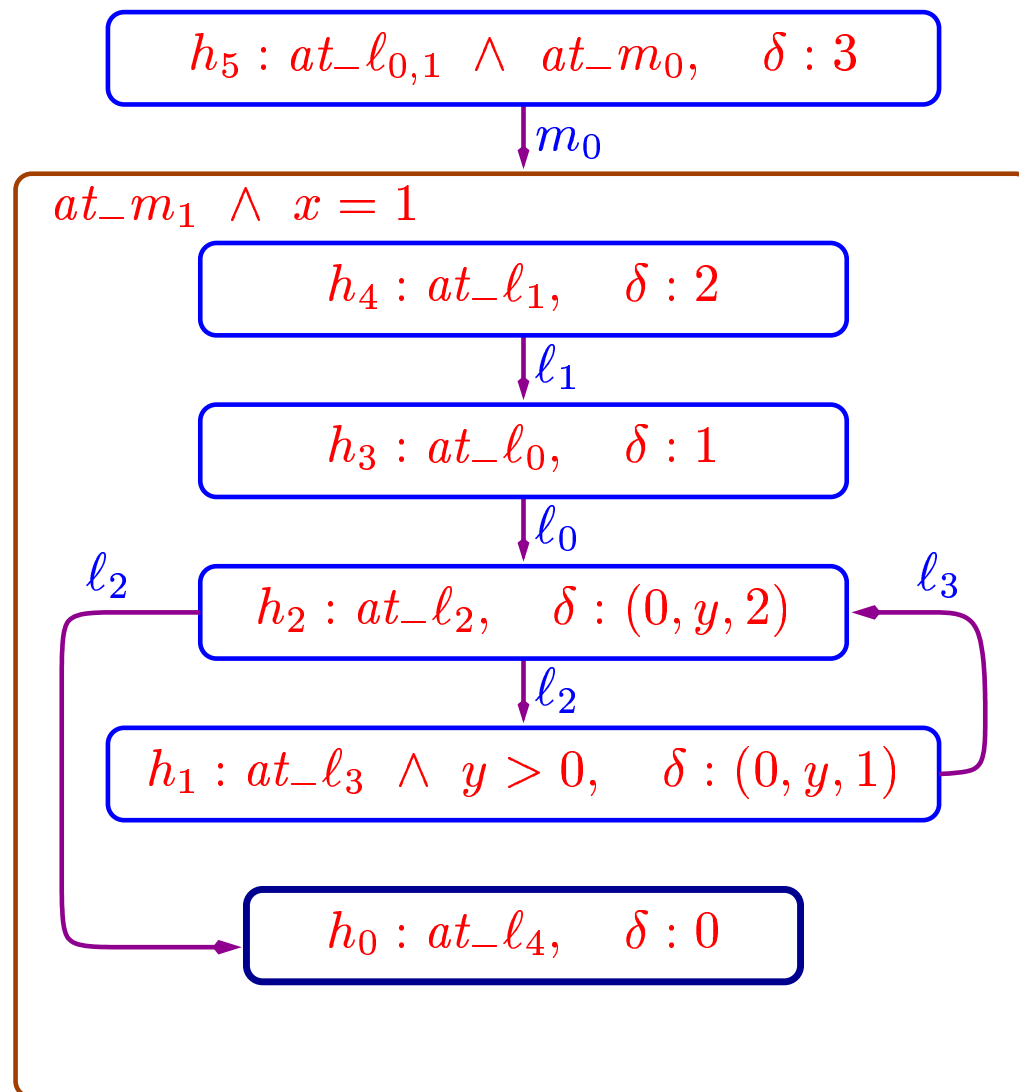
*then we can conclude*

$$p \Rightarrow \diamond q$$

# Apply to Program UP-DOWN

$x, y$ : **natural initially**  $x = y = 0$

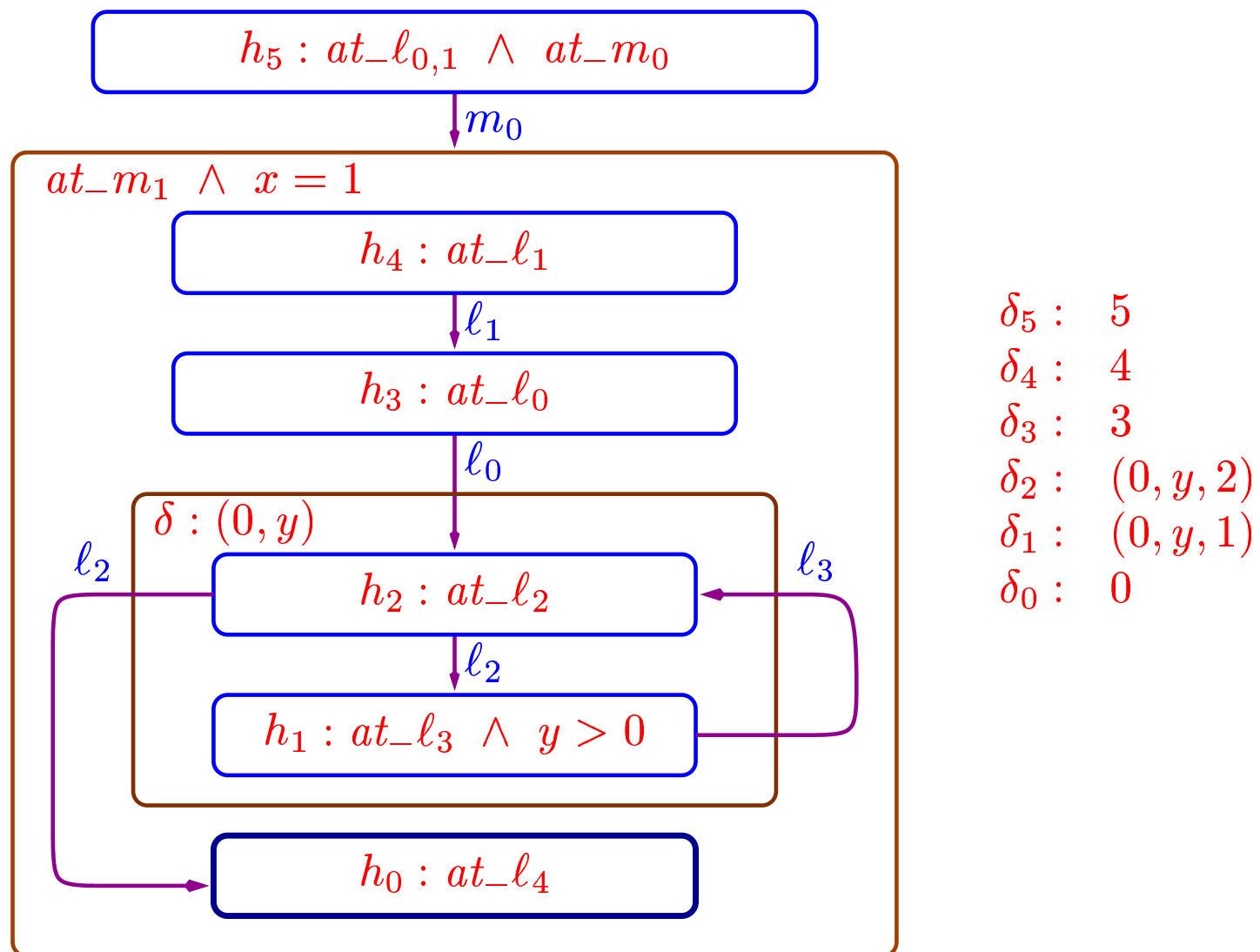
$$P_1 :: \left[ \begin{array}{l} l_0 : \text{while } x = 0 \text{ do} \\ \quad [l_1 : y := y + 1] \\ l_2 : \text{while } y > 0 \text{ do} \\ \quad [l_3 : y := y - 1] \\ l_4 : \end{array} \right] \quad || \quad P_2 :: \left[ \begin{array}{l} m_0 : x := 1 \\ m_1 : \end{array} \right]$$



## Encapsulation Conventions Concerning Ranking

We adopt the additional conventions:

