WFPART = lambda[x, wf[x]]

Johan G. F. Belinfante
2004 September 25

In [1] := SetDirectory["i:"]; << goedel61.24a; << tools.m

:Package Title: goedel61.24a 2004 September 24 at 5:50 p.m.

It is now: 2004 Sep 25 at 8:58

Loading Simplification Rules

TOOLS.M Revised 2004 September 18

weightlimit = 40

summary

The function \( \text{WFPART} = \text{lambda}[x, \text{wf}[x]] \) is studied in this notebook, and a reification formula for \( \text{wf}[x] \) is derived.

definition of WFPART

The following membership rule will serve to define \( \text{WFPART} \).

In [2] := \text{member}[x\_, \text{WFPART}] := \text{and}[\text{member}[\text{first}[x], V]\_, \text{equal}[\text{second}[x], \text{wf}[\text{first}[x]]]]

It is immediately evident from the definition that \( \text{WFPART} \) is a relation.

In [3] := \text{Map}[\text{equal}[0, \#] \&, \text{dif}[\text{WFPART}, \text{cart}[V, V]]] // Normality]

Out[3] = \text{subclass}[\text{WFPART}, \text{cart}[V, V]] = \text{True}

In [4] := \text{subclass}[\text{WFPART}, \text{cart}[V, V]] := \text{True}

In [5] := \text{equal}[\text{composite}[\text{Id}, \text{WFPART}], \text{WFPART}]

Out[5] = \text{True}

In [6] := \text{composite}[\text{Id}, \text{WFPART}] := \text{WFPART}

Another immediate consequence is a formula for its fixed point class.
vertical section rule

In this section, a vertical section rule for WFPART is derived. To produce a clean formula, some lemmas are derived. The first one is needed to get rid of an image[V,x] expression.

\[
\text{In}[9] := \text{wf[union[x, complement[image[V, y]]]] // Normality}
\]
\[
\text{Out}[9] = \text{wf[union[x, complement[image[V, y]]]] == composite[id[image[V, y]], wf[x]]}
\]
\[
\text{In}[10] := \text{wf[union[x_, complement[image[V, y_]]]] := composite[id[image[V, y]], wf[x]]}
\]

The second lemma is not specific to \text{wf[x]}. It reduces a double occurrence of an image[V,x] expression to a single one.

\[
\text{In}[11] := \text{equal[intersection[image[V, x], singleton[composite[id[image[V, x]], y]]], intersection[image[V, x], singleton[composite[Id, y]]]] // assert}
\]
\[
\text{Out}[11] = \text{True}
\]
\[
\text{In}[12] := \text{intersection[image[V, x_], singleton[composite[id[image[V, x_]], y_]]] := intersection[image[V, x], singleton[composite[Id, y]]]}
\]

The vertical section rule is now clean:

\[
\text{In}[13] := \text{image[WFPART, singleton[x]] // Normality}
\]
\[
\text{Out}[13] = \text{image[WFPART, singleton[x]] == intersection[image[V, singleton[x]], singleton[wf[x]]]}
\]
\[
\text{In}[14] := \text{image[WFPART, singleton[x_]] := intersection[image[V, singleton[x]], singleton[wf[x]]]}
\]

WFPART is a function

The derivation in this section can be done without turning off flags, but doing so would increase the execution time from 6.5 seconds to 16 seconds.
In[15]:= simplify = False; cond = False;

With the vertical section rule in place, one can derive the fact that \texttt{WFPART} is a function as follows:

\begin{verbatim}
In[16]:= Map[equal[0, #] & , dif[WFPART, funpart[WFPART]]] // RelnNormality
In[17]:= FUNCTION[WFPART] := True
\end{verbatim}

It is worth noting that the test used for this is \texttt{RelnNormality}, and not \texttt{VSNormality}, as one might have expected because the presence of the vertical section rule is crucial for this derivation. The \texttt{VSNormality} test would not work here.

