strictly increasing sequences of ordinals

Johan G. F. Belinfante
2011 February 15

```
In[1]:= SetDirectory["l:"]; << goedel.11feb14a

        :Package Title: goedel.11feb14a                    2011 February 14 at 9:20 a.m.

        It is now:  2011 Feb 15 at 20:3

        Loading Simplification Rules

        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

        weightlimit = 40
```

summary

A sequence $\mathbf{x}: \omega \to \Omega$ of ordinals is **strictly increasing** if $\mathbf{x_i \in x_{i+1}}$ for all $\mathbf{i} \in \omega$. It is shown in this notebook that the union of such a sequence is a limit ordinal. That is, **U[range[x]] ∈ Ω ⋂ fix[BIGCUP]**.

a comment on terminology

An infinite sequence $\mathbf{x}$ of objects may be regarded as a function from the set ω of natural numbers to a class of objects. It is customary to write $\mathbf{x_i}$ for **APPLY[x, i]**. Common parlance about sequences unfortunately does not very carefully distinguish between the function $\mathbf{x}$ and its range. In particular, the union of a sequence $\mathbf{x}$ of ordinals, often denoted $\bigcup_{\mathbf{i} \in \omega} \mathbf{x_i}$, refers to **U[range[x]]** and not to **U[x]**.

sequences in general

Theorem. If $\mathbf{x}: \omega \to \mathbf{y}$, then $\mathbf{x_i} \in \mathbf{y}$ for all $\mathbf{i} \in \omega$.

```
In[2]:= SubstTest[implies, and[member[x, map[w, y]], member[t, w]],
          member[APPLY[x, t], y], {t → nat[i], w → omega}] // Reverse

Out[2]= or[member[APPLY[x, nat[i]], y], not[member[x, map[omega, y]]]] == True

In[3]:= or[member[APPLY[x_, nat[i_]], y_], not[member[x_, map[omega, y_]]]] := True
```

Theorem. A general result about mappings.

```
In[4]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[and[p0, p2, p3], p4], not[implies[and[p0, p1], p4]],
           {p0 -> member[t, y], p1 -> member[x, map[y, z]], p2 → FUNCTION[x],
            p3 → equal[domain[x], y], p4 -> member[APPLY[x, t], range[x]]}]] // Reverse

Out[4]= or[member[APPLY[x, t], range[x]], not[member[t, y]], not[member[x, map[y, z]]]] == True

In[5]:= or[member[APPLY[x_, t_], range[x_]],
          not[member[t_, y_]], not[member[x_, map[y_, z_]]]] := True
```

Corollary. Specialization to sequences.

```
In[6]:= SubstTest[implies, and[member[x, map[w, y]], member[z, w]],
          member[APPLY[x, z], range[x]], {w → omega, z → nat[t]}] // Reverse

Out[6]= or[member[APPLY[x, nat[t]], range[x]], not[member[x, map[omega, y]]]] == True

In[7]:= or[member[APPLY[x_, nat[t_]], range[x_]], not[member[x_, map[omega, y_]]]] := True
```

Corollary. The same as the preceding, except that **nat[t]** is replaced with its successor.

```
In[8]:= SubstTest[implies, member[x, map[omega, y]],
          member[APPLY[x, nat[z]], range[x]], z → succ[nat[t]]] // Reverse

Out[8]= or[member[APPLY[x, succ[nat[t]]], range[x]], not[member[x, map[omega, y]]]] == True

In[9]:= or[member[APPLY[x_, succ[nat[t_]]], range[x_]], not[member[x_, map[omega, y_]]]] := True
```

---

sequences of ordinals

Theorem. The union of any sequence of ordinals is an ordinal.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 → member[x, map[omega, OMEGA]], p2 → subclass[range[x], OMEGA],
             p3 → member[range[x], V], p4 → member[U[range[x]], OMEGA]}]] // Reverse

Out[10]= or[member[U[range[x]], OMEGA], not[member[x, map[omega, OMEGA]]]] == True

In[11]:= or[member[U[range[x_]], OMEGA], not[member[x_, map[omega, OMEGA]]]] := True
```

Corollary. A variable-free restatement of the preceding theorem.