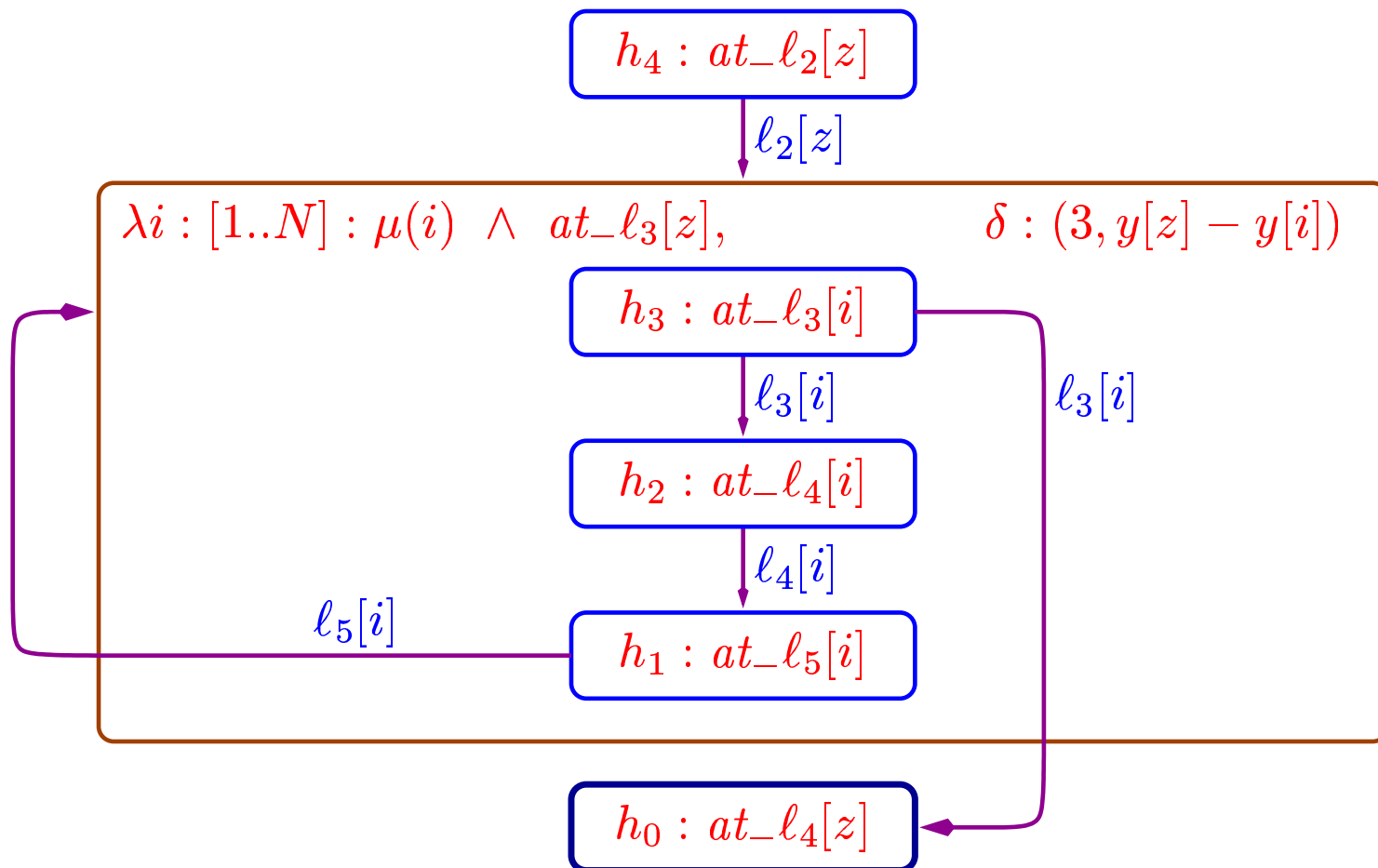
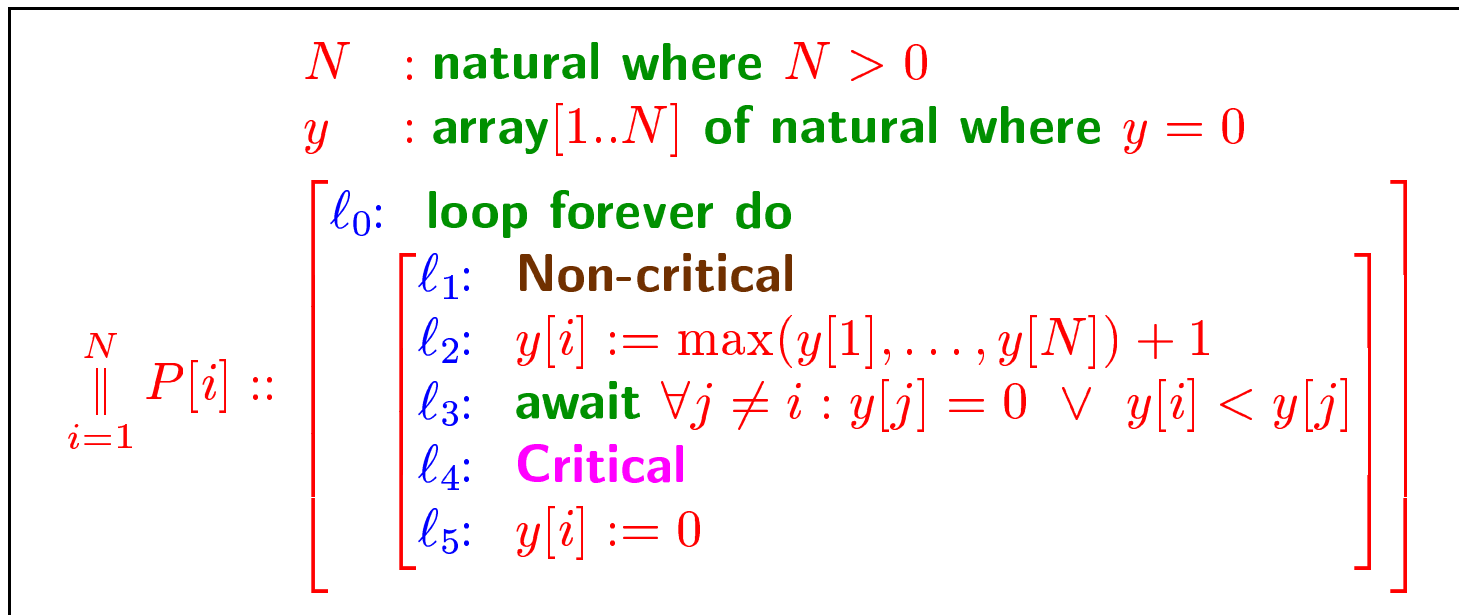
- In case node  $h_i$  does not have an explicit ranking labeling, it is as though it had the label  $\delta : i$ .
- In case a compound node has the transcription  $\delta : f$  at its top left corner, the factor  $f$  is added as a left lexicographic component to all the rankings of the contained nodes.



## Diagrams for Parameterized Systems

To deal with parameterized systems, we introduce the inscription  $\lambda i : [1..N]$  labeling a compound node. This is equivalent to having  $N$  copies of this node, one for each value of  $i \in [1..N]$ . Assertions and transitions within the node may be parameterized by  $i$ .

## Example: a Diagram for BAKERY



## Apply to TOKEN-RING

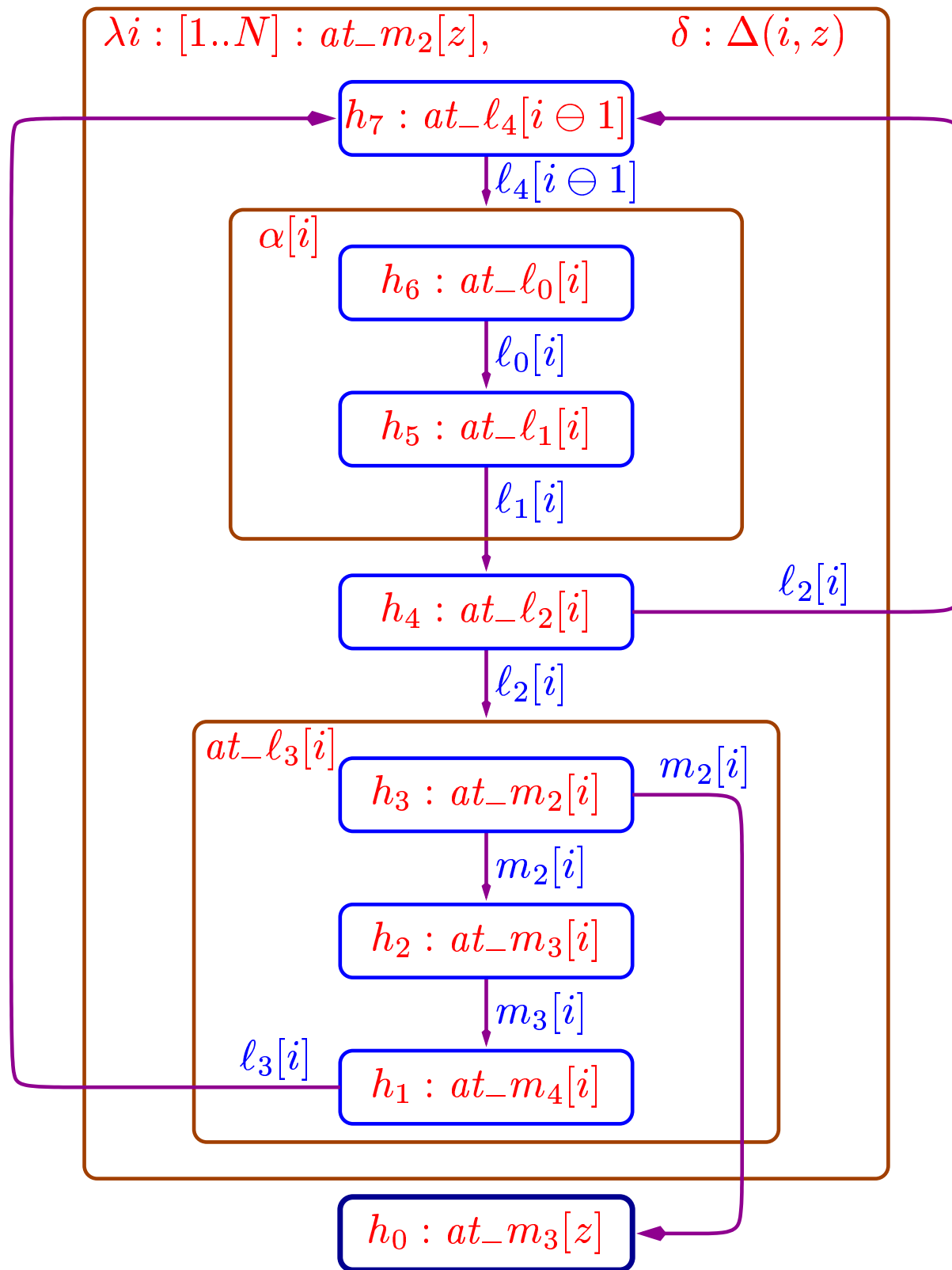
```

local  $\alpha$  : array[1.. $N$ ] of boolean where  $\alpha[1] = 1, \alpha[2] = \dots = \alpha[N] = 0$ 

 $\left[ \begin{array}{l} S :: \\ \\ C :: \end{array} \right] \parallel \left[ \begin{array}{l} l_0 : \text{loop forever do} \\ \left[ \begin{array}{l} l_1 : \text{request } \alpha[i] \\ l_2 : \text{if } at\_m_2[i] \text{ then} \\ \quad [l_3 : \text{await } at\_m_4[i]] \\ l_4 : \text{release } \alpha[i \oplus 1] \end{array} \right] \\ \\ m_0 : \text{loop forever do} \\ \left[ \begin{array}{l} m_1 : \text{Non-critical} \\ m_2 : \text{await } at\_l_3[i] \\ m_3 : \text{Critical} \\ m_4 : \text{await } \neg at\_l_3[i] \end{array} \right] \end{array} \right]$ 

```

# Diagram for TOKEN-RING



## Response Under Compassion

So far, we only considered proofs of response properties under the fairness requirements of justice. Consider now the more general case, where also compassion requirements are included. The following rule can be used to establish response properties for this general case:

### Rule RESP

For a well-founded domain  $(\mathcal{A}, \succ)$ ,

fair transitions  $t_1, \dots, t_m$ ,

assertions  $p, q = h_0, h_1, \dots, h_m$ ,

and ranking functions  $\delta_1, \dots, \delta_m : \Sigma \mapsto \mathcal{A}$

$$\text{R1. } p \Rightarrow \bigvee_{j=0}^m h_j$$

For  $i = 1, \dots, m$

$$\text{R2. } h_i \wedge \rho_t \Rightarrow (h'_i \wedge \delta_i = \delta'_i) \vee \bigvee_{j=0}^m (h'_j \wedge \delta_i \succ \delta'_j) \quad \text{For every } t \neq t_i$$

$$\text{R3. } h_i \wedge \rho_{t_i} \Rightarrow \bigvee_{j=0}^m (h'_j \wedge \delta_i \succ \delta'_j)$$

$$\text{R4. } h_i \Rightarrow \text{En}(t_i) \quad \text{If } t_i \text{ is a just transition}$$

$$\text{R5. } h_i \Rightarrow \diamond \text{En}(t_i) \quad \text{If } t_i \text{ is a compassionate transition}$$

---


$$p \Rightarrow \diamond q$$

Thus, while for a just transition  $t_i$ ,  $h_i$  should imply that  $t_i$  is enabled **now**, in the compassionate case,  $h_i$  only implies that  $t_i$  will be **eventually** enabled.