---

normalization

To make progress, one needs to normalize \texttt{WFPART}, which requires turning the \texttt{cond} flag back on. The \texttt{simplify} flag can be left off for now.

\begin{verbatim}
In[18]:= cond = True;
In[19]:= WFPART // VSNormality // Reverse
Out[19]= intersection[composite[inverse[S], IMAGE[id[cart[V, V]]]],
          composite[inverse[IMAGE[SECOND]], DISJOINT, BIGCUP, SUBVAR],
          fix[composite[inverse[DIF], inverse[IMAGE[SECOND]],
              inverse[S], BIGCUP, SUBVAR, FIRST]]] = WFPART
In[20]:= % /. Equal -> SetDelayed
\end{verbatim}

---

range

The function \texttt{WFPART} is idempotent.

\begin{verbatim}
In[21]:= composite[WFPART, WFPART] // VSNormality
In[22]:= composite[WFPART, WFPART] := WFPART
\end{verbatim}
In[23]:  SubstTest[implies, and[FUNCTION[x], idempotent[x]],
        equal[range[x], fix[x]], x \rightarrow WFPART]

Out[23]= equal[WF, range[WFPART]] \Rightarrow True

In[24]:  range[WFPART] := WF

The *simplify* flag needs to be turned back on now.

In[25]:  simplify = True;

Corollary.

In[26]:  Assoc[id[P[cart[V, V]]], id[WF], WFPART]

Out[26]= composite[id[P[cart[V, V]]], WFPART] \Rightarrow WFPART

In[27]:  composite[id[P[cart[V, V]]], WFPART] := WFPART

Some related results:

In[28]:  Assoc[WFPART, WFPART, inverse[WFPART]]

Out[28]= composite[WFPART, id[WF]] \Rightarrow id[WF]

In[29]:  composite[WFPART, id[WF]] := id[WF]

In[30]:  ImageComp[WFPART, WFPART, V] // Reverse

Out[30]= image[WFPART, WF] \Rightarrow WF

In[31]:  image[WFPART, WF] := WF

---

domain

The function *WFPART* is total:

In[32]:  Map[domain,
        class[pair[x, y], member[y, wf[x]]] // VERTSECT // RelnNormality // Reverse]

Out[32]= domain[WFPART] \Rightarrow V

In[33]:  domain[WFPART] := V

A similar computation yields a formula for *composite[inverse[E], WFPART]*.
The following simple formula for \( \text{WFPART} \) follows as a corollary:

\[
\text{In } [36] := \text{VERTSECT[intersection[composite[inverse[SECOND]], complement[inverse[E]], BIGCUP, SUBVAR], inverse[E]]}
\]

\[
\text{Out } [36] := \text{WFPART}
\]

---

**APPLY formula**

**Lemma.**

\[
\text{In } [37] := \text{SubstTest[implies, and[subclass[z, x], member[x, V]], member[z, V], z \rightarrow \text{composite[Id, w]}} \ . \ w \rightarrow \text{wf[x]}
\]

\[
\text{Out } [37] := \text{or[and[member[domain[wf[x]], V], member[range[wf[x]], V]], not[member[x, V]]]} = \text{True}
\]

\[
\text{In } [38] := \% / . x \rightarrow x_\text{_.} / . \text{Equal} \rightarrow \text{SetDelayed}
\]

**Corollary of the lemma.**

\[
\text{In } [39] := \text{equal[union[complement[image[V, singleton[x]]]], complement[image[V, singleton[domain[wf[x]]]]], complement[image[V, singleton[range[wf[x]]]]], wf[x]], union[complement[image[V, singleton[x]]], wf[x]]}
\]

\[
\text{Out } [39] := \text{True}
\]

\[
\text{In } [40] := \text{union[complement[image[V, singleton[x_\text{_.}]]], complement[image[V, singleton[domain[wf[x_\text{_.}]]]]], complement[image[V, singleton[range[wf[x_\text{_.}]]]]], wf[x_\text{_.}]] := union[complement[image[V, singleton[x_\text{_.}]]], wf[x]]}
\]

\[
\text{In } [41] := \text{SubstTest[A, image[w, singleton[x]], w \rightarrow \text{WFPART}] } / \text{ Reverse}
\]

\[
\text{Out } [41] := \text{APPLY[\text{WFPART}, x] := union[complement[image[V, singleton[x]]], wf[x]]}
\]

\[
\text{In } [42] := \text{APPLY[\text{WFPART}, x_\text{_.}] := union[complement[image[V, singleton[x_\text{_.}]]], wf[x]]}
\]
composites with \( \text{IMAGE}[\text{id[cart[V,V]]}] \)

The following result can also be derived using \textbf{RelnRenormality}.

\begin{verbatim}
In[43]:= \text{Assoc}[\text{IMAGE}[\text{id[cart[V,V]]}], \text{id[P[cart[V,V]]]}, \text{WFPart}] 
Out[43]= \text{composite[IMAGE[id[cart[V,V]]], WFPart]} \equiv \text{WFPart}
In[44]:= \text{composite[IMAGE[id[cart[V,V]]], WFPart]} := \text{WFPart}
\end{verbatim}