```
In[12]:= Map[equal[V, #] &, SubstTest[class, x,
           implies[member[x, y], member[U[range[x]], z]], {y → map[omega, OMEGA], z → OMEGA}]]

Out[12]= subclass[image[BIGCUP, image[IMAGE[SECOND], map[omega, OMEGA]]], OMEGA] == True

In[13]:= subclass[image[BIGCUP, image[IMAGE[SECOND], map[omega, OMEGA]]], OMEGA] := True
```

strictly increasing sequences

In this section it is shown that a sequence **x** of ordinals is strictly increasing if **x ∘ SUCC ⊂ E ∘ x**. A **funpart** wrapper is often helpful when deriving facts about function application. The first lemma introduces a **funpart** wrapper for this purpose.

Lemma.

```
In[14]:= SubstTest[implies, and[member[t, u], subclass[u, v]],
           member[t, v], {t → pair[nat[i], APPLY[funpart[x], succ[nat[i]]]],
            u → composite[funpart[x], SUCC], v → composite[E, funpart[x]]}] // Reverse

Out[14]= or[member[nat[i], image[inverse[funpart[x]], APPLY[funpart[x], succ[nat[i]]]]],
           not[member[succ[nat[i]], domain[funpart[x]]]],
           not[subclass[composite[funpart[x], SUCC, inverse[funpart[x]]], E]],
           not[subclass[image[inverse[SUCC], domain[funpart[x]]], domain[funpart[x]]]] == True

In[15]:= (% /. {i → i_, x → x_}) /. Equal → SetDelayed
```

Some of the complexity of the preceding lemma is eliminated by specializing to the case that **domain[x] = *ω*,** and eliminating the **funpart** wrapper.

Lemma.

```
In[16]:= SubstTest[implies, and[equal[x, funpart[t]], equal[w, domain[x]]],
           or[member[nat[i], image[inverse[x], APPLY[x, succ[nat[i]]]]],
            not[member[succ[nat[i]], w]], not[subclass[composite[x, SUCC], composite[E, x]]],
            not[subclass[image[inverse[SUCC], w], w]]], {t → x, w → omega}] // Reverse

Out[16]= or[member[nat[i], image[inverse[x], APPLY[x, succ[nat[i]]]]],
           not[equal[omega, domain[x]]], not[FUNCTION[x]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[17]:= (% /. {i → i_, x → x_}) /. Equal → SetDelayed
```

Theorem. If **x: *ω* → Ω** satisfies **x ∘ SUCC ⊂ E ∘ x**, then **$x_i \in x_{i+1}$** for all **i ∈ *ω***.

```
In[18]:= Map[not, SubstTest[and, implies[p1, p3], implies[p1, p4],
             implies[and[p2, p3, p4], p5], implies[and[p3, p5], p6],
             not[implies[and[p1, p2], p6]], {p1 → member[x, map[omega, OMEGA]],
              p2 -> subclass[composite[x, SUCC], composite[E, x]],
              p3 → FUNCTION[x], p4 → equal[domain[x], omega],
              p5 -> member[nat[i], image[inverse[x], APPLY[x, succ[nat[i]]]]],
              p6 -> member[APPLY[x, nat[i]], APPLY[x, succ[nat[i]]]]}]] // Reverse

Out[18]= or[member[APPLY[x, nat[i]], APPLY[x, succ[nat[i]]]], not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[19]:= or[member[APPLY[x_, nat[i_]], APPLY[x_, succ[nat[i_]]]],
           not[member[x_, map[omega, OMEGA]]],
           not[subclass[composite[x_, SUCC], composite[E, x_]]]] := True
```

main theorem

In this section it is shown that the union of a strictly increasing sequence of ordinals is a limit ordinal. That is, if **x: $\omega \to \Omega$** satisfies **$x \circ SUCC \subset E \circ x$**, then **U[range[x]]** is a limit ordinal.

Lemma.

```
In[20]:= Map[not,
           SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[and[p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → member[x, map[omega, OMEGA]],
             p2 -> subclass[composite[x, SUCC], composite[E, x]],
             p3 -> member[APPLY[x, nat[i]], APPLY[x, succ[nat[i]]]],
             p4 -> member[APPLY[x, succ[nat[i]]], range[x]],
             p5 → member[APPLY[x, nat[i]], U[range[x]]]}]] // Reverse

Out[20]= or[member[APPLY[x, nat[i]], U[range[x]]], not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[21]:= (% /. {i → i_, x → x_}) /. Equal → SetDelayed
```

The next step is to eliminate the variable **i**.

Lemma. (Eliminating **APPLY** in favor of an inverse image.)