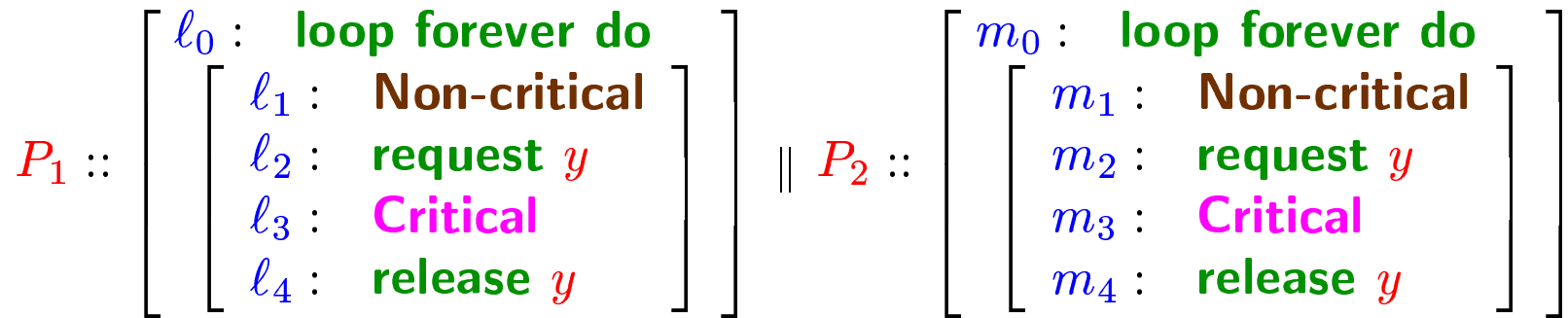
## Justification of the Rule

On the face of it, rule **RESP** may appear to be **circular**. In order to prove a response property it requires, as a premise, another response property.

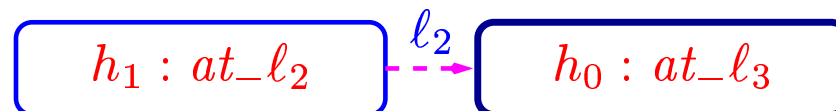
However, there is a certain reduction between the conclusion and the temporal premise. Namely, when establishing the eventual enableness of  $t_i$  we only consider computations which never activate  $t_i$  itself.

## Example: MUX-SEM for 2 Processes

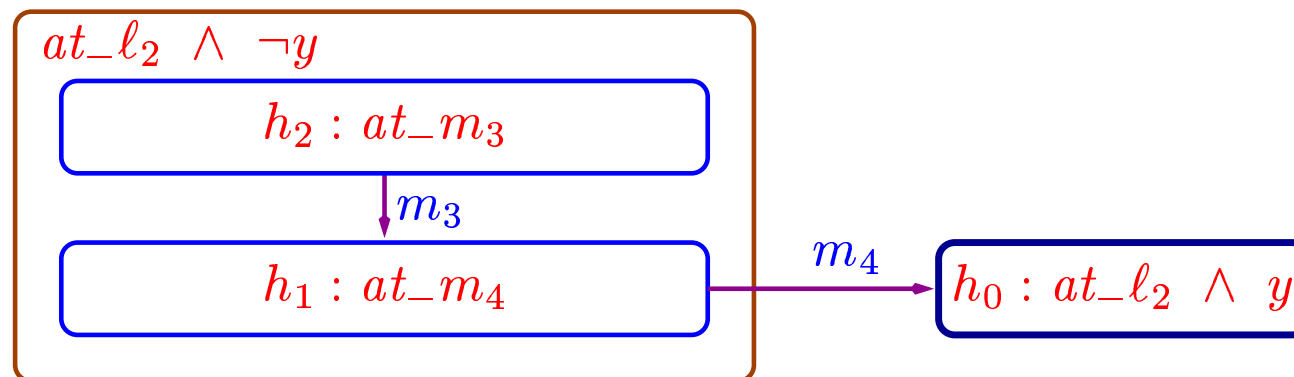
$y$ : natural initially  $y = 1$



Following is a verification diagram for the property  $at\_l_2 \Rightarrow \diamond at\_l_3$ :

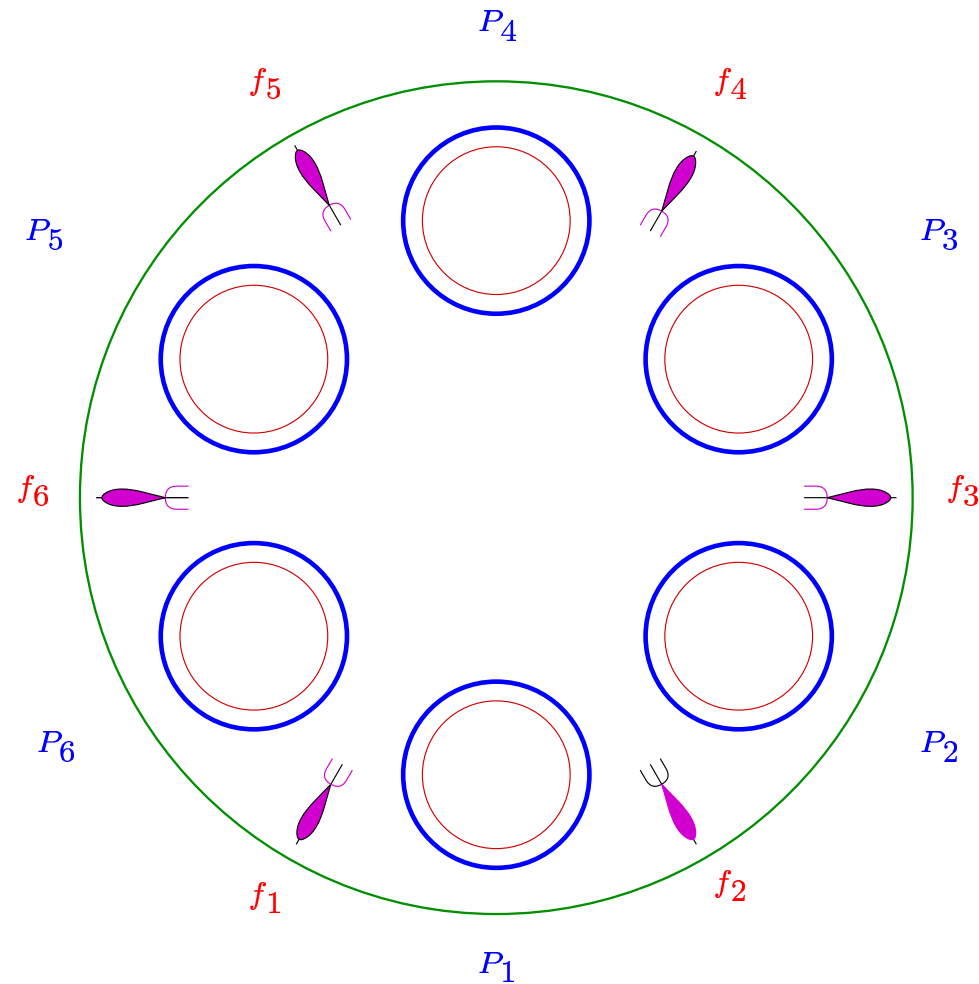


All the verification conditions generated by this verification diagram are non-temporal, except for the instance of premise R5 for transition  $l_2$  which has the form  $at\_l_2 \Rightarrow \diamond (at\_l_2 \wedge y)$ . Using the auxiliary invariant  $at\_l_{3,4} + at\_m_{3,4} + y = 1$ , the required temporal property can be established by the following verification diagram:



## The Dining Philosophers Metaphor

Consider  $n$  philosophers arranged around a table.



The life of a philosopher alternates between a **thinking phase** (a **non-critical** activity) and an **eating phase**. In order to eat, a philosopher needs **both** forks.

## Program Dine

A first attempt yields the following program **Dine**:

|  |
|--|
| <pre> <b>in</b>    <math>n</math> : <b>integer initially</b> <math>n \geq 2</math> <b>local</b> <math>f</math> : <b>array</b> <math>[1..n]</math> <b>of integer initially</b> <math>f = 1</math>  <math>\prod_{j=1}^n P[j] ::</math>     [ <math>\ell_0</math> : <b>loop forever do</b>       [ <math>\ell_1</math> : <b>Non-Critical</b>         [ <math>\ell_2</math> : <b>request</b> <math>f[j]</math>           [ <math>\ell_3</math> : <b>request</b> <math>f[j \oplus_n 1]</math>             [ <math>\ell_4</math> : <b>Critical</b>               [ <math>\ell_5</math> : <b>release</b> <math>f[j]</math>                 [ <math>\ell_6</math> : <b>release</b> <math>f[j \oplus_n 1]</math>                   ]                 ]               ]             ]           ]         ]       ]     ]           </pre> |
|--|

It is not difficult to verify the following **safety** property

$$\square \neg(at_{\ell_4}[1] \wedge at_{\ell_4}[2]),$$

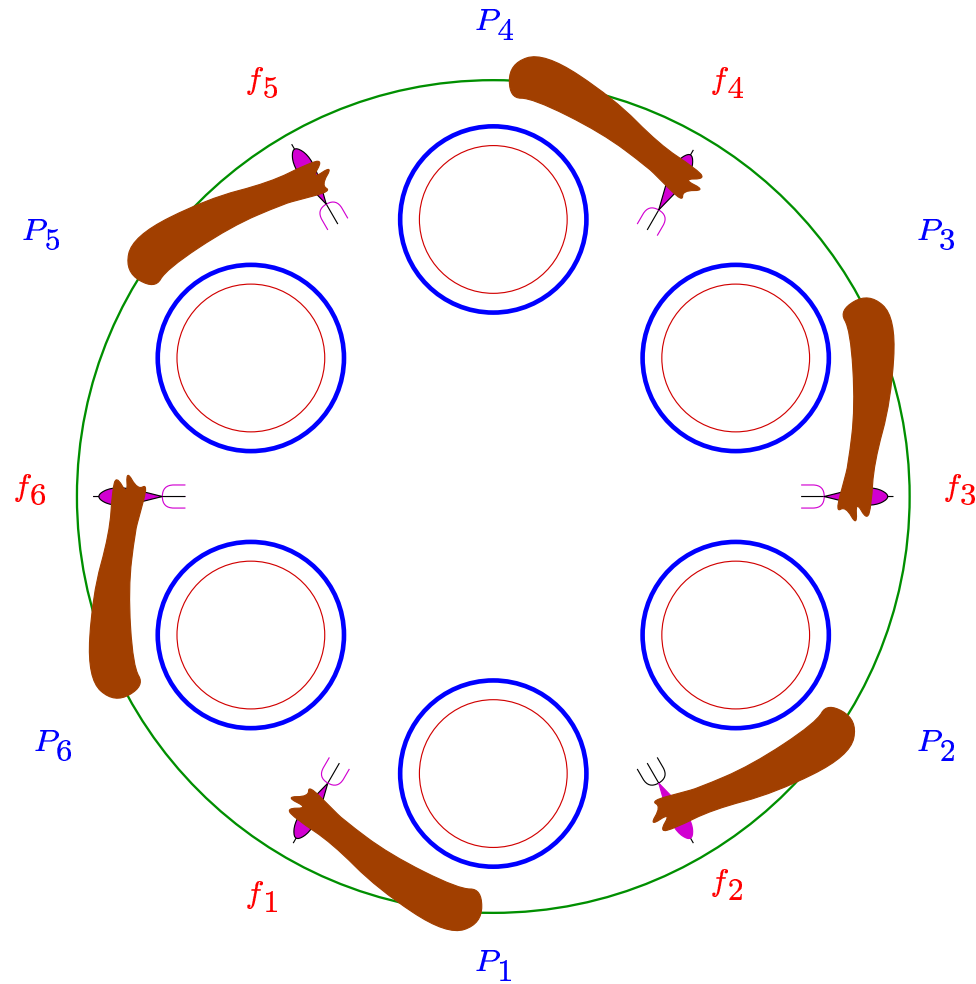
stating that philosophers  $P[1]$  and  $P[2]$  can never eat at the same time.

