wellorderings are transitive

Johan G. F. Belinfante
2004 March 14

In[1]:= << goedel55.09b; << tools.m;

:Package Title: goedel55.09b 2004 March 9 at 12:15 noon
It is now: 2004 Mar 15 at 12:45
Loading Simplification Rules
TOOLS.M Revised 2004 March 14
weightlimit = 40

summary

An Otter proof that well orderings are transitive was obtained 2000 December 24. In this notebook, the fact that well−orderings are transitive is rederived in a technically different fashion, but still based on the key idea that a set of three elements each of which is fixed by the well−ordering has a least element. Three free variables are introduced to represent these elements, replacing Skolem functions that appear in the Otter proof. These introduced variables are eventually eliminated by using Gödel’s algorithm, which is formulated in the GOEDEL program as a set of rewrite rules for class. If no special measures are adopted, the GOEDEL program has no way of knowing whether these variables refer to sets or to proper classes, and rather complex expressions would be encountered to cover all cases. To circumvent this problem, an important new technique is introduced in this notebook, the concept of a setpart wrapper, which permits taking advantage of conditional rewrite rules that are valid for sets only. Nothing much in the way of new rewrite rules are needed for this new concept aside from its definition and the key property that setpart[u] is always a set. The GOEDEL program apparently finds it evident from the definition that every set can be wrapped with a setpart wrapper, and is able to remove these wrappers automatically with no special effort.

normalization results

The removal of set variables requires the use of class, which makes it imperative that normalization rules are in place to prevent the hypotheses from being rewritten. Two rules are needed, the first of which is needed to preserve the hypothesis that a relation be reflexive. A slightly more general rule is derived:

In[2]:= not[subclass[x, cart[y, z]]] // AssertTest // Reverse

Out[2]= or[not[subclass[x, cart[V, z]]], not[subclass[x, cart[y, V]]]] =
      not[subclass[x, cart[y, z]]]

In[3]:= or[not[subclass[x_, cart[V, z_]]], not[subclass[x_, cart[y_, V]]]] :=
      not[subclass[x, cart[y, z]]]

The above rule affects the next one because the hypothesis that x be reflexive is a part of the definition of well−ordering.
existence of least elements

There is a simple formula for `domain[LEAST[x]]` when `x` is a wellordering:

\[
\text{implies[WELLORDER[x], equal[domain[LEAST[x]], dif[P[fix[x]], singleton[0]]]]}
\]

\[
\text{True}
\]

In this section, a version of this formula is introduced with an extra variable, which will be needed shortly. To derive this formula, the following temporary lemma is needed.

\[
\text{Map[not, SubstTest[and, implies[p1, p2],}
\text{implies[and[p2, p3], p4], not[implies[and[p1, p3], p4]], }\{p1 -> \text{WELLORDER[x]}, p2 -> \text{equal[domain[LEAST[x]], intersection[complement[singleton[0]], P[fix[x]]]],}
\text{p3 -> and[not[equal[0, y]], member[y, V], subclass[y, fix[x]]],}
\text{p4 -> not[equal[0, intersection[y, lb[x, y]]]]}]])}
\]

\[
\text{True}
\]

\[
\text{True}
\]

The following special formula for the least element of a singleton will be needed.

\[
\text{least[x, singleton[y]] // Normality}
\]

\[
\text{intersection[lb[x, singleton[y]], singleton[y]] = intersection[fix[x], singleton[y]]}
\]

\[
\text{intersection[lb[x, singleton[y]], singleton[y]] =}
\text{intersection[fix[x], singleton[y]]}
\]
**definition of the setpart wrapper**

The concept of `setpart` is useful to wrap sets to permit automatic simplifications that apply only to sets. The following membership rule defines this wrapper:

```math
In[13]:= \text{member}[x_\_, \text{setpart}[y_\_]] := \text{and}[\text{member}[x, y], \text{member}[y, V]]
```

The following normalization formula could be used to define `setpart` should one want to use this concept for automated reasoning using Otter. This normalization rule is not needed for the application in this notebook, and will not be added as a rewrite rule because doing so would cause `setpart` to be introduced in situations where it is not wanted, and one would then need to add many new rules to the GOEDEL program to eliminate it.

```math
In[14]:= \text{setpart}[x] // \text{Normality} // \text{Reverse}
Out[14]= \text{intersection}[x, \text{image}[V, \text{singleton}[x]]] = \text{setpart}[x]
```

The only important property of `setpart` is that it always be recognized to be a set when it is encountered by conditional rewrite rules.

```math
In[15]:= \text{Map}[\text{member}[\#, V] \&, \%] // \text{Reverse}
Out[15]= \text{member}[\text{setpart}[x], V] = \text{True}
```

```math
In[16]:= \text{member}[\text{setpart}[x_\_], V] := \text{True}
```

