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 cam be specified by the response formula

$$at\_\ell_0 \Rightarrow \diamondsuit\ at\_\ell_2$$

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. It's 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

$$a_0 \succ a_1 \succ \cdots$$

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

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

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

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

Claim 6. *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

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

From the definition of $\succ_{lex}$ it follows that the sequence of first pair members satisfies $a_1 \succeq_1 a_2 \succeq_1 \cdots$. Since $\mathcal{A}_1$ is well founded, it follows that there exists some position k such that $a_k = a_{k+1} = \cdots$. Therefore, the sequence $b_k \succ_2 b_{k+1} \succ_2 \cdots$ 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 $t_1, \ldots, t_m$,

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

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

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

For $i = 1, \ldots, m$

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

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

W4. $h_i \quad \Rightarrow \quad En(t_i)$

$p \quad \Rightarrow \quad \Diamond q$

Soundness of Rule WELL

Claim 7. *Rule* WELL *is sound for proving the response property* $p \Rightarrow \diamondsuit q$.

Proof Assume that the premises of rule WELL are valid. Let $\sigma : s_0, s_1, \ldots$ 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 δ_{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 $i_j, i_{j+1}, \ldots$ where, for each $k \geq j$, $i_k > 0$ and $s_k \models h_{i_k}$. Let us denote by $d_j, d_{j+1}, \ldots$ the sequence of values of the corresponding ranking functions at the respective states. By premises W2 and W3, the sequence $d_j \succeq d_{j+1} \succeq \cdots$ is non-increasing. Since this is an infinite non-increasing sequence over a well-founded domain, there must exist an index n, such that $d_n = d_{n+1} = \cdots$, and consequently (due to W2) $i_n = i_{n+1} = \cdots$.

By premise W3, we can have $i_n = i_{n+1} = \cdots = i$ 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, σ 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 exists a position $k \geq j$ satisfying q. ⌟

Application to Program UP-DOWN

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

$$P_1 :: \left[\begin{array}{ll} \ell_0: & \textbf{while } x = 0 \textbf{ do} \\ & [\ell_1: \ y := y + 1] \\ \ell_2: & \textbf{while } y > 0 \textbf{ do} \\ & [\ell_3: \ y := y - 1] \\ \ell_4: & \end{array}\right] \quad \| \quad P_2 :: \left[\begin{array}{ll} m_0: & x := 1 \\ m_1: & \end{array}\right]$$

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

$$at\_\ell_0 \ \wedge \ at\_m_0 \quad \Rightarrow \quad \diamondsuit\,(at\_\ell_4 \ \wedge \ at\_m_1)$$

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

i	t_i	h_i	δ_i
0	$-$	$at\_\ell_4 \ \wedge \ at\_m_1$	$(0,0,0)$
1	ℓ_3	$at\_\ell_3 \ \wedge \ at\_m_1 \ \wedge \ y > 0$	$(0,y,1)$
2	ℓ_2	$at\_\ell_2 \ \wedge \ at\_m_1$	$(0,y,2)$
3	ℓ_1	$at\_\ell_1 \ \wedge \ at\_m_1 \ \wedge \ (x = 1)$	$(2,0,0)$
4	ℓ_0	$at\_\ell_0 \ \wedge \ at\_m_1 \ \wedge \ (x = 1)$	$(1,0,0)$
5	m_0	$at\_\ell_{0,1} \ \wedge \ at\_m_0$	$(3,0,0)$

