Detailed Catalogue of Canonical Term Rewriting Systems Generated Automatically

Wolfgang Gehrke

November 1994

Research Institute for Symbolic Computation
Johannes Kepler University
A - 4040 Linz, AUSTRIA
$wgehrke@risc.uni - linz.ac.at$

Abstract

In this paper we give a detailed catalogue for several canonical rewrite systems which have been compiled by Hullot in 1980. All the definitions are given precisely and a trace of a completion system (RRL) for those equational theories is also included. In some examples the canonical system generated by RRL differs from the system presented by Hullot since there is no unique canonical system. If no canonical system exists RRL showed unpredictable behavior.

Introduction

In 1980 Hullot compiled a catalogue of canonical term rewriting systems, cf. [Hul80], for various interesting and important equational theories. He used a system implemented in VLISP but not further specified in his article for this purpose. This contribution provided an important benchmark and ground work of that area.

In this paper we continue on his work mainly by filling in the details. The system used here was RRL ("Rewrite Rule Laboratory") which is described in [KZ89]. The details filled in are as follows:

- Precise definitions together with examples are given.
- The canonical term rewriting systems are always summarized in an extra subsection.
- A detailed trace for every example is included.

We believe that it is important to have such a detailed trace for further progress in the area.

RRL produced in some cases a different system. This is due to the fact that there is no unique canonical system. RRL uses a standard method for orienting the equations namely the recursive path ordering. We wanted to achieve a mainly automatic computation of the canonical system, i.e. with as little user interaction as possible. Thus the outcome differed.

RRL worked well in all cases where a canonical system exists. But in the other cases the behavior was unpredictable:

- reported inability,
- infinite loop, or
- wrong result.

But still it was possible to detect a wrong result by checking the rewriting system produced by RRL against the axioms given. Always not all the axioms were derivable any more.

The theories that have been studied here and also in [Hul80] are as follows:

1. groupoids
2. quasigroups and loops
3. semigroups and monoids
4. groups
5. (l,r) systems
6. commutative semigroups and monoids
7. Abelian groups
8. rings
9. anticommutative rings and Lie rings
10. A-modules
11. A-bimodules
12. A-rings
13. A-algebras
14. lattices
15. distributive lattices
16. Boolean algebras
17. R. Milner's theory of nondeterministic machines
18. arithmetic theories
19. R. J. Popplestone's theory of robot's movements

The structure of the paper is as follows: All the theories are introduced in an own section. Each section consists of the 4 subsections axioms, example, canonical system, and trace.

Finally we want to point out that we do not give a theoretical outline of the methods used. So we assume the reader to be familiar with the Knuth-Bendix algorithm and its extensions to associative/commutative rewritings. For getting an overview we refer to [Buc83, Che86, Hue80].

1 Groupoids

1.1 Axioms

Definition 1.1 *A groupoid GP is a set S with a single binary operation* $\cdot$, *i.e.* $\cdot : S \times S \to S$.

Let x, y, z be arbitrary but fixed variables of type S.

Definition 1.2 *A central groupoid CG is a groupoid with the following axiom:*

$$(x \cdot y) \cdot (y \cdot z) = y \tag{1}$$

1.2 Example

A simple but uninteresting model is a groupoid with just one element and the constant $\cdot$ operation mapping to that element. At least this demonstrates that the theory is not contradictory.

1.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S. Then there is the following canonical system for central groupoids:

$$\begin{aligned} (x \cdot y) \cdot (y \cdot z) &\longrightarrow y & (1)\\ x \cdot ((x \cdot y) \cdot z) &\longrightarrow x \cdot y & (2)\\ (x \cdot (y \cdot z)) \cdot z &\longrightarrow y \cdot z & (3) \end{aligned}$$

