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.

\begin{verbatim}
local y natural
\ell_0 : \text{while } y > 0 \text{ do }
\ell_1 : y := y - 1
\ell_2 :
\end{verbatim}

Termination of this program can be specified by the response formula

\[ \text{at}_\ell_0 \Rightarrow \Diamond \text{at}_\ell_2 \]

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.
PossibleDomainsfortheRankingFunction

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 \(\succ\) ordering forms a well-founded domain, denoted \((\mathbb{N}, \succ)\). 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 \((A_1, \succ_1)\) and \((A_2, \succ_2)\), we introduce two ways to construct a composite well-founded domain.

The cross product \(A_1 \times A_2\) is the well-founded domain \((A, \succ)\), where \(A = A_1 \times A_2\) and

\[(a_1, a_2) \succ (b_1, b_2) \iff (a_1 \succ_1 b_1 \land a_2 \succeq_2 b_2) \lor (a_1 \succeq_1 b_1 \land a_2 \succ_2 b_2)\]

The lexicographic product \(A_1 \times_{lex} A_2\) is the well-founded domain \((A, \succ)\), where \(A = A_1 \times A_2\) and

\[(a_1, a_2) \succ_{lex} (b_1, b_2) \iff (a_1 \succ_1 b_1) \lor (a_1 = b_1 \land a_2 \succ_2 b_2)\]

Claim 6. If both \((A_1, \succ_1)\) and \((A_2, \succ_2)\) are well-founded, then so are \(A_1 \times A_2\) and \(A_1 \times_{lex} A_2\).

Proof. It is sufficient to show that \(A_1 \times_{lex} A_2\) 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 \(A_1\) 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 \(A_2\).
**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. \(p \Rightarrow \bigvee_{j=0}^{m} h_j\)

For \(i = 1, \ldots, m\)

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

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

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

\(p \Rightarrow \Diamond q\)
Soundness of Rule \textsc{Well}

\textbf{Claim 7.} \textit{Rule \textsc{Well} is sound for proving the response property }p \Rightarrow \Diamond q.\textit{ }

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

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

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

We conclude that there must exists a position }k \geq j \textit{ satisfying }q. \qed
Application to Program UP-DOWN

\( x, y : \text{natural initially } x = y = 0 \)

\[
P_1 :: \begin{align*}
\ell_0 : & \ 	ext{while } x = 0 \text{ do} \\
[\ell_1 : & \ y := y + 1] \\
\ell_2 : & \ 	ext{while } y > 0 \text{ do} \\
[\ell_3 : & \ y := y - 1] \\
\ell_4 : & 
\end{align*}
\]

\[
P_2 :: \begin{align*}
\ell_0 : & \ x := 1 \\
m_1 : & 
\end{align*}
\]

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

\[
\text{at}_\ell_0 \land \text{at}_m_0 \implies \Diamond (\text{at}_\ell_4 \land \text{at}_m_1)
\]

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

<table>
<thead>
<tr>
<th>i</th>
<th>( t_i )</th>
<th>( h_i )</th>
<th>( \delta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \ell_4 )</td>
<td>( \text{at}_\ell_4 \land \text{at}_m_1 )</td>
<td>( (0, 0, 0) )</td>
</tr>
<tr>
<td>1</td>
<td>( \ell_3 )</td>
<td>( \text{at}_\ell_3 \land \text{at}_m_1 \land y &gt; 0 )</td>
<td>( (0, y, 1) )</td>
</tr>
<tr>
<td>2</td>
<td>( \ell_2 )</td>
<td>( \text{at}_\ell_2 \land \text{at}_m_1 )</td>
<td>( (0, y, 2) )</td>
</tr>
<tr>
<td>3</td>
<td>( \ell_1 )</td>
<td>( \text{at}_\ell_1 \land \text{at}_m_1 \land (x = 1) )</td>
<td>( (2, 0, 0) )</td>
</tr>
<tr>
<td>4</td>
<td>( \ell_0 )</td>
<td>( \text{at}_\ell_0 \land \text{at}_m_1 \land (x = 1) )</td>
<td>( (1, 0, 0) )</td>
</tr>
<tr>
<td>5</td>
<td>( m_0 )</td>
<td>( \text{at}<em>\ell</em>{0,1} \land \text{at}_m_0 )</td>
<td>( (3, 0, 0) )</td>
</tr>
</tbody>
</table>
A Rule with Distributed Ranking

In many cases of parameterized systems $P_1 \parallel \cdots \parallel P_n$, 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 \textbf{DISTR-RANK}.

