well-founded recursion, part 2.

Johan G. F. Belinfante
revised 2004 August 5

summary

In the first notebook on well-founded recursion, the constructors \texttt{partrec[x,y]} and \texttt{rec[x,y]} were introduced, and some of their elementary properties were derived, culminating with the theorem that if \texttt{x} is a function and \texttt{y} is a thin relation whose inverse is a well-founded relation, then \texttt{rec[x,y]} is a function. In this second notebook on well-founded recursion, it is shown that under the same conditions, the function \texttt{rec[x,y]} satisfies the same recursion relation that the partial solutions satisfy. The recursion condition for \texttt{rec[x,y]} can be expressed either as an inclusion, or as a statement involving \texttt{APPLY}, the latter requiring one to introduce an additional variable \texttt{z} to denote the argument to which \texttt{rec[x,y]} is to be applied. All the results in this notebook are obtained twice, one version with \texttt{funpart} and \texttt{thinpars} wrappers, and again with literals replacing the wrappers.

lemma

In this notebook, it is convenient to turn off the \texttt{simplify} flag. Initially, the \texttt{cond} flag will also be turned off, but later it will be turned back on.

\begin{verbatim}
In[2]:= simplify = False; cond = False;
\end{verbatim}

The following lemma is useful for removing \texttt{funpart} and \texttt{thinpars} wrappers.

\begin{verbatim}
In[3]:= Map[equal[0, #] &,
             symdif[partrec[x, y], partrec[x, composite[Id, y]]] // Normality]
Out[3]= equal[partrec[x, y], partrec[x, composite[Id, y]]] = True
\end{verbatim}
replacing wrappers with literals

As a simple illustration, the following main result of the first notebook will be restated without wrappers:

\[
\text{In}[7]:= \text{implies}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{FUNCTION}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]
\]

\[
\text{Out}[7]= \text{True}
\]

The negative form of this is needed to remove the wrappers.

\[
\text{In}[8]:= \text{and}[\text{WELLFOUNDED}[\text{inverse}[y]],
\text{not}[\text{FUNCTION}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]] // \text{NotNotTest}
\]

\[
\text{Out}[8]= \text{and}[\text{not}[\text{FUNCTION}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]]],
\text{WELLFOUNDED}[\text{inverse}[y]]] = \text{False}
\]

\[
\text{In}[9]:= (\% /. \{x \to x\_, \ y \to y\_\}) /. \text{Equal} \rightarrow \text{SetDelayed}
\]

The wrappers are replaced with literals using equality substitution, as follows:

\[
\text{In}[10]:= \text{SubstTest}[\text{implies}, \text{and}[\text{equal}[x, u], \text{equal}[v, w]],
\text{implies}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{FUNCTION}[\text{rec}[u, v]]]],
\text{implies}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{FUNCTION}[\text{rec}[x, w]]],
\{u \to \text{funpart}[x], v \to \text{thinpart}[y], w \to \text{composite}[\text{Id}, y]\}] // \text{MapNotNot}
\]

\[
\text{Out}[10]= \text{or}[\text{FUNCTION}[\text{rec}[x, y]], \text{not}[\text{equal}[V, \text{domain}[\text{VERTSECT}[y]]]],
\text{not}[\text{FUNCTION}[x]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]] = \text{True}
\]

\[
\text{In}[11]:= \text{or}[\text{FUNCTION}[\text{rec}[x\_, y\_]], \text{not}[\text{equal}[V, \text{domain}[\text{VERTSECT}[y\_]]]],
\text{not}[\text{FUNCTION}[x\_]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y\_]]]] := \text{True}
\]
obtaining values of rec[x,y] from those of partial solutions

In this section it will be shown how to derive properties of the complete solution \( \text{rec}[x,y] \) from those of the partial solutions. The derivations are quite time-consuming unless broken up into pieces. The first goal is to show that at any point of the domain of a partial solution, the restriction of the partial solution that appears on the right side of the recursion relation can be replaced with a corresponding restriction of the complete solution. This is done in two steps:

