On Ackermann function and inference rules (η) and $(\mathbf{ap_v})$

$$\begin{aligned}\mathrm{Ack}(0,n) &= n+1\\ \mathrm{Ack}(m+1,0) &= \mathrm{Ack}(m,1)\\ \mathrm{Ack}(m+1,n+1) &= \mathrm{Ack}(m,\mathrm{Ack}(m+1,n))\end{aligned}$$

Remark that this is defined by the lexicographic order of (m,n).

1 First term representation: `ack1`

We first give the first representation `ack1` of Ackermann function. In this article, we sometimes write $f(x,y)$ instead of $f(x)(y)$, for readability.

Definition 1.1 (`ack1`) We define `ack1` by the following equation.

$$\texttt{ack1} = \lambda \mathbf{m}\mathbf{N}.\mathrm{cond}(\texttt{S}\mathbf{N}, \lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(\mathbf{N}))(\mathbf{m})$$

Checking the behavior of `ack1`

$$\begin{aligned}\texttt{ack1}(0,n) &\mapsto \mathrm{cond}(\texttt{S}n, \lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(n))(0)\\ &\mapsto \texttt{S}n\end{aligned}$$

$$\begin{aligned}\texttt{ack1}(\texttt{S}m,0) &\mapsto \mathrm{cond}(\texttt{S}0, \lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(0))(\texttt{S}m)\\ &\mapsto (\lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(0))(m)\\ &\mapsto \mathrm{cond}(\texttt{ack1}(m,1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(0)\\ &\mapsto \texttt{ack1}(m,1)\end{aligned}$$

$$\begin{aligned}\texttt{ack1}(\underline{\texttt{S}m},\texttt{S}n) &\mapsto \mathrm{cond}(\texttt{SS}n, \lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(\texttt{S}n))(\underline{\texttt{S}m})\\ &\mapsto (\lambda m'.\mathrm{cond}(\texttt{ack1}(m',1), \lambda n'.\texttt{ack1}(m', \texttt{ack1}(\texttt{S}m',n')))(\texttt{S}n))(\underline{m})\\ &\mapsto \mathrm{cond}(\texttt{ack1}(m,1), \lambda n'.\texttt{ack1}(m, \texttt{ack1}(\texttt{S}\underline{m},n')))(\texttt{S}n)\\ &\mapsto (\lambda n'.\texttt{ack1}(m, \texttt{ack1}(\texttt{S}\underline{m},n')))(n)\\ &\mapsto \texttt{ack1}(m, \texttt{ack1}(\texttt{S}\underline{m},n))\end{aligned}$$

The point of `ack1` is the last case. The term $\underline{\texttt{S}m}$ of the first line and $\texttt{S}\underline{m}$ of the last line are slightly diffelent: $\underline{\texttt{S}m}$ is decomposed into $\underline{m}$ by the cond-reduction, then $\texttt{S}\underline{m}$ is constructed by substituting m' of $\texttt{S}m'$ by $\underline{m}$.

1.1 $\texttt{ack1} : N \to N \to N$ is not in GTC (with (η))

Claim $\texttt{ack1} : N \to N \to N$ is well-typed, but does not satisfy GTC in the system with (η).

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$n : N \vdash n : N$}
\UnaryInfC{$n : N \vdash \mathtt{S}n : N$}
\AxiomC{$(\dagger 1)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to N \to N$}
\RightLabel{(η)}
\UnaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m') : N \to N$}
\AxiomC{}
\UnaryInfC{$\vdash 1 : N$}
\BinaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m', 1) : N$}
\AxiomC{$(\dagger 2)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to \mathbf{N} \to N$}
\RightLabel{(η)}
\UnaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m') : \mathbf{N} \to N$}
\AxiomC{$(\dagger 3)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : N \to \mathbf{N} \to N$}
\AxiomC{}
\UnaryInfC{$m' : \mathbf{N} \vdash m' : N$}
\UnaryInfC{$m' : \mathbf{N} \vdash \mathtt{S}m' : N$}
\RightLabel{$(*)$}
\BinaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}m') : \mathbf{N} \to N$}
\RightLabel{(η)}
\UnaryInfC{$m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}m', n') : N$}
\BinaryInfC{$m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')) : N$}
\UnaryInfC{$m' : \mathbf{N} \vdash \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')) : \mathbf{N} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$m' : \mathbf{N} \vdash \mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n'))) : \mathbf{N} \to N$}
\RightLabel{(η)}
\UnaryInfC{$n : \mathbf{N}, m' : \mathbf{N} \vdash \mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n) : N$}
\UnaryInfC{$n : \mathbf{N} \vdash \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n) : \mathbf{N} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n)) : \mathbf{N} \to N$}
\RightLabel{(η)}
\UnaryInfC{$m : \mathbf{N}, n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n))(m) : N$}
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to \mathbf{N} \to N \ (\dagger)$}
\end{prooftree}
$$