```
In[22]:= ((Map[implies[and[equal[x, funpart[t]], member[x, map[omega, OMEGA]],
                 subclass[composite[x, SUCC], composite[E, x]]], #] &,
              (member[j, image[inverse[funpart[t]], u]] // AssertTest) /.
               {j → nat[i], u → U[range[x]]}]) /. t → x) // MapNotNot

Out[22]= or[member[nat[i], image[inverse[funpart[x]], U[range[x]]]],
           not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[23]:= (% /. {i → i_, x → x_}) /. Equal → SetDelayed
```

Lemma. (Eliminating the **nat** and **funpart** wrappers.)

```
In[24]:= SubstTest[implies, and[equal[i, nat[j]], equal[t, funpart[x]]],
            or[member[i, image[inverse[t], U[range[x]]]], not[member[x, map[omega, OMEGA]]],
             not[subclass[composite[x, SUCC], composite[E, x]]]],
            {t → x, j → i}] // Reverse // MapNotNot

Out[24]= or[member[i, image[inverse[x], U[range[x]]]],
           not[member[i, omega]], not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[25]:= (% /. {i → i_, x → x_}) /. Equal → SetDelayed
```

Lemma. A simplification rule.

```
In[26]:= subclass[composite[x, SUCC], composite[E, x]] // AssertTest // Reverse

Out[26]= equal[0, fix[composite[SUCC, LB[complement[x]], x]]] ==
          subclass[composite[x, SUCC], composite[E, x]]

In[27]:= equal[0, fix[composite[SUCC, LB[complement[x_]], x_]]] :=
          subclass[composite[x, SUCC], composite[E, x]]
```

Theorem. (Eliminating the variable **i**.)

```
In[28]:= Map[equal[V, #] &, SubstTest[class, i,
           or[member[i, u], not[member[i, omega]], not[member[x, v]], not[subclass[y, z]]],
           {u → image[inverse[x], U[range[x]]], v → map[omega, OMEGA],
            y -> composite[x, SUCC], z -> composite[E, x]}]]

Out[28]= or[not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]],
           subclass[omega, image[inverse[x], U[range[x]]]]] == True

In[29]:= (% /. x → x_) /. Equal → SetDelayed
```

Main Theorem. If $\mathbf{x}: \omega \to \Omega$ satisfies $\mathbf{x} \circ \mathbf{SUCC} \subset \mathbf{E} \circ \mathbf{x}$, then $\mathbf{range[x]} \subset \mathbf{U[range[x]]}$.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p3], implies[p1, p4],
            implies[and[p1, p2], p5], implies[and[p3, p5], p6], implies[and[p4, p6], p7],
            not[implies[and[p1, p2], p7]], {p1 -> member[x, map[omega, OMEGA]],
             p2 -> subclass[composite[x, SUCC], composite[E, x]], p3 → equal[domain[x], omega],
             p4 → FUNCTION[x], p5 -> subclass[omega, image[inverse[x], U[range[x]]]],
             p6 → subclass[domain[x], image[inverse[x], U[range[x]]]],
             p7 → subclass[range[x], U[range[x]]]}]] // Reverse

Out[30]= or[not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]],
           subclass[range[x], U[range[x]]]] == True

In[31]:= or[not[member[x_, map[omega, OMEGA]]],
           not[subclass[composite[x_, SUCC], composite[E, x_]]],
           subclass[range[x_], U[range[x_]]]] := True
```

Corollary. The union of a strictly increasing sequence of ordinals is a limit ordinal.

```
In[32]:= Map[not,
           SubstTest[and, implies[p1, p3], implies[and[p1, p2], p4], implies[and[p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 -> member[x, map[omega, OMEGA]],
             p2 -> subclass[composite[x, SUCC], composite[E, x]],
             p3 → subclass[range[x], OMEGA], p4 -> subclass[range[x], U[range[x]]],
             p5 → equal[U[U[range[x]]], U[range[x]]]}] // Reverse

Out[32]= or[equal[U[range[x]], U[U[range[x]]]], not[member[x, map[omega, OMEGA]]],
           not[subclass[composite[x, SUCC], composite[E, x]]]] == True

In[33]:= or[equal[U[range[x_]], U[U[range[x_]]]], not[member[x_, map[omega, OMEGA]]],
           not[subclass[composite[x_, SUCC], composite[E, x_]]]] := True
```