A Rule with Distributed Ranking

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

Rule DISTR-RANK

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

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

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

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

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

For $i = 1, \ldots, m$

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

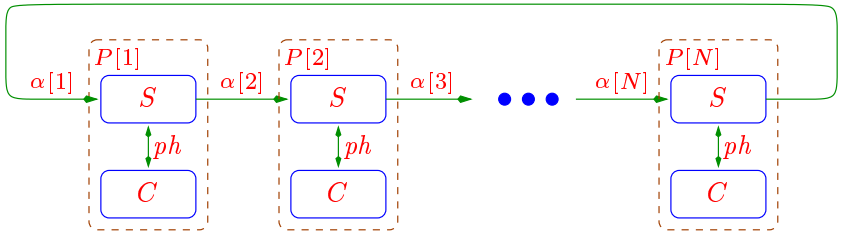
W3. $h_i \wedge \rho_{t_i} \Rightarrow (\bigvee_{j=0}^{m} h'_j) \wedge (\bigvee_{j=1}^{m} (\delta_j \succ \delta'_j))$

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

W5. $h_i \Rightarrow En(t_i)$

$p \Rightarrow \diamondsuit q$

Example: Mutual Exclusion by Token Passing



local α : **array**$[1..N]$ **of boolean where** $\alpha[1] = 1, a[2] = \cdots = a[N] = 0$

$$\|_{i=1}^{N} P[i] :: \left[\begin{array}{l} S :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{ll} \ell_1 : & \textbf{request } \alpha[i] \\ \ell_2 : & \textbf{if } at\_m_2[i] \textbf{ then} \\ & [\ell_3 : \ \textbf{await } at\_m_4[i]] \\ \ell_4 : & \textbf{release } \alpha[i \oplus 1] \end{array} \right] \end{array} \right] \\ \| \\ C :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{ll} m_1 : & \textbf{Non-critical} \\ m_2 : & \textbf{await } at\_\ell_3[i] \\ m_3 : & \textbf{Critical} \\ m_4 : & \textbf{await } \neg at\_\ell_3[i] \end{array} \right] \end{array} \right] \end{array} \right]$$

First Some Invariants

local α : **array**$[1..N]$ **of boolean where** $\alpha[1] = 1, a[2] = \cdots = a[N] = 0$

$$\begin{array}{l} \|_{i=1}^{N} P[i] :: \\ \end{array} \left[\begin{array}{ll} S :: & \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{ll} \ell_1 : & \textbf{request } \alpha[i] \\ \ell_2 : & \textbf{if } at\_m_2[i] \textbf{ then} \\ & [\ell_3 : \quad \textbf{await } at\_m_4[i]] \\ \ell_4 : & \textbf{release } \alpha[i \oplus 1] \end{array} \right] \end{array} \right] \\ \| & \\ C :: & \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{ll} m_1 : & \textbf{Non-critical} \\ m_2 : & \textbf{await } at\_\ell_3[i] \\ m_3 : & \textbf{Critical} \\ m_4 : & \textbf{await } \neg at\_\ell_3[i] \end{array} \right] \end{array} \right] \end{array} \right]$$

The following are invariants of TOKEN-RING:

$$\begin{array}{ll} \varphi_1 : & \sum_{i=1}^{N} (\alpha[i] + at\_\ell_{2..4}[i]) = 1 \\ \varphi_2 : & at\_m_3[i] \rightarrow at\_\ell_3[i]) \end{array}$$

Together they imply mutual exclusion!