The problem is (ap). This cuts the trace chasing of $\mathbf{N}$. The infinite path $(\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 3) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 3) \rightsquigarrow \cdots$ does not contains progressing trace.

1.2 $\mathtt{ack1} : N \to N \to N$ is not in GTC (with $(\mathrm{ap_v})$)

Claim $\mathtt{ack1} : N \to N \to N$ is well-typed, but does not satisfy GTC in the system with $(\mathrm{ap_v})$.

The situation is the same as before. In the following, weakening is implicitly applied.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$n : N \vdash n : N$}
\UnaryInfC{$n : N \vdash \mathtt{S}n : N$}
\AxiomC{$(\dagger 1)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to N \to N$}
\UnaryInfC{$m' : N \vdash \mathtt{ack1} : \mathbf{N} \to N \to N$}
\RightLabel{$(\mathrm{ap_v})$}
\UnaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m') : N \to N$}
\AxiomC{}
\UnaryInfC{$\vdash 1 : N$}
\BinaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m', 1) : N$}
\AxiomC{$(\dagger 2)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to N \to N$}
\UnaryInfC{$m' : N \vdash \mathtt{ack1} : \mathbf{N} \to N \to N$}
\RightLabel{$(\mathrm{ap_v})$}
\UnaryInfC{$m' : \mathbf{N} \vdash \mathtt{ack1}(m') : N \to N$}
\AxiomC{$(\dagger 3)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack1} : N \to \mathbf{N} \to N$}
\AxiomC{}
\UnaryInfC{$m' : N \vdash m' : N$}
\UnaryInfC{$m' : N \vdash \mathtt{S}m' : N$}
\RightLabel{$(\mathrm{ap_{\neg v}})$}
\BinaryInfC{$m' : N, n' : N \vdash \mathtt{ack1}(\mathtt{S}m') : \mathbf{N} \to N$}
\RightLabel{$(\mathrm{ap_v})$}
\UnaryInfC{$m' : N, n' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}m', n') : N$}
\BinaryInfC{$m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')) : N$}
\UnaryInfC{$m' : \mathbf{N} \vdash \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')) : \mathbf{N} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$n : N, m' : \mathbf{N} \vdash \mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n'))) : \mathbf{N} \to N$}
\RightLabel{$(\mathrm{ap_v})$}
\UnaryInfC{$n : \mathbf{N}, m' : \mathbf{N} \vdash \mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n) : N$}
\UnaryInfC{$n : \mathbf{N} \vdash \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n) : \mathbf{N} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$m : N, n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n)) : \mathbf{N} \to N$}
\RightLabel{$(\mathrm{ap_v})$}
\UnaryInfC{$m : \mathbf{N}, n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack1}(m', 1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}m', n')))(n))(m) : N$}
\UnaryInfC{$\vdash \mathtt{ack1} : \mathbf{N} \to \mathbf{N} \to N \ (\dagger)$}
\end{prooftree}
$$

2 Second term representation: ack2

We give the second representation ack2 of Ackermann function.

Definition 2.1 (ack2) We define ack2 by the following equation.

$$\mathtt{ack2} = \lambda \mathbf{m}\mathbf{N}.\mathrm{cond}(\mathtt{S}\mathbf{N}, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\mathbf{m}, n')))(\mathbf{N}))(\mathbf{m})$$

Checking the behavior of `ack2`

We check only the last case.

$$
\begin{aligned}
\mathtt{ack2}(\underline{\mathtt{S}m}, \mathtt{S}n) &\mapsto \mathrm{cond}(\mathtt{SS}n, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\underline{\mathtt{S}m}, n')))(\mathtt{S}n))(\underline{\mathtt{S}m}) \\
&\mapsto (\lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\underline{\mathtt{S}m}, n')))(\mathtt{S}n))(\underline{m}) \\
&\mapsto \mathrm{cond}(\mathtt{ack2}(m, 1), \lambda n'.\mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}m}, n')))(\mathtt{S}n) \\
&\mapsto (\lambda n'.\mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}m}, n')))(n) \\
&\mapsto \mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}m}, n))
\end{aligned}
$$

The point of `ack2` is that $\underline{\mathtt{S}m}$ at the first line and the one at the last line are exactly the same.