## Accessibility not Guaranteed

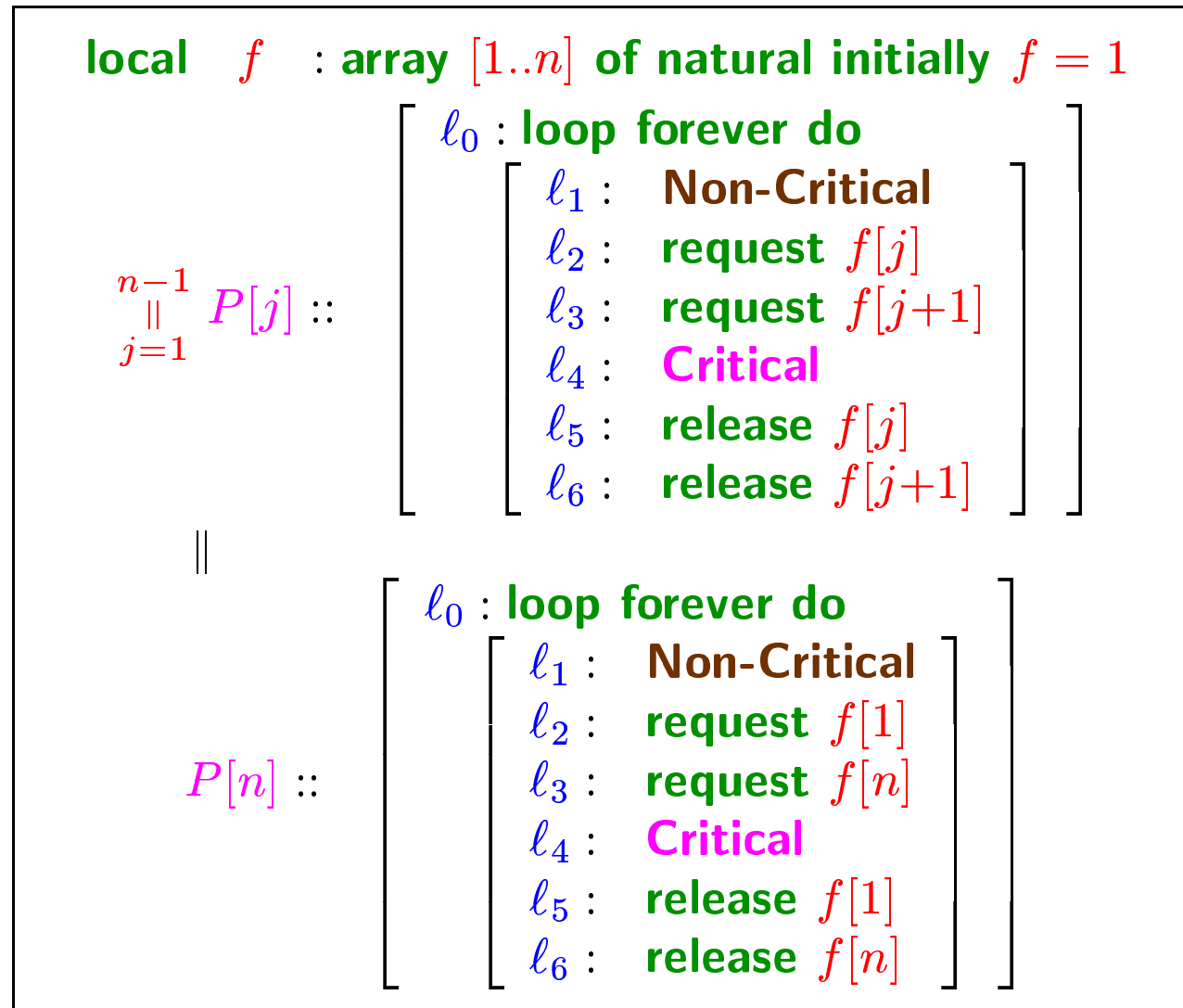
Unfortunately, Dine cannot ensure **accessibility** for  $P[1]$ , specifiable by

$$\square (at\_l_2[1] \rightarrow \diamond at\_l_4[1])$$

Because all philosophers may deadlock together.



## Solution: One Contrary Philosopher



Wish to establish **accessibility**, expressible by

$$\psi_{acc}: \square (at\_l_2[j] \rightarrow \diamond (at\_l_4[j]))$$

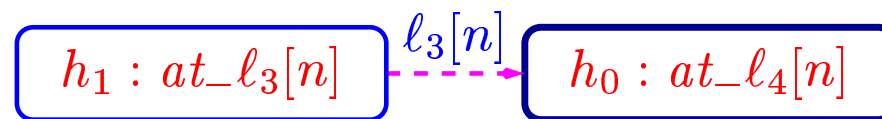
## Prove A Chain of Eventualities

Before proving accessibility for arbitrary  $j$ , we will establish

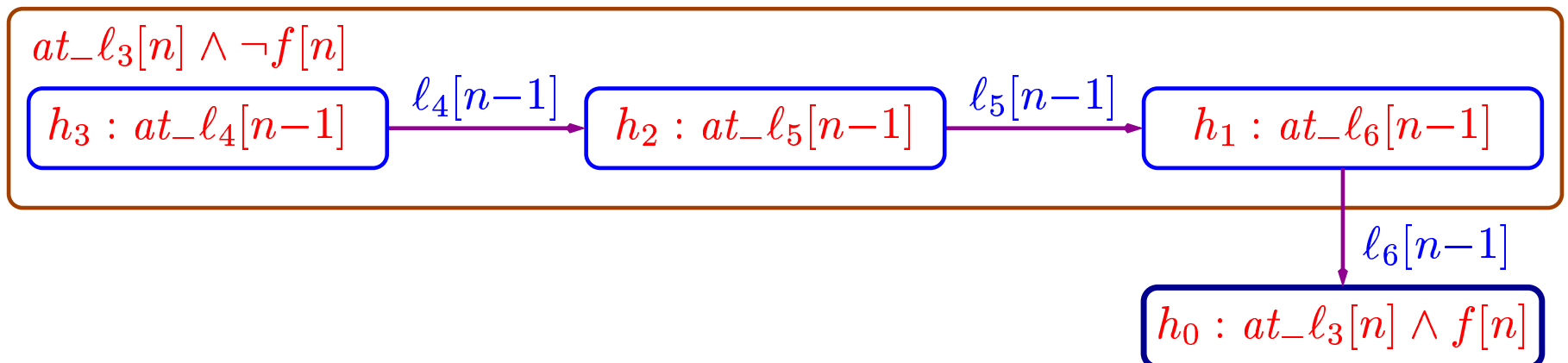
$$A_{3,4}[i] : at\_l_3[i] \Rightarrow \diamond at\_l_4[i]$$

by induction for  $i = n, n - 1, \dots, 1$ .

**Induction Base:**  $A_{3,4}[n] : at\_l_3[n] \Rightarrow \diamond at\_l_4[n]$

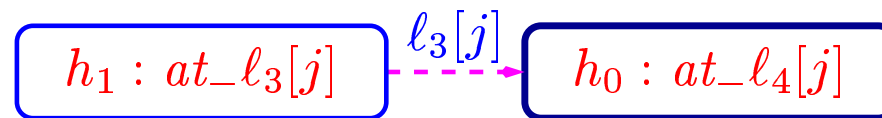


Premise R5 for  $l_3[n]$  requires showing  $at\_l_3[n] \Rightarrow \diamond (at\_l_3[n] \wedge f[n])$ . Using the invariant  $at\_l_{4..6}[n] + at\_l_{4..6}[n-1] + f[n] = 1$ , this can be established by the following verification diagram:



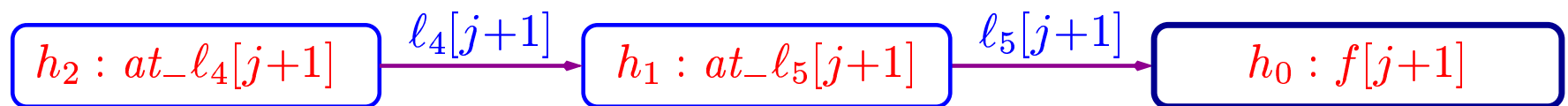
## The Induction Step

We will now show that, assuming  $A_{3,4}[j+1] : at\_l_3[j+1] \Rightarrow \Diamond at\_l_4[j+1]$ , we can establish  $A_{3,4}[j] : at\_l_3[j] \Rightarrow \Diamond at\_l_4[j]$ , for every  $j < n$ . This is established by the following verification diagram:



Premise R5 for  $l_3[j]$  requires showing  $at\_l_3[j] \Rightarrow \Diamond (at\_l_3[j] \wedge f[j+1])$ . Using the invariant  $at\_l_{4..6}[j] + at\_l_{3..5}[j+1] + f[j+1] = 1$ , we construct the following proof:

1.  $at\_l_3[j] \Rightarrow at\_l_3[j+1] \vee at\_l_{4,5}[j+1] \vee f[j+1]$   
According to the invariant
2.  $at\_l_3[j+1] \Rightarrow \Diamond at\_l_4[j+1]$  By induction hypothesis
3.  $at\_l_{4,5}[j+1] \Rightarrow \Diamond f[j+1]$  Verification diagram below
4.  $at\_l_3[j] \Rightarrow \Diamond f[j+1]$  Temporal reasoning on 1–3

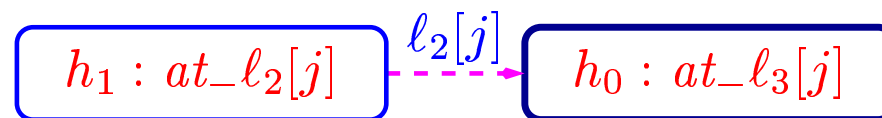


## Verifying Accessibility

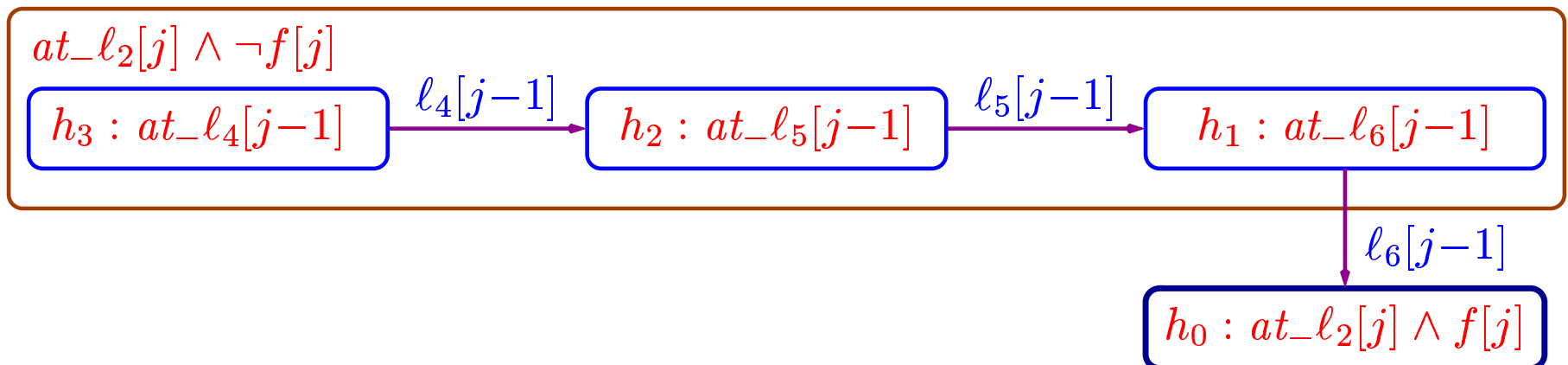
Finally, we verify  $at\_l_2[j] \Rightarrow \diamond at\_l_4[j]$ , for all  $j$ ,  $1 < j < n$ . The proof follows:

1.  $at\_l_2[j] \Rightarrow \diamond at\_l_3[j]$  Verification diagram below
2.  $at\_l_3[j] \Rightarrow \diamond at\_l_4[j]$  Proven by induction
3.  $at\_l_2[j] \Rightarrow \diamond at\_l_4[j]$  Temporal reasoning on 1–2

The verification diagram for  $at\_l_2[j] \Rightarrow \diamond at\_l_3[j]$  is given by:



Premise R5 for  $l_2[j]$  requires showing  $at\_l_2[j] \Rightarrow \diamond (at\_l_2[j] \wedge f[j])$ . Using the invariant  $at\_l_{3..5}[j] + at\_l_{4..6}[j-1] + f[j] = 1$ , this can be established by the following verification diagram:



## A Distributed Rank Justice-Base Rule

In some cases there is no 1–1 correspondence between justice requirements and transitions. In this case, we have to go back to a rule which is based on justice requirements rather than on transitions.

