HULL[fix[ACLOSURE]]

Johan G. F. Belinfante
2002 January 24

<<goedel52.m66; <<tests.m

:Package Title: GOEDEL52.M66 2002 January 22 at 8:15 p.m.

It is now: 2002 Jan 23 at 23:48

Loading Simplification Rules

TESTS.M Revised 2002 January 12

weightlimit = 40

Context switch to 'Goedel'Private is needed for ReplaceTest

Just ignore the error message about Unterminated use of BeginPackage

Get::*bebal: Unterminated uses of BeginPackage or Begin in <<tests.m.

■ Introduction

In this notebook it is proved that HULL[fix[ACLOSURE]] = ACLOSURE. This is analogous to a previously established formula for UCLOSURE. For that case the proof used the idempotence of the function symbol Uclosure[x]. In the proof that Uclosure is idempotent, a choice function was used, whose construction depends crucially on the fact that the inverse of the function BIGCUP is a thin relation. Unfortunately this argument does not carry over to the case of Aclosure[x] because the inverse of BIGCAP is not thin. It is not known at present whether the function symbol Aclosure is idempotent or not. The function ACLOSURE, however, is known to be idempotent, from which it is shown below that Aclosure is idempotent for the special case of sets. It turns out that this special case suffices for the proof of the formula for HULL[fix-ACLOSURE]].

■ Idempotence of Aclosure for the special case of sets.

A useful membership rule:

\[
\text{member[pair}[x, y], \text{composite}[z, \text{ACLOSURE}]] \quad \text{// AssertTest}
\]

\[
\text{member[pair}[x, y], \text{composite}[z, \text{ACLOSURE}]] =
\quad \text{and}[\text{member}[x, V], \text{member}[y, V], \text{member}[\text{pair}[\text{Aclosure}[x], y], z]]
\]

\[
\text{member[pair}[x, y], \text{composite}[z, \text{ACLOSURE}]] :=
\quad \text{and}[\text{member}[x, V], \text{member}[y, V], \text{member}[\text{pair}[\text{Aclosure}[x], y], z]]
\]

The idempotence of Aclosure for sets is:
Map[implies[member[x, V], #]] &, SubstTest[member, pair[x, y], composite[z, ACLOSURE], {y -> Aclosure[x], z -> ACLOSURE}]] // Reverse

or[equal[Aclosure[x], Aclosure[Aclosure[x]]], not[member[x, V]]] == True

or[equal[Aclosure[x_], Aclosure[Aclosure[x_]]], not[member[x_, V]]] := True

This result is weaker than the corresponding result for Uclos u r e; in that case one does not need the hypothesis member[x, V]. The following corollary is needed later.

Map[not, SubstTest[and, p, implies[p, q], {p -> member[x, V], q -> equal[Aclosure[x], Aclosure[Aclosure[x]]]]]] // Reverse

or[not[equal[Aclosure[x], Aclosure[Aclosure[x]]]], not[member[x, V]]] == not[member[x, V]]

or[not[equal[Aclosure[x_], Aclosure[Aclosure[x_]]]], not[member[x_, V]]] := not[member[x, V]]

hereditary closure lemma

Map[class[x, #]] &, SubstTest[implies, and[subclass[x, y], member[y, z], member[x, image[inverse[S], z]], {y -> Aclosure[x], z -> fix[ACLOSURE]]}]

image[inverse[S], fix[ACLOSURE]] == V

image[inverse[S], fix[ACLOSURE]] := V

one direction

SubstTest[implies, member[u, v], subclass[A[v], u], {u -> Aclosure[x], v -> intersection[fix[ACLOSURE], image[S, singleton[x]]]}]

or[not[member[x, V]]], subclass[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x]]] == True

or[not[member[x_, V]]], subclass[A[intersection[y, image[S, singleton[x]]]], Aclosure[x]]], y -> fix[ACLOSURE]] // Reverse

fix[composite[inverse[ACLOSURE], S, HULL[fix[ACLOSURE]]]] == V

fix[composite[inverse[ACLOSURE], S, HULL[fix[ACLOSURE]]]] := V

monotonicity of ACLOSURE

union[composite[inverse[ACLOSURE], S, ACLOSURE], complement[S]] // Renormality

union[complement[S], composite[inverse[ACLOSURE], S, ACLOSURE]] == V
union[complement[S], composite[inverse[ACLOSURE], S, ACLOSURE]] := V

SubstTest[equal, V, union[complement[x], y],
{x -> S, y -> composite[inverse[ACLOSURE], S, ACLOSURE]]] // Reverse
subclass[S, composite[inverse[ACLOSURE], S, ACLOSURE]] == True

subclass[S, composite[inverse[ACLOSURE], S, ACLOSURE]] := True

Various corollaries can be obtained:

SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
{u -> S, v -> composite[inverse[ACLOSURE], S, ACLOSURE], w -> ACLOSURE}]
subclass[composite[ACLOSURE, S], composite[S, ACLOSURE]] == True

subclass[composite[ACLOSURE, S], composite[S, ACLOSURE]] := True

SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
{u -> S, v -> composite[inverse[ACLOSURE], S, ACLOSURE], w -> inverse[ACLOSURE]}]
subclass[composite[S, inverse[ACLOSURE]], composite[inverse[ACLOSURE], S]] == True

subclass[composite[S, inverse[ACLOSURE]], composite[inverse[ACLOSURE], S]] := True