**key ideas**

The key idea is that if \( x \) is a well-ordering, then any set of three elements of \( \text{fix}[x] \) must have a least member. It is convenient to introduce these three elements one at a time to prevent rules for `pairset` from entering into the formulas that are produced. Each of the variables will be wrapped with `setpart` to make it known that it is a set. The first element to be introduced is called `setpart[u]`.

```math
In[17]:= \text{SubstTest}[\text{implies}, \text{and}[\text{WELLORDER}[x], \text{member}[w, \text{dif}[P[\text{fix}[x]], \text{singleton}[0]]], 
\text{not}[\text{empty}[\text{least}[x, w]]], w -> \text{union}[\text{singleton}[\text{setpart}[u]], z]]]
Out[17]= \text{or}[\text{not}[\text{equal}[0, \text{intersection}[z, \text{lb}[x, z], \text{lb}[x, \text{singleton}[\text{setpart}[u]]]]]], 
\text{not}[\text{member}[z, V]], \text{not}[\text{member}[\text{setpart}[u], \text{fix}[x]]], \text{not}[\text{subclass}[z, \text{fix}[x]]], 
\text{not}[\text{WELLORDER}[x]], \text{subclass}[z, \text{image}[x, \text{singleton}[\text{setpart}[u]]]]] = \text{True}
```

The second element to be introduced is called `setpart[v]`.

```math
In[18]:= \% /. z -> \text{union}[y, \text{singleton}[\text{setpart}[v]]]
Out[18]= \text{or}[\text{and}[\text{member}[\text{pair}[\text{setpart}[u], \text{setpart}[v]], x], 
\text{subclass}[y, \text{image}[x, \text{singleton}[\text{setpart}[u]]]], 
\text{and}[\text{member}[\text{pair}[\text{setpart}[v], \text{setpart}[u]], x], 
\text{subclass}[y, \text{image}[x, \text{singleton}[\text{setpart}[v]]]], 
\text{not}[\text{equal}[0, \text{intersection}[y, \text{lb}[x, y], \text{lb}[x, \text{singleton}[\text{setpart}[u]]]], 
\text{lb}[x, \text{singleton}[\text{setpart}[v]]]]]], \text{not}[\text{member}[y, V]], 
\text{not}[\text{member}[\text{setpart}[u], \text{fix}[x]]], \text{not}[\text{member}[\text{setpart}[v], \text{fix}[x]]], 
\text{not}[\text{subclass}[y, \text{fix}[x]]], \text{not}[\text{WELLORDER}[x]]] = \text{True}
```

The final variable is called `setpart[w]`. 
eliminating the set variables

To eliminate the variables, it helps to turn off the `simplify` flag.

```mathematica
In[22]:= `simplify = False;
```

The elimination of variables yields the following complicated mess, but note that `setpart` has been eliminated without having to introduce any special rewrite rules.

```mathematica
In[23]:= Map[equal[0, composite[complement[#], id[cart[V, V]]]] &,
  SubstTest[class, pair[pair[u, v], w],
    implies[WELLORDER[x], member[pair[setpart[u], setpart[v], setpart[w]], x]],
    z -> union[complement[cart[cart[fix[x], fix[x]], fix[x]]],
      composite[x, FIRST, id[x]], composite[x, SECOND, id[reverse[x]]],
      intersection[composite[reverse[x], FIRST], composite[reverse[x], SECOND]]] // Reverse]
```

The conclusion in the above statement contains three assertions, which is more than one actually needs. It suffices to cut the conclusion down to one of these three:

```mathematica
In[24]:= Map[or[#, subclass[composite[id[fix[x]], intersection[x, complement[reverse[x]]]],
  id[fix[]], complement[reverse[x]], id[fix[]]], x] &, %]
```

```mathematica
In[25]:= (% / . x -> x_) /. Equal -> SetDelayed
```
In the next section, special rules are derived to eliminate the expression `complement[inverse[x]]` that appears in this statement, effectively replacing that with the simpler expression `intersection[Di,x]`.

---

**a theorem about inverse complements of well orderings**

**Lemma.** (The fact that a well ordering $x$ is reflexive is combined with the property that any two elements of $\text{fix}[x]$ can be compared.)

```plaintext
In[26]:= Map[implies[WELLORDER[x], #] & , SubstTest[subclass, x, intersection[u, v], {u -> cart[fix[x], fix[x]], v -> union[complement[inverse[x]], Id]]] // MapNotNot
Out[26]= or[not[WELLORDER[x]], subclass[x, union[ composite[id[fix[x]], complement[inverse[x]], id[fix[x]]], id[fix[x]]]]] = True
In[27]:= %% /._ x -> x_ /. Equal -> SetDelayed
```