### Rule DISTR-JUST

For a well-founded domain  $(\mathcal{A}, \succ)$

For justice requirements

$$J_1, \dots, J_m,$$

assertions

$$p, q = h_0, h_1, \dots, h_m,$$

and distributed ranking functions

$$\delta_1, \dots, \delta_m : \Sigma \mapsto \mathcal{A}$$

$$\text{D1. } p \Rightarrow \bigvee_{j=0}^m h_j$$

For  $i = 1, \dots, m$

$$\text{D2. } h_i \wedge \rho \Rightarrow h'_i \vee \left( \delta_i \succ \delta'_i \wedge \bigvee_{j=0}^m h'_j \right)$$

$$\text{D3. } h_i \wedge \rho \Rightarrow \bigwedge_{j=1}^m (\delta_j \succeq \delta'_j)$$

$$\text{D4. } h_i \Rightarrow \neg J_i$$

---


$$p \Rightarrow \diamond q$$

## Reducing Compassion to Justice

An alternative approach to the verification of response properties over systems with compassion requirements is based on the reduction of compassion into justice.

Let  $\mathcal{D} : \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$  be an FDS with a non-empty set of compassion requirements. We construct a system  $\mathcal{D}_{\mathcal{J}} : \langle V_{\mathcal{J}}, \Theta_{\mathcal{J}}, \rho_{\mathcal{J}}, \mathcal{J}_{\mathcal{J}}, \emptyset \rangle$  which contains no compassion requirements. Its constituents are given by:

$$V_{\mathcal{J}} : V \cup \{ \text{nevermore}_i : \text{boolean} \mid (p_i, q_i) \in \mathcal{C} \}$$

$$\Theta_{\mathcal{J}} : \Theta \wedge \bigwedge_{(p_i, q_i) \in \mathcal{C}} \neg \text{nevermore}_i$$

$$\rho_{\mathcal{J}} : \rho \vee \bigvee_{(p_i, q_i) \in \mathcal{C}} (\text{nevermore}'_i = 1 \wedge \text{pres}(V - \{ \text{nevermore}_i \}))$$

$$\mathcal{J}_{\mathcal{J}} : \mathcal{J} \cup \{ \text{nevermore}_i \vee q_i \mid (p_i, q_i) \in \mathcal{C} \}$$

$$\mathcal{C}_{\mathcal{J}} : \emptyset$$

Then, we can use the following reduction:

In order to prove  $\mathcal{D} \models \varphi \Rightarrow \diamond \psi$ , it is sufficient to prove

$$\mathcal{D}_{\mathcal{J}} \models \varphi \wedge \neg \text{misprediction} \Rightarrow \diamond (\psi \vee \text{misprediction}),$$

where,  $\text{misprediction} = \bigvee_{(p_i, q_i) \in \mathcal{C}} p_i \wedge \text{nevermore}_i$

## Justification of the Reduction

The reduction is based on the observation that a state-sequence  $\sigma$  satisfies the compassion requirement  $(p_i, q_i)$  if either  $\sigma$  contains only finitely many  $p_i$ -states, or it contains infinitely many  $q_i$ -states.

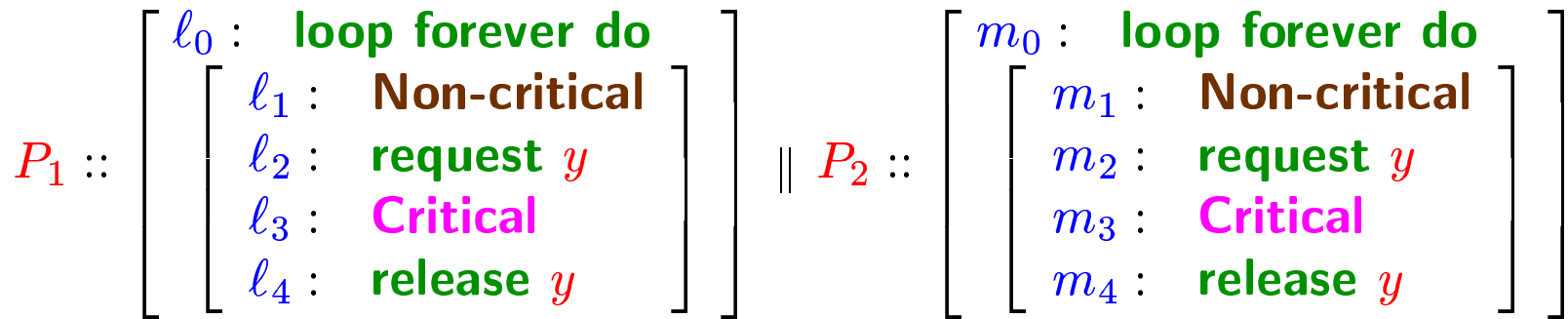
The boolean variable  $nevermore_i$  is intended to be set to **1** at a point, beyond which, there will be no further  $p_i$ -states. Thus,  $nevermore_i$  predicts the absence of  $p_i$ -states. If this prediction is correct, then the newly introduced justice requirement  $nevermore_i \vee q_i$  is equivalent to the original compassion requirement.

In the revised FDS  $\mathcal{D}_J$ , the prediction by  $nevermore_i$  is implemented as a non-deterministic assignment of **1** to  $nevermore_i$ . Therefore, the correctness of the prediction cannot be guaranteed.

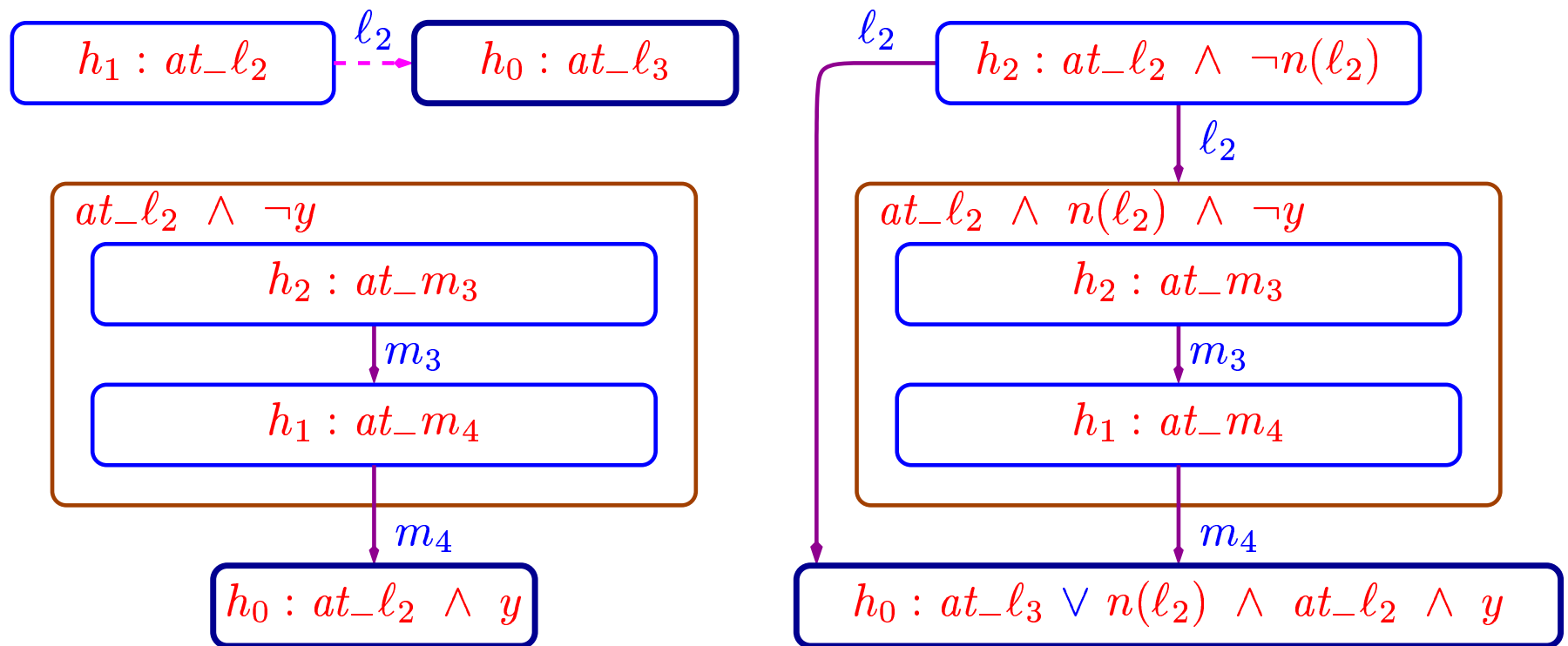
To counter this difficulty, we modify the response property which we aim to prove. The revised property claims that any  $\varphi$ -state in which no mis-prediction has been detected yet, must be followed by a goal state which, either satisfies  $\psi$ , or detects a mis-prediction. **Mis-prediction** is identified as a state in which  $nevermore_i$  and  $p_i$  are both true.