\begin{center}
\textbf{Rule DISTR-RANK}
For a well-founded domain $(A, \succ)$
For just statements $t_1, \ldots, t_m$,
assertions $p, q = h_0, h_1, \ldots, h_m$,
and ranking functions $\delta_1, \ldots, \delta_m : \Sigma \mapsto A$

\begin{align*}
W1. \quad p & \Rightarrow \bigvee_{j=0}^{m} h_j \\
\text{For } i = 1, \ldots, m \quad & \\
W2. \quad h_i \land \rho_t & \Rightarrow h_i' \lor \left( \bigvee_{j=0}^{m} h_j' \land \bigvee_{j=1}^{m} (\delta_j \succ \delta_j') \right) \\
\text{For every } t \neq t_i \quad & \\
W3. \quad h_i \land \rho_{t_i} & \Rightarrow \left( \bigvee_{j=0}^{m} h_j' \land \bigvee_{j=1}^{m} (\delta_j \succ \delta_j') \right) \\
W4. \quad h_i \land \rho & \Rightarrow \bigwedge_{j=1}^{m} (\delta_j \succeq \delta_j') \\
W5. \quad h_i & \Rightarrow En(t_i) \\
\end{align*}

$p \Rightarrow \lozenge q$
\end{center}
Example: Mutual Exclusion by Token Passing

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

$S ::$

$\ell_0 : \text{loop forever do}$

$\ell_1 : \text{request } \alpha[i]$

$\ell_2 : \text{if } \text{at}_m2[i] \text{ then}$

$[\ell_3 : \text{await } \text{at}_m4[i]]$

$\ell_4 : \text{release } \alpha[i \oplus 1]$

$C ::$

$m_0 : \text{loop forever do}$

$m_1 : \text{Non-critical}$

$m_2 : \text{await } \text{at}_m3[i]$

$m_3 : \text{Critical}$

$m_4 : \text{await } \neg \text{at}_m3[i]$
First Some Invariants

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

\[
S :: \begin{cases}
\ell_0 : \text{ loop forever do} \\
\ell_1 : \text{ request } \alpha[i] \\
\ell_2 : \text{ if } at_m2[i] \text{ then} \\
\quad \ell_3 : \text{ await } at_m4[i] \\
\ell_4 : \text{ release } \alpha[i \oplus 1]
\end{cases}
\]

\[
P[i] :: \begin{cases}
N \parallel \\
i=1
\end{cases}
\]

\[
C :: \begin{cases}
\ell_0 : \text{ loop forever do} \\
m_0 : \text{ Non-critical} \\
m_1 : \text{ await } at_\ell3[i] \\
m_2 : \text{ Critical} \\
m_3 : \text{ await } at_\ell3[i] \\
m_4 : \text{ Non-critical}
\end{cases}
\]

The following are invariants of \text{TOKEN-RING}:

\[
\varphi_1 : \sum_{i=1}^N (\alpha[i] + at_\ell2..4[i]) = 1
\]
\[
\varphi_2 : at_m3[i] \rightarrow at_\ell3[i]
\]

Together they imply \text{mutual exclusion!}
Now to Liveness

Accessibility can be specified by

\[ \text{at}_m[z] \Rightarrow \Diamond \text{at}_n[z] \]

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:

<table>
<thead>
<tr>
<th>Trans.</th>
<th>( h(t) )</th>
<th>( \delta(t) )</th>
<th>Successors</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_0[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_e[l][i] \land \alpha[i] )</td>
<td>( \Delta(i, z), 6 )</td>
<td>( \ell_1[i] )</td>
</tr>
<tr>
<td>( \ell_1[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_e[l][i] \land \alpha[i] )</td>
<td>( \Delta(i, z), 5 )</td>
<td>( \ell_2[i] )</td>
</tr>
<tr>
<td>( \ell_2[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_e[l][i] )</td>
<td>( \Delta(i, z), 4 )</td>
<td>( \ell_4[i], m_2[i] )</td>
</tr>
<tr>
<td>( m_2[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_m_2[i] \land \text{at}_e[l][i] )</td>
<td>( \Delta(i, z), 3 )</td>
<td>( m_3[i], \text{at}_m_3[z] )</td>
</tr>
<tr>
<td>( m_3[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_m_2[i] \land \text{at}_e[l][i] )</td>
<td>( \Delta(i, z), 2 )</td>
<td>( \ell_3[i] )</td>
</tr>
<tr>
<td>( \ell_3[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_e[l][i] \land \text{at}_m_4[i] )</td>
<td>( \Delta(i, z), 1 )</td>
<td>( \ell_4[i] )</td>
</tr>
<tr>
<td>( \ell_4[i] )</td>
<td>( \text{at}_m[z] \land \text{at}_e[l][i] )</td>
<td>( \Delta(i \oplus 1, z), 7 )</td>
<td>( \ell_0[i \oplus 1], \ell_1[i \oplus 1] )</td>
</tr>
</tbody>
</table>
The **BAKERY** Algorithm