2.1 $\mathtt{ack2} : N \to N \to N$ is not in GTC (with (η))

Claim $\mathtt{ack2} : N \to N \to N$ is well-typed, but does not satisfy GTC in the system with (η).

$$
\dfrac{
  \dfrac{
    \dfrac{\dfrac{}{n : N \vdash n : N}}{n : N \vdash \mathtt{S}n : N}
    \quad
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{\overset{(\dagger 1)}{\vdash \mathtt{ack2} : N \to N \to N}}{m' : N \vdash \mathtt{ack2}(m') : N \to N}\,(\eta)
            \quad
            \dfrac{}{\vdash 1 : N}
          }{m' : N \vdash \mathtt{ack2}(m', 1) : N}
          \quad
          \dfrac{
            \dfrac{
              \dfrac{\overset{(\dagger 2)}{\vdash \mathtt{ack2} : N \to N \to N}}{m' : N \vdash \mathtt{ack2}(m') : N \to N}\,(\eta)
              \quad
              \dfrac{\dfrac{\overset{(\dagger 3)}{\vdash \mathtt{ack2} : \mathbf{N} \to \mathbf{N} \to N}}{m : \mathbf{N} \vdash \mathtt{ack2}(m) : \mathbf{N} \to N}\,(\eta)}{m : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack2}(m, n') : N}\,(\eta)
            }{m : \mathbf{N}, m' : N, n' : \mathbf{N} \vdash \mathtt{ack2}(m', \mathtt{ack2}(m, n')) : N}
          }{m : \mathbf{N}, m' : N \vdash \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')) : \mathbf{N} \to N}
        }{m : \mathbf{N}, m' : N \vdash \mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n'))) : \mathbf{N} \to N}\,(\mathrm{cond})
      }{m : \mathbf{N}, n : \mathbf{N}, m' : N \vdash \mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n) : N}\,(\eta)
    }{m : \mathbf{N}, n : \mathbf{N} \vdash \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n) : N \to N}
  }{m : \mathbf{N}, n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n)) : N \to N}\,(\mathrm{cond})
  \quad
  \dfrac{}{m : \mathbf{N} \vdash m : N}
}{
  \dfrac{m : \mathbf{N}, n : \mathbf{N} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n))(m) : N}{\vdash \mathtt{ack2} : \mathbf{N} \to \mathbf{N} \to N\ (\dagger)}
}\,(\mathrm{ap})
$$

The problem is (ap). This cuts the trace chasing of $\mathbf{N}$. The infinite path $(\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow \cdots$ does not contains progressing trace.

2.2 $\mathtt{ack2} : N \to N \to N$ is not in GTC (with $(\mathbf{ap_v})$)

Proposition 2.2 $\mathtt{ack2} : N \to N \to N$ *has a proof that satisfis GTC in the system with* (ap_v).