# Comparing General Rule RESP to the *nevermore* Reduction

$y$ : natural initially  $y = 1$



Following are verification diagrams for the two approaches:



## Example: MUX-SEM

Reconsider program MUX-SEM:

|   |  |
|---|--|
| <pre> <b>in</b>    <math>N</math> : <b>integer where</b> <math>N &gt; 1</math> <b>local</b> <math>y</math> : <math>\{0, 1\}</math> <b>where</b> <math>y = 1</math>  <math>\prod_{p=1}^N P[p] ::</math> </pre> | $\left[ \begin{array}{l} \ell_0 : \text{loop forever do} \\ \left[ \begin{array}{l} \ell_1 : \text{noncritical} \\ \ell_2 : \text{request } y \\ \ell_3 : \text{critical} \\ \ell_4 : \text{release } y \end{array} \right] \end{array} \right]$ |
|---|--|

For which we wish to prove the response property

$$at\_l_2[z] \Rightarrow \diamond at\_l_3[z]$$

We start by establishing the following invariants:

$$\varphi_1 : \forall i : at\_l_{3,4}[i] \rightarrow y = 0$$

$$\varphi_2 : \forall i \neq j : at\_l_{0..2}[i] \vee at\_l_{0..2}[j]$$

$$\varphi_3 : y = 0 \rightarrow \exists i : at\_l_{3,4}[i]$$

— — Mutual Exclusion

## MUX-SEM Continued

Applying the **compassion**→**justice** reduction, we introduce the boolean variables  $n[i]$ ,  $i = 1, \dots, N$  (abbreviations for *nevermore*[ $i$ ]). The added justice requirements are  $J_2[i] : n[i] \vee \neg at\_l_2[i]$ . The **mis-prediction** predicate is given by:

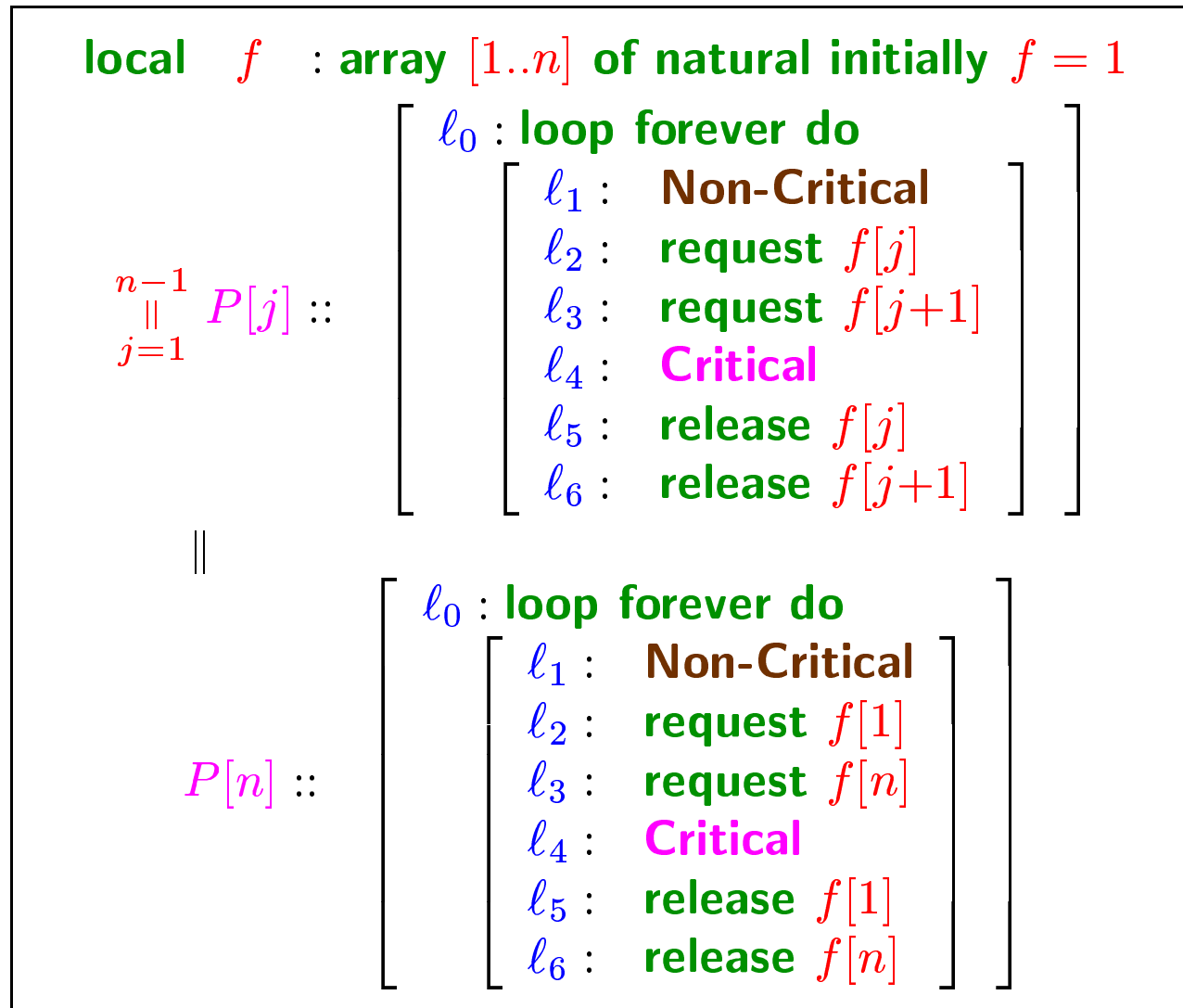
$$misprediction : \bigvee_{i=1}^N at\_l_2[i] \wedge y \wedge n[i]$$

The helpful justice requirements for this proof are  $J_2[z]$  and  $\{J_{3,4}[i] \mid i \in [1..N]\}$ . The helpful conditions and ranking functions for these transitions are given in the following table:

| Id. $p$  | Requirement                 | $h(p)$                                     | $\delta(p)$                     |
|----------|-----------------------------|--|---------------------------------|
| $J_2[z]$ | $n[z] \vee \neg at\_l_2[z]$ | $at\_l_2[z] \wedge \neg n[z]$              | $\neg n[z]$                     |
| $J_3[i]$ | $\neg at\_l_3[i]$           | $at\_l_2[z] \wedge n[z] \wedge at\_l_3[i]$ | $\neg n[z] \vee at\_l_3[i]$     |
| $J_4[i]$ | $\neg at\_l_4[i]$           | $at\_l_2[z] \wedge n[z] \wedge at\_l_4[i]$ | $\neg n[z] \vee at\_l_{3,4}[i]$ |

The ranking functions range over the domain  $\{0, 1\}$ . The assertion  $\delta(p)$  is true at a state if the corresponding ranking of  $J(p)$  is 1. Usually, this is the case if requirement  $p$  may still become helpful. If  $\delta(p)$  is false, then the corresponding ranking is 0.

# Example: Dining Philosophers with One Contrary Philosopher



We wish to establish part of **accessibility**, expressible by

$$\psi_{acc}: \quad \square (at\_l_3[z] \rightarrow \diamond (at\_l_4[z]))$$

for  $z \in [2..N - 1]$ .

## Dining Philosophers Continued

Applying the **compassion**→**justice** reduction, we introduce two arrays of *nevermore* variables,  $n_2[i]$  and  $n_3[i]$  corresponding to locations  $\ell_2[i]$  and  $\ell_3[i]$ .

The helpful justice requirements are  $J_{3..6}[z-1]$ ,  $J_{2,3}[z]$ ,  $\{J_{3..5}[i] \mid i \in [z+1..N-1]\}$  and  $J_{4..6}[N]$ . The helpful conditions for these transitions are given in the following table:

| Id. $p$                     | $h(p)$  |
|-----------------------------|---|
| $J_4[z-1]$                  | $at\_l_4[z-1] \wedge at\_l_2[z] \wedge n_2[z]$  |
| $J_5[z-1]$                  | $at\_l_5[z-1] \wedge at\_l_2[z] \wedge n_2[z]$  |
| $J_6[z-1]$                  | $at\_l_6[z-1] \wedge at\_l_2[z] \wedge n_2[z]$  |
| $J_2[z]$                    | $at\_l_2[z] \wedge \neg n_2[z]$   |
| $J_3[z]$                    | $at\_l_3[z] \wedge \neg n_3[z]$   |
| $J_3[i] : i \in [z+1..N-1]$ | $at\_l_3[z] \wedge at\_l_3[i] \wedge \neg n_3[i] \wedge at\_l_3[i-1] \wedge n_3[i-1]$ |
| $J_4[i] : i \in [z+1..N-1]$ | $at\_l_3[z] \wedge at\_l_4[i] \wedge at\_l_3[i-1] \wedge n_3[i-1]$                    |
| $J_5[i] : i \in [z+1..N-1]$ | $at\_l_3[z] \wedge at\_l_5[i] \wedge at\_l_3[i-1] \wedge n_3[i-1]$                    |
| $J_4[N]$                    | $at\_l_3[z] \wedge at\_l_4[N] \wedge at\_l_3[N-1] \wedge n_3[N-1]$                    |
| $J_5[N]$                    | $at\_l_3[z] \wedge at\_l_5[N] \wedge at\_l_3[N-1] \wedge n_3[N-1]$                    |
| $J_6[N]$                    | $at\_l_3[z] \wedge at\_l_6[N] \wedge at\_l_3[N-1] \wedge n_3[N-1]$                    |

## Dining Philosophers: Ranking Functions

The following table presents the distributed ranking functions  $\delta(p)$  for each of the helpful requirements  $J(p)$ . The ranking functions range over  $\{0, 1\}$ , and the assertion  $\delta(p)$  tells us when the ranking of  $J(p)$  is 1.

|                                  |   |
|----------------------------------|---|
| $\delta_4[z-1]$                  | $n_2[z] \rightarrow at_{\ell_{0..4}}[z-1]$  |
| $\delta_5[z-1]$                  | $n_2[z] \rightarrow at_{\ell_{0..5}}[z-1]$  |
| $\delta_6[z-1]$                  | 1   |
| $\delta_2[z]$                    | $at_{\ell_2}[z] \wedge \neg n_2[z]$   |
| $\delta_3[z]$                    | $\neg n_3[z]$   |
| $\delta_3[i] : i \in [z+1..N-1]$ | $\neg n_3[i] \wedge (at_{\ell_3}[i-1] \wedge n_3[i-1] \rightarrow at_{\ell_{0..3,6}}[i])$ |
| $\delta_4[i] : i \in [z+1..N-1]$ | $at_{\ell_3}[i-1] \wedge n_3[i-1] \rightarrow at_{\ell_{0..4,6}}[i]$                      |
| $\delta_5[i] : i \in [z+1..N-1]$ | 1   |
| $\delta_4[N]$                    | $at_{\ell_3}[N-1] \wedge n_3[N-1] \rightarrow at_{\ell_{0..4}}[N]$                        |
| $\delta_5[N]$                    | $at_{\ell_3}[N-1] \wedge n_3[N-1] \rightarrow at_{\ell_{0..5}}[N]$                        |
| $\delta_6[N]$                    | 1   |

**Assignment 1.** Draw a verification diagram for the proof of accessibility for the *dining-philosophers* system.

## The Centralized vs. Distributed Versions of Rule Well

The premises of the **centralized** version are:

$$D1. \quad p \rightarrow \bigvee_{j=0}^m h_j$$

$$D2. \quad h_i \wedge \rho \rightarrow (h'_i \wedge \delta_i = \delta'_i \wedge \neg J'_i) \vee \left( \bigvee_{j=0}^m h'_j \wedge \delta_i \succ \delta'_j \right)$$

In the **distributed** version, premise **D2** is replaced by:

$$D2. \quad h_i \wedge \rho \rightarrow (h'_i \wedge \neg J'_i) \vee \left( \delta_i > \delta'_i \wedge \bigvee_{j=0}^m h'_j \right)$$

$$D3. \quad h_i \wedge \rho \rightarrow q' \vee \delta_j \geq \delta'_j$$

Thus, in both versions, we have to identify for each requirement  $J_i$ , the helpful assertion  $h_i$  characterizing the states at which progress is guaranteed by satisfaction of  $J_i$ .