SubstTest[implies, subclass[u, v], subclass[composite[u, z], composite[v, z]],
{u -> composite[ACLOSURE, S], v -> composite[S, ACLOSURE], z -> ACLOSURE}]
subclass[composite[ACLOSURE, S, ACLOSURE], composite[S, ACLOSURE]] == True

subclass[composite[ACLOSURE, S, ACLOSURE], composite[S, ACLOSURE]] := True

**going in the other direction**

ImageComp[ACLOSURE, inverse[ACLOSURE], x] // Reverse
image[ACLOSURE, image[inverse[ACLOSURE], x]] == intersection[x, fix[ACLOSURE]]

image[ACLOSURE, image[inverse[ACLOSURE], x_]] := intersection[x, fix[ACLOSURE]]

Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
not[implies[and[p1, p2], p3]],
{p1 -> member[y, fix[ACLOSURE]], p2 -> subclass[x, y], p3 -> equal[Aclosure[y], y], p4 -> subclass[Aclosure[x], Aclosure[y]], p5 -> subclass[Aclosure[x], y]]}
or[not[equal[y, Aclosure[y]]], not[member[y, V]],
not[subclass[x, y]], subclass[Aclosure[x], y]] == True

or[not[equal[y_, Aclosure[y_]]], not[member[y_, V]],
not[subclass[x_, y_]], subclass[Aclosure[x_], y_]] := True

implies[member[y, intersection[fix[ACLOSURE], image[S, singleton[x]]]],
subclass[Aclosure[x], y]]

True
SubstTest[class, y, implies[member[y, z], subclass[w, y]],
    {z -> intersection[fix[ACLOSURE], image[S, singleton[x]]], w -> Aclosure[x]}] // Reverse
union[complement[fix[ACLOSURE]],
    complement[image[S, singleton[x]]], image[S, singleton[Aclosure[x]]]] == V
union[complement[fix[ACLOSURE]],
    complement[image[S, singleton[x_]]], image[S, singleton[Aclosure[x_]]]] := V
SubstTest[equal, V, union[complement[u], v],
    {u -> intersection[fix[ACLOSURE], image[S, singleton[x]]],
    v -> image[S, singleton[Aclosure[x]]]}] // Reverse
subclass[Aclosure[x_], A[intersection[fix[ACLOSURE], image[S, singleton[x]]]]] == True
subclass[Aclosure[x_], A[intersection[fix[ACLOSURE], image[S, singleton[x_]]]]] := True
Map[impliessubclass[
    A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x], #] &,
    SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> A[intersection[fix[ACLOSURE], image[S, singleton[x]]]],
    v -> Aclosure[x]}]] // Reverse
or[equal[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x]],
    not[subclass[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x]]]] == True
or[equal[A[intersection[fix[ACLOSURE], image[S, singleton[x_]]]], Aclosure[x_]],
    not[subclass[A[intersection[fix[ACLOSURE], image[S, singleton[x_]]]], Aclosure[x_]]]] := True
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p4],
    not[implies[p1, p4]],
    {p1 -> member[x, V],
    p2 -> subclass[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x]],
    p3 -> subclass[Aclosure[x],
    A[intersection[fix[ACLOSURE], image[S, singleton[x]]]]],
    p4 -> equal[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]],
    Aclosure[x]]}]]
or[equal[A[intersection[fix[ACLOSURE], image[S, singleton[x]]]], Aclosure[x]],
    not[member[x, V]]] == True
or[equal[A[intersection[fix[ACLOSURE], image[S, singleton[x_]]]], Aclosure[x_]],
    not[member[x_, V]]] := True

**final steps**

implies[member[x, V],
    member[x, fix[composite[inverse[ACLOSURE], HULL[fix[ACLOSURE]]]]]] // AssertTest
or[member[pair[x, x], composite[inverse[ACLOSURE], HULL[fix[ACLOSURE]]]],
    not[member[x, V]]] == True
or[member[pair[x_, x_], composite[inverse[ACLOSURE], HULL[fix[ACLOSURE]]]],
    not[member[x_, V]]] := True
fix[composite[inverse[ACLOSURE], HULL[fix[ACLOSURE]]]] // Renormality
fix[composite[inverse[ACLOSURE], HULL[fix[ACLOSURE]]]] == V
fix[composite[inverse[ACLOSEURE], HULL[fix[ACLOSEURE]]]] := V

SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
   {u -> Id, v -> composite[inverse[ACLOSEURE], HULL[fix[ACLOSEURE]]], w -> ACLOSEURE}]
subclass[ACLOSEURE, HULL[fix[ACLOSEURE]]] == True

subclass[ACLOSEURE, HULL[fix[ACLOSEURE]]] := True

SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
   {u -> Id, v -> composite[inverse[ACLOSEURE], HULL[fix[ACLOSEURE]]],
   w -> inverse[HULL[fix[ACLOSEURE]]]]]
subclass[HULL[fix[ACLOSEURE]], ACLOSEURE] == True

subclass[HULL[fix[ACLOSEURE]], ACLOSEURE] := True

SubstTest[and, subclass[u, v], subclass[v, u],
   {u -> HULL[fix[ACLOSEURE]], v -> ACLOSEURE} // Reverse
equal[ACLOSEURE, HULL[fix[ACLOSEURE]]] == True

HULL[fix[ACLOSEURE]] := ACLOSEURE