\[\begin{align*}
\text{In [12]}: &= \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p2], \text{implies}[p2, p3], \\
&\quad \text{implies}[p1, p4], \text{implies}[\text{and}[p3, p4], p5], \text{not}[\text{implies}[p1, p5]], \\
&\quad \{p1 \rightarrow \text{and}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{member}[z, \text{domain}[w]], \\
&\quad \ \text{member}[w, \text{partrc}[\text{funpart}[x], \text{thinpart}[y]]], \\
&\quad p2 \rightarrow \text{subclass}[w, \text{rec}[\text{funpart}[x], \text{thinpart}[y]]], \\
&\quad p3 \rightarrow \text{equal}[\text{domain}[w], \text{fix}[\text{composite}[\text{inverse}[w], \\
&\quad \text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]], \\
&\quad p4 \rightarrow \text{subclass}[\text{image}[\text{thinpart}[y], \text{singleton}[z]], \text{domain}[w]], \\
&\quad p5 \rightarrow \text{subclass}[\text{image}[\text{thinpart}[y], \text{singleton}[z]], \\
&\quad \text{fix}[\text{composite}[\text{inverse}[w], \text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]] \\
\text{Out [12]}: &= \text{or}[\text{not}[\text{member}[w, \text{partrc}[\text{funpart}[x], \text{thinpart}[y]]]], \\
&\quad \text{not}[\text{member}[z, \text{domain}[w]]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]], \\
&\quad \text{subclass}[\text{image}[\text{thinpart}[y], \text{singleton}[z]], \\
&\quad \text{fix}[\text{composite}[\text{inverse}[w], \text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]] = \text{True}
\end{align*}\]

\[\begin{align*}
\text{In [13]}: &= (\% / . (w \rightarrow w_, x \rightarrow x_, y \rightarrow y_, z \rightarrow z_-)) / . \text{Equal} \rightarrow \text{SetDelayed} \\
\text{In [14]}: &= \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p2], \text{implies}[p1, p3], \text{implies}[p1, p4], \\
&\quad \text{implies}[\text{and}[p2, p3, p4], p5], \text{not}[\text{implies}[p1, p5]], \\
&\quad \{p1 \rightarrow \text{and}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{member}[z, \text{domain}[w]], \\
&\quad \ \text{member}[w, \text{partrc}[\text{funpart}[x], \text{thinpart}[y]]], \\
&\quad p2 \rightarrow \text{subclass}[\text{image}[\text{thinpart}[y], \text{singleton}[z]], \\
&\quad \text{fix}[\text{composite}[\text{inverse}[w], \text{rec}[\text{funpart}[x], \text{thinpart}[y]]]]], \\
&\quad p3 \rightarrow \text{FUNCTION}[w], \\
&\quad p4 \rightarrow \text{FUNCTION}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]], \\
&\quad p5 \rightarrow \text{equal}[\text{composite}[w, \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]], \text{composite}[ \\
&\quad \text{rec}[\text{funpart}[x], \text{thinpart}[y]], \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]]] \\
\text{Out [14]}: &= \text{or}[\text{equal}[\text{composite}[w, \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]], \text{composite}[ \\
&\quad \text{rec}[\text{funpart}[x], \text{thinpart}[y]], \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]], \\
&\quad \text{not}[\text{member}[w, \text{partrc}[\text{funpart}[x], \text{thinpart}[y]]]], \\
&\quad \text{not}[\text{member}[z, \text{domain}[w]]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]) = \text{True}
\end{align*}\]

\[\begin{align*}
\text{In [15]}: &= (\% / . (w \rightarrow w_, x \rightarrow x_, y \rightarrow y_, z \rightarrow z_-)) / . \text{Equal} \rightarrow \text{SetDelayed}
\end{align*}\]
an APPLY formula

The result in this section originally took 148 seconds until it was broken up into two pieces. Now it takes very little time.

Lemma 1.

\[ \text{In[16]} = \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p2], \text{implies}[p1, p3], \text{implies}[p1, p2, p3], \text{implies}[p1, p2, p3, p4], \text{not}[\text{implies}[p1, p4]], \{p1 \rightarrow \text{and}[\text{WELLFOUNDED}[\text{inverse}[y]], \text{member}[z, \text{domain}[w]], \text{member}[w, \text{partrec}[\text{funpart}[x], \text{thinp}[y]]], p2 \rightarrow \text{subclass}[w, \text{rec}[\text{funpart}[x], \text{thinp}[y]]], p3 \rightarrow \text{FUNCTION}[\text{rec}[\text{funpart}[x], \text{thinp}[y]]], p4 \rightarrow \text{equal}[\text{APPLY}[w, z], \text{APPLY}[\text{rec}[\text{funpart}[x], \text{thinp}[y]], z]]]] \]

\[ \text{Out[16]} = \text{or}[\text{equal}[\text{APPLY}[w, z], \text{APPLY}[\text{rec}[\text{funpart}[x], \text{thinp}[y]], z]], \text{not}[\text{member}[w, \text{partrec}[\text{funpart}[x], \text{thinp}[y]]]], \text{not}[\text{member}[z, \text{domain}[w]]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]] = \text{True} \]

\[ \text{In[17]} = (\% \/. \{w \rightarrow \_\_, x \rightarrow \_\_, y \rightarrow \_\_, z \rightarrow \_\_\}) \/. \text{Equal} \rightarrow \text{SetDelayed} \]

Lemma 2.
This result gives an equation for applying the complete solution to an argument \( z \) under the condition that this argument is in the domain of some partial solution. This condition is equivalent to requiring that \( z \) be in the domain of the complete solution. To show this, one must eliminate the variable \( w \). Unfortunately, a direct approach to this elimination did not work because of the presence of \texttt{APPLY}, which one can not easily shield from the \texttt{class} rules.