**Lemma.** (Equality implies inclusion.)

```plaintext
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]], {p1 -> WELLORDER[x], p2 -> equal[cart[fix[x], fix[x]], union[x, inverse[x]]], p3 -> subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]]]]
Out[28]= or[not[WELLORDER[x]], subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]]] = True
In[29]:= %% /._ x -> x_ /. Equal -> SetDelayed
```

The above two inclusions can be combined into an equation expressing a well order $x$ in terms of `complement[inverse[x]]`:

```plaintext
In[30]:= SubstTest[and, implies[p, subclass[x, y]], implies[p, subclass[y, x]], {p -> WELLORDER[x], y -> union[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]], id[fix[x]]]]] // Reverse
Out[30]= or[equal[x, union[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]], id[fix[x]]]], not[WELLORDER[x]]] = True
In[31]:= or[equal[x_, union[composite[id[fix[x_]], complement[inverse[x_]], id[fix[x_]]], id[fix[x_]]]], not[WELLORDER[x_]]] := True
```

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

```plaintext
In[32]:= simplify = True;
```

**Lemma.** (Eliminating an unneeded intersection with $\text{Di}$.)

```plaintext
In[33]:= composite[id[fix[x]], intersection[Di, complement[inverse[x]]], id[fix[x]]] // VSNormality
Out[33]= composite[id[fix[x]], intersection[Di, complement[inverse[x]]], id[fix[x]]] = composite[id[fix[x]], complement[inverse[x]], id[fix[x]]]
In[34]:= composite[id[fix[x_]], intersection[Di, complement[inverse[x_]]], id[fix[x_]]] := composite[id[fix[x]], complement[inverse[x]], id[fix[x]]]
```

**Lemma.** (The equation derived above for a well order $x$ is solved for `complement[inverse[x]]` by intersecting with $\text{Di}$.)
In[35]:= SubstTest[\text{implies, equal}[x, y], equal[image[z, x], image[z, y]],
   \{y \rightarrow \text{union[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   id[\text{fix}[x]]], z \rightarrow id[Di]\}]

Out[35]= \text{or[equal[composite][id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]], \text{not[equal}[x, \text{union[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   id[\text{fix}[x]]]]]} = \text{True}

In[36]:= (\% / . x \rightarrow x_) /. \text{Equal} \rightarrow \text{SetDelayed}

Main result: (The expression \text{complement}[\text{inverse}[x]] can be replaced with \text{intersection}[Di,x].)

In[37]:= \text{Map[not, SubstTest[and, \text{implies}[p1, p2], \text{implies}[p2, p3], \text{not[\text{implies}[p1, p3]]},
   \{p1 \rightarrow \text{WELLORDER}[x],
   p2 \rightarrow \text{equal}[x, \text{union[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]]],
   p3 \rightarrow \text{equal[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]]\}]]

Out[37]= \text{or[equal[composite][id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]], \text{not[\text{WELLORDER}[x]]]} = \text{True}

In[38]:= \text{or[equal[composite][id[\text{fix}[x_]], \text{complement}[\text{inverse}[x_]], id[\text{fix}[x_]]],
   \text{intersection}[Di, x_]], \text{not[\text{WELLORDER}[x_]]}} := \text{True}

A variant is also needed.

In[39]:= SubstTest[\text{implies, equal}[u, v], equal[image[w, u], image[w, v]],
   \{u \rightarrow \text{composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   v \rightarrow \text{intersection}[Di, x], w \rightarrow id[x]\}]

Out[39]= \text{or[equal[composite][id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]], \text{not[equal[composite][id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]]]} = \text{True}

In[40]:= (\% / . x \rightarrow x_) /. \text{Equal} \rightarrow \text{SetDelayed}

In[41]:= \text{Map[not, SubstTest[and, \text{implies}[p1, p2], \text{implies}[p2, p3], \text{not[\text{implies}[p1, p3]]},
   \{p1 \rightarrow \text{WELLORDER}[x], p2 \rightarrow \text{equal[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   p3 \rightarrow \text{equal[composite}[id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]],
   id[\text{fix}[x]], \text{intersection}[Di, x]]\}]]

Out[41]= \text{or[equal[composite][id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]], \text{not[\text{WELLORDER}[x]]}} = \text{True}

In[42]:= (\% / . x \rightarrow x_) /. \text{Equal} \rightarrow \text{SetDelayed}

Equality substitution for composites is used:

In[43]:= SubstTest[\text{implies, and[equal}[u, v], equal[w, z]],
   \text{equal[composite}[u, w, \text{composite}[v, z]]],
   \{u \rightarrow \text{composite}[id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]], id[\text{fix}[x]]],
   w \rightarrow \text{composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   v \rightarrow \text{intersection}[Di, x], z \rightarrow \text{intersection}[Di, x]\}]