\[
\begin{align*}
N & : \text{natural where } N > 0 \\
y & : \text{array}[1..N] \text{ of natural where } y = 0 \\
\end{align*}
\]

\[
P[i] :: \begin{cases}
\ell_0: \text{ loop forever do} \\
\quad \ell_1: \text{ Non-critical} \\
\quad \ell_2: y[i] := \max(y[1], \ldots, y[N]) + 1 \\
\quad \ell_3: \text{ await } \forall j \neq i : y[j] = 0 \lor y[i] < y[j] \\
\quad \ell_4: \text{ Critical} \\
\quad \ell_5: y[i] := 0 \\
\end{cases}
\]

Program **BAKERY**: the Bakery Algorithm.

Some useful invariants:

\[
\begin{align*}
\varphi_1 : & \ y[i] > 0 \iff \text{at } \ell_3..5[i] \\
\varphi_2 : & \ \text{at } \ell_4,5[i] \to \forall j \neq i : y[j] = 0 \lor y[i] < y[j] \\
\end{align*}
\]

Together, they imply mutual exclusion.
Verifying Accessibility

Next, let us verify accessibility, specifiable by

\[ at_\ell_2[z] \Rightarrow \Diamond at_\ell_4[z] \]

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

\[
\begin{align*}
\delta(t) &= 1 & \text{If } t \text{ is currently enabled. This is also the case that } t \text{ is helpful.} \\
\delta(t) &= 2 & \text{If } t \text{ is currently disabled, but may become helpful on the way from } p \text{ to } q. \\
\delta(t) &= 0 & \text{If } t \text{ is disabled and can never become helpful before } q \text{ is achieved.}
\end{align*}
\]

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

<table>
<thead>
<tr>
<th>( t )</th>
<th>( h(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_5[i] )</td>
<td>( at_\ell_3[z] \land at_\ell_5[i] )</td>
</tr>
<tr>
<td>( \ell_4[i] )</td>
<td>( at_\ell_3[z] \land at_\ell_4[i] )</td>
</tr>
<tr>
<td>( \ell_3[i] )</td>
<td>( at_\ell_3[z] \land at_\ell_3[i] \land \mu(i) )</td>
</tr>
<tr>
<td>( \ell_2[i] )</td>
<td>( i = z \land at_\ell_2[i] )</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>( t )</th>
<th>( \delta(t) = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_5[i] )</td>
<td>( at_\ell_2[z] \lor at_\ell_3[z] \land at_\ell_{3,4}[i] \land y[i] &lt; y[z] )</td>
</tr>
<tr>
<td>( \ell_4[i] )</td>
<td>( at_\ell_2[z] \lor at_\ell_3[z] \land at_\ell_3[i] \land y[i] &lt; y[z] )</td>
</tr>
<tr>
<td>( \ell_3[i] )</td>
<td>( at_\ell_2[z] \lor at_\ell_3[z] \land at_\ell_3[i] \land \neg\mu(i) \land (i = z \lor y[i] &lt; y[z]) )</td>
</tr>
</tbody>
</table>

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

\[ \text{at}_2[z] \Rightarrow \Diamond \text{at}_4[z] \]

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:

<table>
<thead>
<tr>
<th>( t )</th>
<th>( h(t) )</th>
<th>( \delta(t) )</th>
<th>Successors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2z</td>
<td>at_2[z]</td>
<td>(1, 0, 0)</td>
<td>( { l_{3,4,5}[j] } )</td>
</tr>
<tr>
<td>3i</td>
<td>at_3[z] \land at_3[i] \land \mu(i)</td>
<td>(0, ( \Delta ), 2)</td>
<td>( l_4[i], at_4[z] )</td>
</tr>
<tr>
<td>4i</td>
<td>at_3[z] \land at_4[i]</td>
<td>(0, ( \Delta ), 1)</td>
<td>( l_5[i] )</td>
</tr>
<tr>
<td>5i</td>
<td>at_3[z] \land at_5[i]</td>
<td>(0, ( \Delta ), 0)</td>
<td>( { l_3[j] } )</td>
</tr>
</tbody>
</table>