trv[x]

Johan G. F. Belinfante
2002 June 8

```
<< goedel52.o37; << tools.m
```

```
:Package Title: goedel52.o37            2002 June 8 at 12:05 noon

It is now:  2002 Jun 8 at 18:18

Loading Simplification Rules

TOOLS.M                    Revised 2002 June 8

weightlimit = 40
```

■ Introduction

This notebook is devoted to deriving wo basic rules for **trv[x]** that had been overlooked before. These rules are useful in establishing the connection between iteration and transitive closure.

■ transitive relations are their own transitive closures

The **GOEDEL** program already has a rule about transitive closures of transitive relations:

```
implies[subclass[composite[x, x], x], equal[trv[x], composite[Id, x]]]
```

```
True
```

For applications, this rule is admittedly not as convenient as it ought to be. For this reason it is incumbent on us to add another rule to make this connection more transparent. We begin with a variant of the transitive law of equality.

```
SubstTest[implies, and[equal[x, z], equal[z, y]], equal[x, y], z -> composite[Id, x]]
```

```
or[equal[x, y], not[equal[y, composite[Id, x]]], not[subclass[x, cart[V, V]]]] == True
```

```
or[equal[x_, y_], not[equal[y_, composite[Id, x_]]],
  not[subclass[x_, cart[V, V]]]] := True
```

We can now derive a more convenient rule:

```
Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
        not[implies[and[p1, p2], p4]],
        {p1 -> subclass[composite[x, x], x],
    p2 -> equal[x, composite[Id, x]],
    p3 -> equal[composite[Id, x], trv[x]],
    p4 -> equal[x, trv[x]]}]]
```

```
or[equal[x, trv[x]], not[subclass[x, cart[V, V]]],
  not[subclass[composite[x, x], x]]] == True
```

```
or[equal[x_, trv[x_]], not[subclass[x_, cart[V, V]]],
  not[subclass[composite[x_, x_], x_]]] := True
```

With this in place, we can now write:

```
implies[TRANSITIVE[x], equal[x, trv[x]]]
```

```
True
```

■ from transitive to idempotent

The transitivity condition is an inclusion rather than an equality. From a any transitive relation, one can get an idempotent relation by forming the union with an identity relation. But the **GOEDEL** program needs an additional rule so that this fact can be recognized. The needed fact is already known:

```
equal[union[composite[trv[x], trv[x]], trv[x]], trv[x]]
```

```
True
```

The corresponding rewrite rule is missing, however, and needs to be added. One can discover what this rule should look like by replacing **equal** by **Equal**:

```
Equal[union[composite[trv[x], trv[x]], trv[x]], trv[x]]
```

```
union[composite[trv[x], trv[x]], trv[x]] == trv[x]
```

The rewrite rule is now added:

```
union[composite[trv[x_], trv[x_]], trv[x_]] := trv[x]
```

With this rule in place, the **GOEDEL** program can recognize the idempotence property in question:

```
equal[composite[u, u], u] /. u -> union[Id, trv[x]]
```

```
True
```