Out[43]= \text{or[equal[composite][\text{intersection}[Di, x], \text{intersection}[Di, x]],
   \text{composite}[id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]],
   id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]], \text{not[equal[composite}[id[\text{fix}[x]], \text{complement}[\text{inverse}[x]], id[\text{fix}[x]]],
   \text{intersection}[Di, x]], \text{not[equal[composite][id[\text{fix}[x]], \text{intersection}[x, \text{complement}[\text{inverse}[x]]],
   id[\text{fix}[x]]], \text{intersection}[Di, x]]]} = \text{True}
SubstTest[implies, and[equal[u, v], subclass[v, w]], subclass[u, w], 
{u \rightarrow \text{composite}[\text{intersection}[\text{Di}, x], \text{intersection}[\text{Di}, x]],
v \rightarrow \text{composite}[\text{id}[\text{fix}[x]], \text{intersection}[x, \text{complement}[^\text{inverse}[x]]],}
\text{id}[\text{fix}[x]], \text{complement}[^\text{inverse}[x]], \text{id}[\text{fix}[x]], w \rightarrow x}]

\text{In}\[45\]= subclass[\text{composite}[x, x], x]

Lemma.

\text{In}\[46\]= \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[\text{p1}, \text{p2}], 
\text{implies}[\text{p1}, \text{p4}], \text{implies}[\text{and}[\text{p3}, \text{p4}], \text{p5}], \text{implies}[\text{and}[\text{p2}, \text{p5}], \text{p6}], 
\text{not}[\text{implies}[\text{p1}, \text{p6}], 
\{\text{p1} \rightarrow \text{WELLORDER}[x],
\text{p2} \rightarrow \text{subclass}[\text{composite}[\text{id}[\text{fix}[x]], \text{intersection}[x, \text{complement}[^\text{inverse}[x]]],}
\text{id}[\text{fix}[x]], \text{complement}[^\text{inverse}[x]], \text{id}[\text{fix}[x]], x],
\text{p3} \rightarrow \text{equal}[\text{composite}[\text{id}[\text{fix}[x]], \text{complement}[^\text{inverse}[x]], \text{id}[\text{fix}[x]], \text{intersection}[\text{Di}, x]],
\text{p4} \rightarrow \text{equal}[\text{composite}[\text{id}[\text{fix}[x]], \text{intersection}[x, \text{complement}[^\text{inverse}[x]]],}
\text{id}[\text{fix}[x]], \text{complement}[^\text{inverse}[x]], \text{id}[\text{fix}[x]], \text{intersection}[\text{Di}, x], 
\text{p5} \rightarrow \text{equal}[\text{composite}[\text{id}[\text{fix}[x]], \text{intersection}[x, \text{complement}[^\text{inverse}[x]]],}
\text{id}[\text{fix}[x]], \text{complement}[^\text{inverse}[x]], \text{id}[\text{fix}[x]], \text{intersection}[\text{Di}, x], 
\text{p6} \rightarrow \text{subclass}[\text{composite}[x, x], x]]}
\text{Out}\[48\]= \text{or}[\text{not}[\text{WELLORDER}[x]], \text{subclass}[\text{composite}[x, x], x]] = True

\text{In}\[49\]= \text{or}[\text{not}[\text{WELLORDER}[\text{x\_}]], \text{subclass}[\text{composite}[\text{x\_}, \text{x\_}, \text{x\_}]]] := True

Corollary:

\text{In}\[50\]= \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[\text{p1}, \text{p2}], 
\text{implies}[\text{p1}, \text{p3}], \text{implies}[\text{and}[\text{p2}, \text{p3}], \text{p4}], \text{not}[\text{implies}[\text{p1}, \text{p4}], 
\{\text{p1} \rightarrow \text{WELLORDER}[x], \text{p2} \rightarrow \text{subclass}[x, \text{cart}[V, V]],
\text{p3} \rightarrow \text{subclass}[\text{composite}[x, x], x], \text{p4} \rightarrow \text{TRANSITIVE}[x]]}
\text{Out}\[50\]= \text{or}[\text{not}[\text{WELLORDER}[x]], \text{TRANSITIVE}[x]] = True

\text{In}\[51\]= \text{or}[\text{not}[\text{WELLORDER}[\text{x\_}]], \text{TRANSITIVE}[\text{x\_}]] := True

A variable–free version of this can be derived for the special case of small well–orderings:
Not all well orderings are sets. For example, the restriction of the subclass relation $S$ to the class $\Omega$ of all ordinals is a well ordering which is a proper class.

In this final section, the results obtained above are combined with other previously derived facts about well orderings to derive various useful corollaries.

If $x$ is a well ordering, then $\text{intersection}[D_i,x]$ is acyclic.

Well orderings are total orderings.