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$n : N \vdash n : N$}
\UnaryInfC{$n : N \vdash \mathtt{S}n : N$}
\AxiomC{$(\dagger 1)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack2} : \textcolor{blue}{\mathbf{N}} \to N \to N$}
\UnaryInfC{$m' : N \vdash \mathtt{ack2} : \textcolor{blue}{\mathbf{N}} \to N \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m' : \textcolor{blue}{\mathbf{N}} \vdash \mathtt{ack2}(m') : N \to N$}
\AxiomC{}
\UnaryInfC{$\vdash 1 : N$}
\BinaryInfC{$m' : \textcolor{blue}{\mathbf{N}} \vdash \mathtt{ack2}(m', 1) : N$}
\AxiomC{$(\dagger 2)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack2} : \textcolor{blue}{\mathbf{N}} \to N \to N$}
\UnaryInfC{$m' : N \vdash \mathtt{ack2} : \textcolor{blue}{\mathbf{N}} \to N \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m' : \textcolor{blue}{\mathbf{N}} \vdash \mathtt{ack2}(m') : N \to N$}
\AxiomC{$(\dagger 3)$}
\noLine
\UnaryInfC{$\vdash \mathtt{ack2} : \textcolor{red}{\mathbf{N}} \to \textcolor{olive}{\mathbf{N}} \to N$}
\UnaryInfC{$m : N \vdash \mathtt{ack2} : \textcolor{red}{\mathbf{N}} \to \textcolor{olive}{\mathbf{N}} \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}} \vdash \mathtt{ack2}(m) : \textcolor{olive}{\mathbf{N}} \to N$}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}}, n' : \textcolor{olive}{\mathbf{N}} \vdash \mathtt{ack2}(m) : \textcolor{olive}{\mathbf{N}} \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}}, n' : \textcolor{olive}{\mathbf{N}} \vdash \mathtt{ack2}(m, n') : N$}
\BinaryInfC{$m : \textcolor{red}{\mathbf{N}}, m' : \textcolor{blue}{\mathbf{N}}, n' : \textcolor{olive}{\mathbf{N}} \vdash \mathtt{ack2}(m', \mathtt{ack2}(m, n')) : N$}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}}, m' : \textcolor{blue}{\mathbf{N}} \vdash \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')) : \textcolor{olive}{\mathbf{N}} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$m : \textcolor{red}{\mathbf{N}}, n : N, m' : \textcolor{blue}{\mathbf{N}} \vdash \mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n'))) : \textcolor{olive}{\mathbf{N}} \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}}, n : \textcolor{olive}{\mathbf{N}}, m' : \textcolor{blue}{\mathbf{N}} \vdash \mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n) : N$}
\UnaryInfC{$m : \textcolor{red}{\mathbf{N}}, n : \textcolor{olive}{\mathbf{N}} \vdash \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n) : \textcolor{blue}{\mathbf{N}} \to N$}
\RightLabel{(cond)}
\BinaryInfC{$m : \textcolor{red}{\mathbf{N}}, n : \textcolor{olive}{\mathbf{N}} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n)) : \textcolor{blue}{\mathbf{N}} \to N$}
\RightLabel{(ap$_\mathrm{v}$)}
\UnaryInfC{$m : \textcolor{purple}{\mathbf{N}}, n : \textcolor{olive}{\mathbf{N}} \vdash \mathrm{cond}(\mathtt{S}n, \lambda m'.\mathrm{cond}(\mathtt{ack2}(m', 1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m, n')))(n))(m) : N$}
\UnaryInfC{$\vdash \mathtt{ack2} : \textcolor{purple}{\mathbf{N}} \to \textcolor{olive}{\mathbf{N}} \to N \ (\dagger)$}
\end{prooftree}
$$

This proof satisfies the global trace condition. Note that:

- The path $(\dagger) \rightsquigarrow (\dagger 1)$ contains a progressing trace $\tau_1 = (\textcolor{purple}{\mathbf{N}}, \textcolor{purple}{\mathbf{N}}, \textcolor{blue}{\mathbf{N}}, \dots, \textcolor{blue}{\mathbf{N}})$.
- The path $(\dagger) \rightsquigarrow (\dagger 2)$ contains a progressing trace $\tau_2 = (\textcolor{purple}{\mathbf{N}}, \textcolor{purple}{\mathbf{N}}, \textcolor{blue}{\mathbf{N}}, \dots, \textcolor{blue}{\mathbf{N}})$.
- The path $(\dagger) \rightsquigarrow (\dagger 3)$ contains a progressing trace $\tau'_3 = (\textcolor{olive}{\mathbf{N}}, \dots, \textcolor{olive}{\mathbf{N}})$ and a non-progressing trace $\tau_3 = (\textcolor{purple}{\mathbf{N}}, \textcolor{purple}{\mathbf{N}}, \textcolor{red}{\mathbf{N}}, \dots, \textcolor{red}{\mathbf{N}})$.

Take an infinite path π from this proof. If π passes through $(\dagger 1)$ (or $(\dagger 2)$) infinitely many times, take its infinitely progressing trace by combining τ_1, τ_2, and τ_3. If π passes through $(\dagger 1)$ and $(\dagger 2)$ finitely many times, namely it eventually becomes a loop of $(\dagger)$ and $(\dagger 3)$, take its infinitely progressing trace by combining τ_1, τ_2, and τ'_3.