The composite in the reverse order can be derived most simply using \textbf{RelnNormality}.

\begin{verbatim}
In[45]:= \text{composite[WFPart, IMAGE[id[cart[V,V]]]]} // \text{RelnNormality} 
Out[45]= \text{composite[WFPart, IMAGE[id[cart[V,V]]]]} \equiv \text{WFPart}
In[46]:= \text{composite[WFPart, IMAGE[id[cart[V,V]]]]} := \text{WFPart}
\end{verbatim}

other composites

\textbf{Theorem}  This formula is a variable-free version of the equation  \( \text{wf[id[x]]} = 0 \).

\begin{verbatim}
In[47]:= \text{composite[WFPart, IDP]} // \text{VSNormality} 
Out[47]= \text{composite[WFPart, IMAGE[DUP]]} \equiv \text{cart[V, singleton[0]]}
In[48]:= \text{composite[WFPart, IMAGE[DUP]]} := \text{cart[V, singleton[0]]}
\end{verbatim}

Deriving a variable-free counterpart of the equation  \( \text{fix[wf[x]]} = 0 \) is somewhat trickier:

\begin{verbatim}
In[49]:= \text{Map[VERTSECT,} 
\text{composite[\text{inverse[E]}, IMAGE[\text{inverse[DUP]]]}, \text{WFPart}] // \text{VSRenormality]} 
Out[49]= \text{composite[IMAGE[\text{inverse[DUP]]}, \text{WFPart}]} \equiv \text{cart[V, singleton[0]]}
In[50]:= \text{composite[IMAGE[\text{inverse[DUP]]}, \text{WFPart}]} := \text{cart[V, singleton[0]]}
\end{verbatim}

\textbf{Corollary}.

\begin{verbatim}
In[51]:= \text{ImageComp[IMAGE[\text{inverse[DUP]]}, \text{WFPart}, V]} // \text{Reverse} 
Out[51]= \text{image[IMAGE[\text{inverse[DUP]]}, \text{WF}]} = \text{singleton[0]}
In[52]:= \text{image[IMAGE[\text{inverse[DUP]]}, \text{WF}]} := \text{singleton[0]}
\end{verbatim}
Such results can more easily obtained using reification. The needed formulas are derived in the next section.

reify

The following lemmas are used to eliminate RIF.

\[
\text{In}[53]:= \text{fix[composite[complement[composite[}
\text{intersection[composite[inverse[FIRST], inverse[SECOND], inverse[x]],}
\text{composite[inverse[SECOND], E, FIRST]], inverse[SECOND]]],}
\text{inverse[E], SECOND]]} \quad /\text{ InvertFixTest}
\]

\[
\text{Out}[53]= \text{fix[composite[complement[composite[}
\text{intersection[composite[inverse[FIRST], inverse[SECOND], inverse[x]],}
\text{composite[inverse[SECOND], E, FIRST]], inverse[SECOND]]],}
\text{inverse[E], SECOND]]} = \text{fix[composite[inverse[SECOND], E,}
\text{complement[composite[SECOND, intersection[composite[x, SECOND, FIRST],}
\text{composite[inverse[FIRST], inverse[E], SECOND]]]]}}]
\]

\[
\text{In}[54]:= \text{fix[composite[complement[composite[}
\text{intersection[composite[inverse[FIRST], inverse[SECOND], inverse[x_]],}
\text{composite[inverse[SECOND], E, FIRST]], inverse[SECOND]]],}
\text{inverse[E], SECOND]]} := \text{fix[composite[inverse[SECOND], E,}
\text{complement[composite[SECOND, intersection[composite[x, SECOND, FIRST],}
\text{composite[inverse[FIRST], inverse[E], SECOND]]]]}}]
\]

\[
\text{In}[55]:= \text{Map[inverse,}
\text{composite[intersection[composite[inverse[x], SECOND], composite[complement[}
\text{composite[complement[composite[composite[inverse[x],}
\text{cross[inverse[E], Id]]], id[inverse[E]], inverse[FIRST]]], E]],}
\text{inverse[DUP, FIRST]], inverse[RIF], SWAP} \quad /\text{ VSTriNormality}
\]