The versions differ in the type of the **ranking functions**  $\delta_i$  and the heuristics for their identification.

## Identifying the Ranking Functions

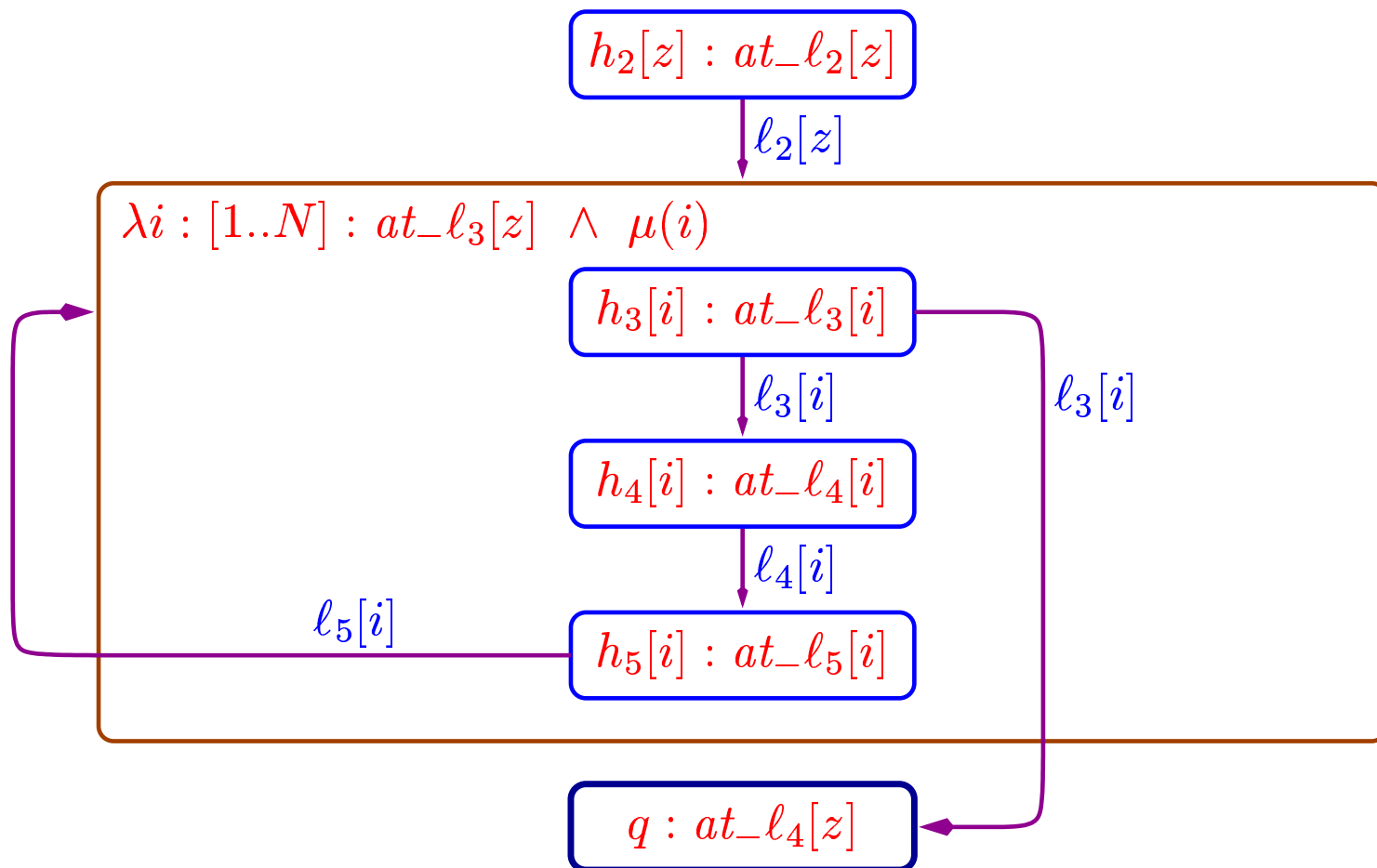
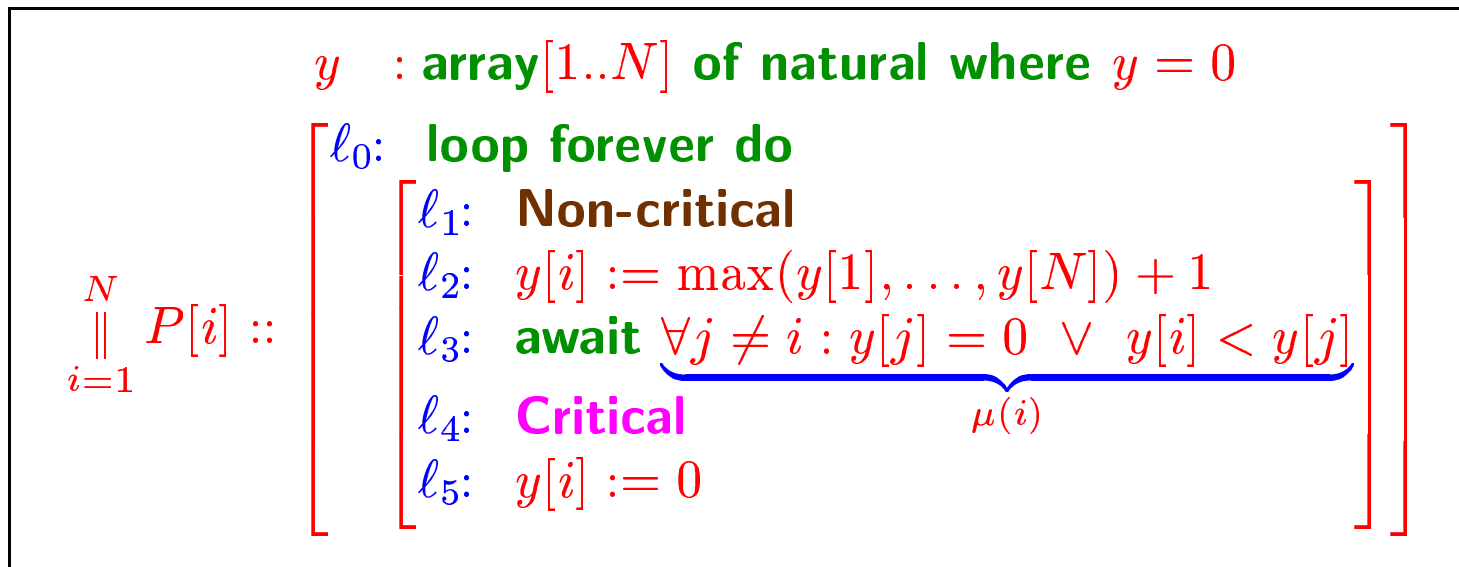
In the **centralized** version, the **ranking functions** are determined to identify global progress. They often are lexicographic tuples of well-founded domains.

For the **distributed** version, the **ranking functions** are often **binary** (range over  $\{0, 1\}$ ). We can represent them by an assertion  $\delta_i$  true whenever  $\delta_i = 1$ . There are essentially two heuristics for determining  $\delta$ .

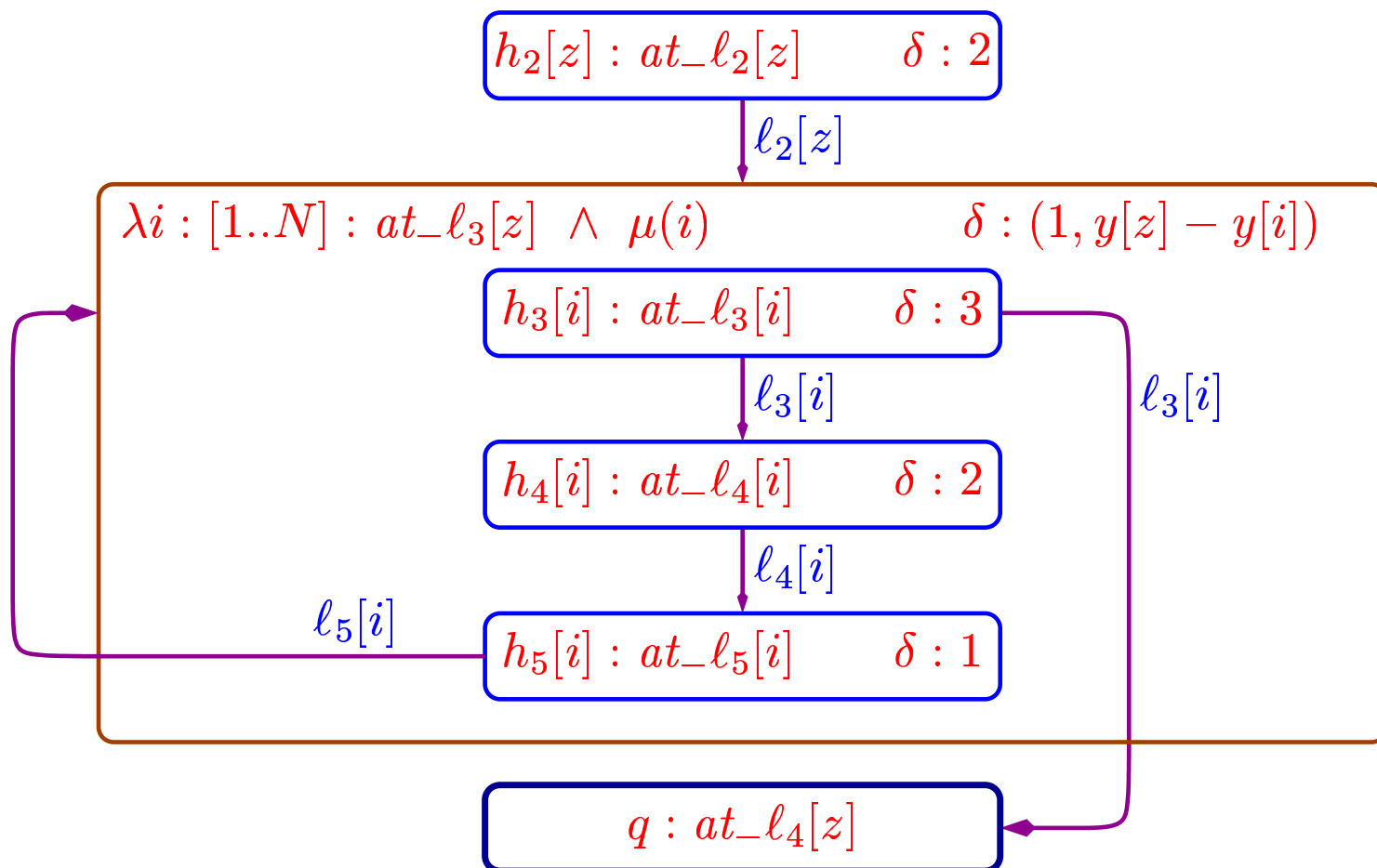
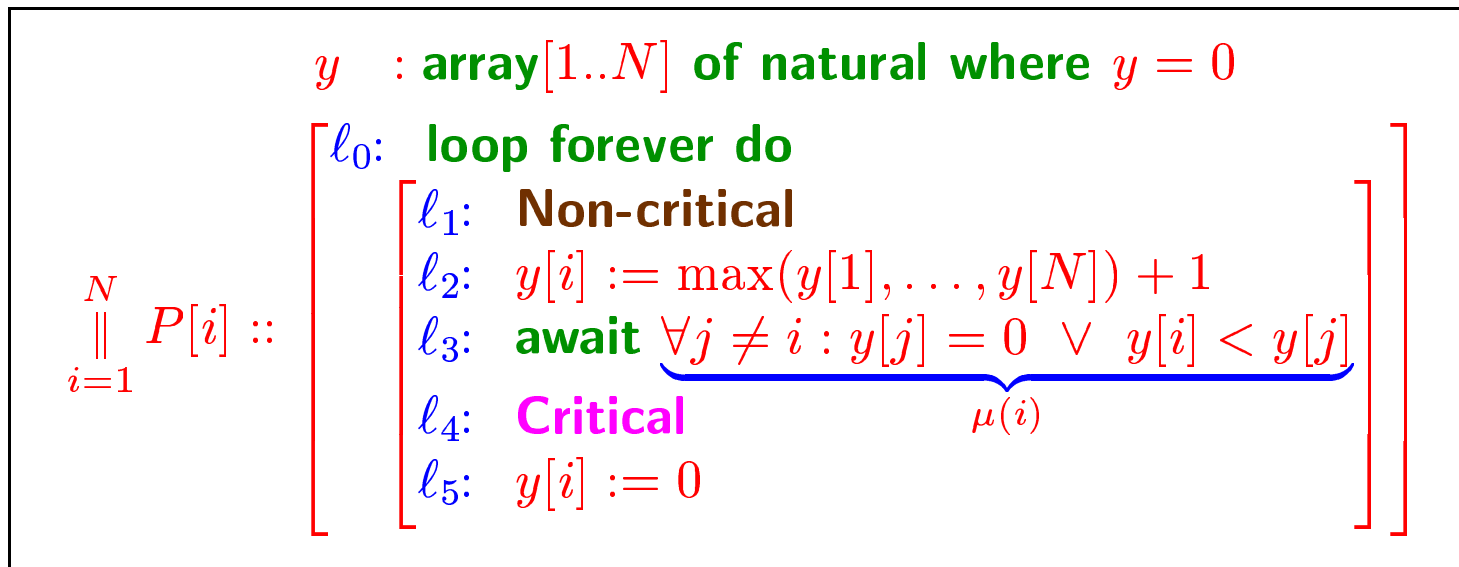
- $\delta_i$  should characterize the states from which  $J_i$  may still become helpful.
- $\neg\delta_i$  should characterize all  $J_i$ -states immediately following an  $h_i$ -state, and their descendants. More generally,  $\neg\delta_i$  should characterize all states at which  $J_i$  is not helpful and can never become helpful in the future.