Now to Liveness

Accessibility can be specified by

$$at\_m_2[z] \quad \Rightarrow \quad \diamondsuit\, at\_m_3[z]$$

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

We define the cyclic distance between j and z as

$$\Delta(j, z) = (z - j) \; mod \; N$$

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
$\ell_0[i]$	$at\_m_2[z] \ \wedge \ at\_\ell_0[i] \ \wedge \ \alpha[i]$	$(\Delta(i, z), 6)$	$\ell_1[i]$
$\ell_1[i]$	$at\_m_2[z] \ \wedge \ at\_\ell_1[i] \ \wedge \ \alpha[i]$	$(\Delta(i, z), 5)$	$\ell_2[i]$
$\ell_2[i]$	$at\_m_2[z] \ \wedge \ at\_\ell_2[i]$	$(\Delta(i, z), 4)$	$\ell_4[i], m_2[i]$
$m_2[i]$	$at\_m_2[z] \ \wedge \ at\_m_2[i] \ \wedge \ at\_\ell_3[i]$	$(\Delta(i, z), 3)$	$m_3[i], at\_m_3[z]$
$m_3[i]$	$at\_m_2[z] \ \wedge \ at\_m_3[i]$	$(\Delta(i, z), 2)$	$\ell_3[i]$
$\ell_3[i]$	$at\_m_2[z] \ \wedge \ at\_\ell_3[i] \ \wedge \ at\_m_4[i]$	$(\Delta(i, z), 1)$	$\ell_4[i]$
$\ell_4[i]$	$at\_m_2[z] \ \wedge \ at\_\ell_3[i]$	$(\Delta(i \oplus 1, z), 7)$	$\ell_0[i \oplus 1], \ell_1[i \oplus 1]$

The BAKERY Algorithm

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

$$\prod_{i=1}^{N} P[i] :: \left[\begin{array}{ll} \ell_0\text{:} & \textbf{loop forever do} \\ & \left[\begin{array}{ll} \ell_1\text{:} & \textbf{Non-critical} \\ \ell_2\text{:} & y[i] := \max(y[1], \ldots, y[N]) + 1 \\ \ell_3\text{:} & \textbf{await } \forall j \neq i : y[j] = 0 \ \vee \ y[i] < y[j] \\ \ell_4\text{:} & \textbf{Critical} \\ \ell_5\text{:} & y[i] := 0 \end{array}\right] \end{array}\right]$$

Program BAKERY: the Bakery Algorithm.

Some useful invariants:

φ_1 : $y[i] > 0 \iff at\_\ell_{3..5}[i]$

φ_2 : $at\_\ell_{4,5}[i] \rightarrow \underbrace{\forall j \neq i : y[j] = 0 \ \vee \ y[i] < y[j]}_{\mu(i)}$

Together, they imply mutual exclusion.

Verifying Accessibility

Next, let us verify accessibility, specifiable by

$$at\_\ell_2[z] \quad \Rightarrow \quad \Diamond \, at\_\ell_4[z]$$

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	$h(t)$
$\ell_5[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_5[i]$
$\ell_4[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_4[i]$
$\ell_3[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_3[i] \ \wedge \ \mu(i)$
$\ell_2[i]$	$i = z \ \wedge \ at\_\ell_2[i]$

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	$\delta(t) = 2$
$\ell_5[i]$	$at\_\ell_2[z] \ \vee \ at\_\ell_3[z] \ \wedge \ at\_\ell_{3,4}[i] \ \wedge \ y[i] < y[z]$
$\ell_4[i]$	$at\_\ell_2[z] \ \vee \ at\_\ell_3[z] \ \wedge \ at\_\ell_3[i] \ \wedge \ y[i] < y[z]$
$\ell_3[i]$	$at\_\ell_2[z] \ \vee \ at\_\ell_3[z] \ \wedge \ at\_\ell_3[i] \ \wedge \ \neg\mu(i) \ \wedge \ (i = z \ \vee \ y[i] < y[z])$

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

$$at\_\ell_2[z] \quad \Rightarrow \quad \diamondsuit\, at\_\ell_4[z]$$

for the BAKERY algorithm.

Define a ranking function

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

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
$\ell_2[z]$	$at\_\ell_2[z]$	$(1,0,0)$	$\{\ell_{3,4,5}[j]\}$
$\ell_3[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_3[i] \ \wedge \ \mu(i)$	$(0,\Delta,2)$	$\ell_4[i], at\_\ell_4[z]$
$\ell_4[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_4[i]$	$(0,\Delta,1)$	$\ell_5[i]$
$\ell_5[i]$	$at\_\ell_3[z] \ \wedge \ at\_\ell_5[i]$	$(0,\Delta,0)$	$\{\ell_3[j]\}$