\[
\text{Out}[55]= \text{composite[SWAP, RIF,}
\text{intersection[composite[inverse[SECOND], x], composite[inverse[FIRST],}
\text{DUP, complement[composite[inverse[E], complement[composite[FIRST,}
\text{id[inverse[E]], complement[composite[cross[E, Id], x]]]]]]]]} = \text{composite[id[cart[V, V]], intersection[x, complement[}
\text{inverse[fix[composite[complement[inverse[}
\text{fix[composite[inverse[SECOND], E, complement[composite[SECOND,}
\text{intersection[composite[x, SECOND, FIRST], composite[inverse[}
\text{FIRST], inverse[E], SECOND]]]]]]]], E, SECOND, FIRST]]]]]]}
\]

\[
\text{In}[56]:= \quad (\text{First[\%} /. \text{x \to x_}) = \text{Last[\%}] \quad /\text{ Equal} \text{ SetDelayed}
\]

The reification formula for \text{wf[x]} is derived as follows:
This is the reification rule for \texttt{wf[x]}.

\begin{verbatim}
In[58]:  reify[x_, wf[g[x]]] :=
  composite[id[cart[V, V]],
  intersection[complement[inverse[
    fix[composite[complement[inverse[fix[composite[inverse[SECOND], E,
      complement[composite[SECOND, intersection[composite[inverse[
        FIRST], inverse[E], SECOND], composite[reify[x, g[x]], SECOND,
        FIRST]]]]]]], E, SECOND, FIRST]]],
  reify[x, g[x]]]]
\end{verbatim}

variable-free formulation of \texttt{fix[trv[wf[x]]] = 0}

Here is a simple application of reification that appears to be hard to derive by other means.

\begin{verbatim}
In[59]:  Map[VERTSECT, SubstTest[reify, x, fix[trv[f[x]]], f \to \texttt{wf}]] // Reverse
Out[59]=
  composite[\texttt{IMAGE}[inverse[DUP]], \texttt{HULL}[\texttt{TRV}], \texttt{WFPART} := cart[V, singleton[0]]

In[60]:  composite[\texttt{IMAGE}[inverse[DUP]], \texttt{HULL}[\texttt{TRV}], \texttt{WFPART} := cart[V, singleton[0]]
\end{verbatim}

inclusions and intersections

\begin{verbatim}
In[61]:  Map[equal[0, #] \&, dif[\texttt{WFPART}, inverse[S]]] // VSNormality
Out[61]=
  subclass[\texttt{WFPART}, inverse[S]] = True

In[62]:  subclass[\texttt{WFPART}, inverse[S]] := True
\end{verbatim}

Corollary.

\begin{verbatim}
In[63]:  equal[intersection[\texttt{WFPART}, inverse[S]], \texttt{WFPART}]
Out[63]=
  True

In[64]:  intersection[\texttt{WFPART}, inverse[S]] := \texttt{WFPART}
\end{verbatim}
Corollary.

\[ \text{In[65]} := \text{AssInt}[S, \text{WFPART}, \text{inverse}[S]] \]
\[ \text{Out[65]} = \text{intersection}[S, \text{WFPART}] := \text{id}[\text{WF}] \]
\[ \text{In[66]} := \text{intersection}[S, \text{WFPART}] := \text{id}[\text{WF}] \]

fix formulas

\[ \text{In[67]} := \text{IminInt}[S, \text{WFPART}, V] \text{ // Reverse} \]
\[ \text{Out[67]} = \text{fix}[\text{composite}[\text{inverse}[S], \text{WFPART}]] := \text{WF} \]
\[ \text{In[68]} := \text{fix}[\text{composite}[\text{inverse}[S], \text{WFPART}]] := \text{WF} \]
\[ \text{In[69]} := \text{SubstTest}[\text{range}, \text{intersection}[w, \text{inverse}[S]], w \to \text{WFPART}] \text{ // Reverse} \]
\[ \text{Out[69]} = \text{fix}[\text{composite}[\text{WFPART}, S]] := \text{WF} \]
\[ \text{In[70]} := \text{fix}[\text{composite}[\text{WFPART}, S]] := \text{WF} \]
\[ \text{In[71]} := \text{ImageInt}[S, \text{WFPART}, V] \text{ // Reverse \ // InvertFix} \]
\[ \text{Out[71]} = \text{fix}[\text{composite}[\text{WFPART}, \text{inverse}[S]]] := \text{WF} \]
\[ \text{In[72]} := \text{fix}[\text{composite}[\text{WFPART}, \text{inverse}[S]]] := \text{WF} \]