We will illustrate this on two of our running examples.

# The BAKERY Algorithm



# With Centralized Ranking Functions



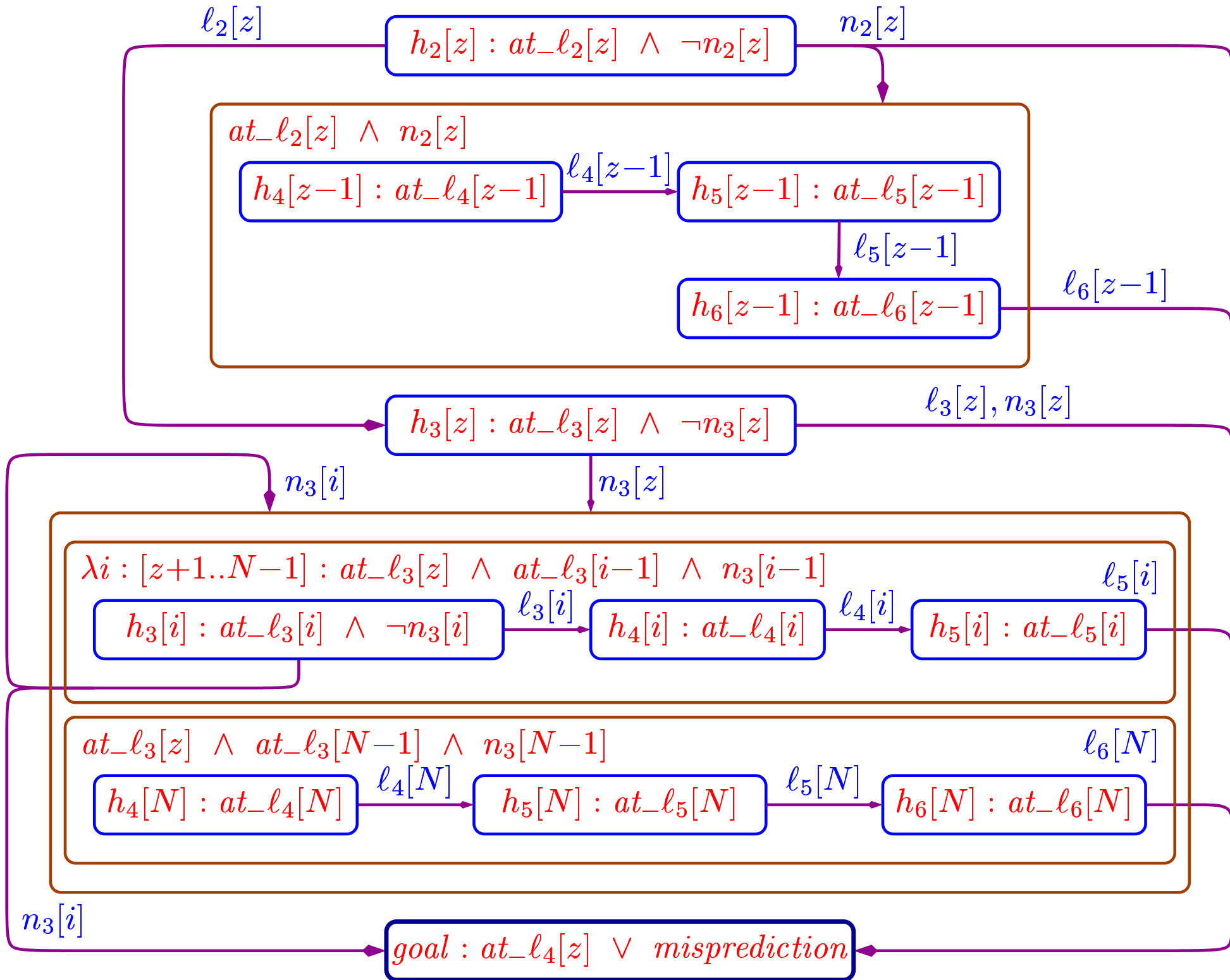
## With Distributed Ranking Functions

|  |  |                      |  |   |  |  |  |                  |  |                     |  |
|--|--|----------------------|--|---|--|--|--|------------------|--|---------------------|--|
|  | $y$ : array[1..N] of natural where $y = 0$   |                      |  |   |  |  |  |                  |  |                     |  |
| $\prod_{i=1}^N P[i] ::$  | $l_0$ : loop forever do <table style="border-left: 1px solid black; border-right: 1px solid black; padding-left: 10px; margin-left: 20px;"> <tr> <td style="padding-right: 10px;"><math>l_1</math>: Non-critical</td> <td></td> </tr> <tr> <td style="padding-right: 10px;"><math>l_2</math>: <math>y[i] := \max(y[1], \dots, y[N]) + 1</math></td> <td></td> </tr> <tr> <td style="padding-right: 10px;"><math>l_3</math>: await <math>\underbrace{\forall j \neq i : y[j] = 0 \vee y[i] &lt; y[j]}_{\mu(i)}</math></td> <td></td> </tr> <tr> <td style="padding-right: 10px;"><math>l_4</math>: Critical</td> <td></td> </tr> <tr> <td style="padding-right: 10px;"><math>l_5</math>: <math>y[i] := 0</math></td> <td></td> </tr> </table> | $l_1$ : Non-critical |  | $l_2$ : $y[i] := \max(y[1], \dots, y[N]) + 1$ |  | $l_3$ : await $\underbrace{\forall j \neq i : y[j] = 0 \vee y[i] < y[j]}_{\mu(i)}$ |  | $l_4$ : Critical |  | $l_5$ : $y[i] := 0$ |  |
| $l_1$ : Non-critical   |  |                      |  |   |  |  |  |                  |  |                     |  |
| $l_2$ : $y[i] := \max(y[1], \dots, y[N]) + 1$                                      |  |                      |  |   |  |  |  |                  |  |                     |  |
| $l_3$ : await $\underbrace{\forall j \neq i : y[j] = 0 \vee y[i] < y[j]}_{\mu(i)}$ |  |                      |  |   |  |  |  |                  |  |                     |  |
| $l_4$ : Critical   |  |                      |  |   |  |  |  |                  |  |                     |  |
| $l_5$ : $y[i] := 0$  |  |                      |  |   |  |  |  |                  |  |                     |  |

The ranking functions are given by:

|                |   |
|----------------|---|
| $\delta_2[z]:$ | $at\_l_2[z]$  |
| $\delta_3[i]:$ | $at\_l_2[z] \vee at\_l_3[z] \wedge y[i] \leq y[z] \wedge at\_l_3[i]$      |
| $\delta_4[i]:$ | $at\_l_2[z] \vee at\_l_3[z] \wedge y[i] \leq y[z] \wedge at\_l_{3,4}[i]$  |
| $\delta_5[i]:$ | $at\_l_2[z] \vee at\_l_3[z] \wedge y[i] \leq y[z] \wedge at\_l_{3..5}[i]$ |

# Verification Diagram for Dining



## Distributed Ranking Functions

The useful heuristic here is to identify  $\neg h_i$ -states following an  $h_i$ -state:

|                                  |   |
|----------------------------------|---|
| $\delta_4[z-1]$                  | $\neg(n_2[z] \wedge at\_l_{5,6}[z-1])$  |
| $\delta_5[z-1]$                  | $\neg(n_2[z] \wedge at\_l_6[z-1])$  |
| $\delta_6[z-1]$                  | 1   |
| $\delta_2[z]$                    | $at\_l_2[z] \wedge \neg n_2[z]$   |
| $\delta_3[z]$                    | $\neg n_3[z]$   |
| $\delta_3[i] : i \in [z+1..N-1]$ | $\neg n_3[i] \wedge \neg(at\_l_3[i-1] \wedge n_3[i-1] \wedge at\_l_{4,5}[i])$ |
| $\delta_4[i] : i \in [z+1..N-1]$ | $\neg(at\_l_3[i-1] \wedge n_3[i-1] \wedge at\_l_5[i])$                        |
| $\delta_5[i] : i \in [z+1..N-1]$ | 1   |
| $\delta_4[N]$                    | $\neg(at\_l_3[N-1] \wedge n_3[N-1] \wedge at\_l_{5,6}[N])$                    |
| $\delta_5[N]$                    | $\neg(at\_l_3[N-1] \wedge n_3[N-1] \wedge at\_l_6[N])$                        |
| $\delta_6[N]$                    | 1   |

A **centralized** ranking function can be obtained by counting how many requirements  $J_i$  currently satisfy  $\delta_i$ .