---

\textbf{eliminating the APPLY}

\texttt{In[20]}:= \texttt{cond }= \texttt{True;}

The key to eliminating the \texttt{APPLY} is to rewrite the statement in the preceding section as a membership of \( z \) in a domain of commonality for two functions. For convenience, a new variable \( t \) is introduced to simplify this reasoning process, although introducing this equation does contribute to the execution time. To expedite matters, a lemma is introduced, which alone takes almost a minute:
The following step not only causes `APPLY` to be eliminated, but at the same time the temporary variable `t` is eliminated.
At this point it is feasible to eliminate the variables \( w \) and \( z \).

The final step is to use the fact that if the domain of one function is contained in the domain of commonality of it and another function, then the other is an extension of the first one.

The main result is first derived in the following negative form to facilitate removing the wrappers:
This result becomes more readable if the `funpart` and `thinpart` wrappers are removed and replaced with literals.

This says that if `x` is a function and if `y` is thin, and its `inverse` is well-founded, then the function `rec[x,y]` satisfies the same recursion relation that the partial solutions do. To replace this inclusion with an equation, it is necessary to study the domain of the function `rec[x,y]`. This matter will be taken up in another notebook.
rewriting the recursion relation in terms of APPLY

The recursion relation derived in the preceding section can be rewritten in terms of APPLY. The result with wrappers of the preceding section is now needed in positive form.

\[
\text{In}[33]:= \text{or[subclass[rec[funpart[x], thinpart[y]], composite[funpart[x],}
\text{id[composite[IMAGE[composite[id[rec[funpart[x], thinpart[y]]],}
\text{inverse[FIRST]]], VERTSECT[thinpart[y]]]], inverse[FIRST]]],}
\text{not[WELLFOUNDED[inverse[y]]]} // \text{NotNotTest}
\]

\[
\text{Out}[33]= \text{or[not[WELLFOUNDED[inverse[y]]]},
\text{subclass[rec[funpart[x], thinpart[y]], composite[funpart[x],}
\text{id[composite[IMAGE[composite[id[rec[funpart[x], thinpart[y]]],}
\text{inverse[FIRST]]], VERTSECT[thinpart[y]]]], inverse[FIRST]]]] = \text{True}
\]

\[
\text{In}[34]:= (\% /\ {x \rightarrow x\_, y \rightarrow y\_}) /\ . \text{Equal} \rightarrow \text{SetDelayed}
\]

Reintroducing APPLY, and using wrappers, the following formulation of the recursion relation is obtained:

\[
\text{In}[35]:= \text{Map[not, SubstTest[and, implies[p2, p3], implies[p2, p4],}
\text{implies[and[p1, p3, p4], p5], not[implies[and[p1, p2], p5]],}
\text{p1 \rightarrow \text{member}[z, \text{domain[rec[funpart[x], thinpart[y]]]]],}
\text{p2 \rightarrow WELLFOUNDED[inverse[y]],}
\text{p3 \rightarrow \text{FUNCTION[rec[funpart[x], thinpart[y]],}}
\text{p4 \rightarrow \text{subclass[rec[funpart[x], thinpart[y]], composite[funpart[x],}
\text{id[composite[IMAGE[composite[id[rec[funpart[x], thinpart[y]]],}
\text{inverse[FIRST]]], VERTSECT[thinpart[y]]]], inverse[FIRST]]]],}
\text{p5 \rightarrow \text{equal[APPLY[funpart[x], PAIR[z, composite[rec[funpart[x],}
\text{thinpart[y]], id[\text{image[thinpart[y], singleton[z]]]]]]],}
\text{APPLY[rec[funpart[x], thinpart[y]], thinpart[y]], z]]]]]
\]

\[
\text{Out}[35]= \text{or[equal[APPLY[funpart[x], PAIR[z, composite[
\text{rec[funpart[x], thinpart[y]], id[\text{image[thinpart[y], singleton[z]]]]],}
\text{APPLY[rec[funpart[x], thinpart[y]], z]],}
\text{not[member[z, \text{domain[rec[funpart[x], thinpart[y]]]]]]],}
\text{not[WELLFOUNDED[inverse[y]]}] = \text{True}
\]

\[
\text{In}[36]:= \text{or[}
\text{equal[APPLY[funpart[x\_], PAIR[z\_, composite[rec[funpart[x\_], thinpart[y\_]],}
\text{id[\text{image[thinpart[y\_], singleton[z\_]]]]],}
\text{APPLY[rec[funpart[x\_], thinpart[y\_]], z\_]],}
\text{not[member[z\_, \text{domain[rec[funpart[x\_], thinpart[y\_]]]]]]],}
\text{not[WELLFOUNDED[inverse[y\_]]]] := \text{True}
\]
The `funpart` and `thinpart` wrappers could be replaced with literals, if desired. For this one needs the negative version of the above:

\[
\begin{align*}
\text{In}[37]:= & \quad \text{and}[\text{.member}[z, \text{domain}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]]],
\quad \text{not}[\text{equal}[\text{APPLY}[\text{funpart}[x], \text{PAIR}[z, \text{composite}[
\quad \text{rec}[\text{funpart}[x], \text{thinpart}[y]], \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]]]]],
\quad \text{APPLY}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]], z]],
\quad \text{WELLFOUNDED}[\text{inverse}[y]]] \\
& \quad \text{// NotNotTest}
\end{align*}
\]

\[
\begin{align*}
\text{Out}[37]:= & \quad \text{and}[\text{member}[z, \text{domain}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]]]],
\quad \text{not}[\text{equal}[\text{APPLY}[\text{funpart}[x], \text{PAIR}[z, \text{composite}[
\quad \text{rec}[\text{funpart}[x], \text{thinpart}[y]], \text{id}[\text{image}[\text{thinpart}[y], \text{singleton}[z]]]]]]],
\quad \text{APPLY}[\text{rec}[\text{funpart}[x], \text{thinpart}[y]], z]], \text{WELLFOUNDED}[\text{inverse}[y]]] = \text{False}
\end{align*}
\]

\[
\begin{align*}
\text{In}[38]:= & \quad (\% /\! / \{x \to x_-, y \to y_-, z \to z_-\}) /\! / \text{Equal} \to \text{SetDelayed}
\end{align*}
\]

The wrappers are now removed:

\[
\begin{align*}
\text{In}[39]:= & \quad \text{SubstTest}[\text{implies}, \text{and}[\text{equal}[x, u], \text{equal}[v, w]], \text{or}[
\quad \text{equal}[\text{APPLY}[u, \\text{PAIR}[z, \text{composite}[
\quad \text{rec}[u, v]], \text{id}[\text{image}[v, \text{singleton}[z]]]]]],
\quad \text{APPLY}[\text{rec}[u, v], z]], \text{not}[\text{member}[z, \text{domain}[\text{rec}[u, v]]]],
\quad \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]]],
\quad \text{or}[\text{equal}[\text{APPLY}[x, \\text{PAIR}[z, \text{composite}[
\quad \text{rec}[x, w]], \text{id}[\text{image}[w, \text{singleton}[z]]]]]],
\quad \text{APPLY}[\text{rec}[x, w], z]],
\quad \text{not}[\text{member}[z, \text{domain}[\text{rec}[x, w]]]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]],
\quad \{u \to \text{funpart}[x], v \to \text{thinpart}[y], w \to \text{composite}[\text{Id}, y]\}]
\end{align*}
\]

\[
\begin{align*}
\text{Out}[39]:= & \quad \text{or}[\text{equal}[\text{APPLY}[x, \\text{PAIR}[z, \text{composite}[
\quad \text{rec}[x, y]], \text{id}[\text{image}[y, \text{singleton}[z]]]]]],
\quad \text{APPLY}[\text{rec}[x, y], z]], \text{not}[\text{equal}[v, \text{domain}[\text{VERTSECT}[y]]]], \text{not}[\text{FUNCTION}[x]],
\quad \text{not}[\text{member}[z, \text{domain}[\text{rec}[x, y]]]], \text{not}[\text{WELLFOUNDED}[\text{inverse}[y]]]] = \text{True}
\end{align*}
\]

\[
\begin{align*}
\text{In}[40]:= & \quad \text{or}[\text{equal}[
\quad \text{APPLY}[x_-, \\text{PAIR}[z_-, \text{composite}[
\quad \text{rec}[x_-, y_-]], \text{id}[\text{image}[y_, \text{singleton}[z_-]]]]]],
\quad \text{APPLY}[\text{rec}[x_, y_-], z_-]], \text{not}[\text{equal}[v, \text{domain}[\text{VERTSECT}[y_-]]]],
\quad \text{not}[\text{FUNCTION}[x_-]], \text{not}[\text{member}[z_, \text{domain}[\text{rec}[x_, y_-]]]],
\quad \text{not}[\text{WELLFOUNDED}[\text{inverse}[y_-]]]] = \text{True}
\end{align*}
\]