1.4 Trace

The operation $\cdot$ is represented by $*$.

```
RRL-> read test/1.eq
Equations read in are:
 1. ((X * Y) * (Y * Z)) == Y  [USER, 1]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] ((X * Y) * (Y * Z)) ---> Y  [USER, 1]

Computing critical pairs with:
    [1] ((X * Y) * (Y * Z)) ---> Y  [USER, 1]

Adding Rule:   [2] (Y * ((Y * Z) * Z1)) ---> (Y * Z)  [1, 1]
```

```
Adding Rule:   [3] ((X * (X1 * Y)) * Y) ---> (X1 * Y)  [1, 1]

Computing critical pairs with:
    [3] ((X * (X1 * Y)) * Y) ---> (X1 * Y)  [1, 1]

Computing critical pairs with:
    [2] (Y * ((Y * Z) * Z1)) ---> (Y * Z)  [1, 1]

Your system is canonical.

  [1] ((X * Y) * (Y * Z)) ---> Y  [USER, 1]
  [2] (Y * ((Y * Z) * Z1)) ---> (Y * Z)  [1, 1]
  [3] ((X * (X1 * Y)) * Y) ---> (X1 * Y)  [1, 1]

Time used                          =   0.13 sec
Number of rules generated          = 3
Number of rules retained           = 3
Number of critical pairs           = 18
Time spent in normalization        =   0.01 sec  (7.0 percent of time)
Time spent in unification          =   0.04 sec   (30.0 percent of time)
Time spent in ordering             =   0.02 sec   (15.0 percent of time)
Time spent in simplifying the rules =   0.01 sec   (7.0 percent of time)
Total time (including 'undo' action) = 15.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

2 Quasigroups and Loops

2.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 2.1 *A quasigroup QG is a groupoid having additionally two binary operations* $\backslash$ *and* $/$, *i.e.* $\backslash : S \times S \to S$ *and* $/ : S \times S \to S$ *fulfilling the following axioms:*

$$x \cdot (x \backslash y) = y \tag{1}$$
$$(x/y) \cdot y = x \tag{2}$$
$$x \backslash (x \cdot y) = y \tag{3}$$
$$(x \cdot y)/y = x \tag{4}$$

Definition 2.2 *An abtract loop AL is a quasigroup with identity e, i.e. there hold addtionally the following axioms:*

$$e \cdot x = x \tag{5}$$
$$x \cdot e = x \tag{6}$$

Definition 2.3 *An idempotent quasigroup IQ is a quasigroup with the additional property:*

$$x \cdot x = x \tag{5}$$

Definition 2.4 *An unipotent quasigroup UQ is a quasigroup with the additional property:*

$$x \cdot x = 1 \tag{5}$$

where 1 is a new constant symbol.

Definition 2.5 *A semisymmetric quasigroup SQ is a quasigroup with the additional property:*

$$x \cdot (y \cdot x) = y \tag{5}$$

2.2 Example

See the references which are mentioned in [Hul80].

2.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical system for quasigroups:

$$
\begin{aligned}
x \cdot (x \backslash y) &\longrightarrow y && (1)\\
(x/y) \cdot y &\longrightarrow x && (2)\\
x \backslash (x \cdot y) &\longrightarrow y && (3)\\
(x \cdot y)/y &\longrightarrow x && (4)\\
(x/y) \backslash x &\longrightarrow y && (5)\\
y/(x \backslash y) &\longrightarrow x && (6)
\end{aligned}
$$

There is the following canonical system for abstract loops:

$$
\begin{aligned}
x \cdot (x \backslash y) &\longrightarrow y && (1)\\
(x/y) \cdot y &\longrightarrow x && (2)\\
x \backslash (x \cdot y) &\longrightarrow y && (3)\\
(x \cdot y)/y &\longrightarrow x && (4)\\
(x/y) \backslash x &\longrightarrow y && (5)\\
y/(x \backslash y) &\longrightarrow x && (6)\\
e \cdot x &\longrightarrow x && (7)\\
x \cdot e &\longrightarrow x && (8)\\
x/e &\longrightarrow x && (9)\\
x \backslash x &\longrightarrow e && (10)\\
y/y &\longrightarrow e && (11)\\
e \backslash y &\longrightarrow y && (12)
\end{aligned}
$$

There is the following canonical system for idempotent quasigroups:

$$
\begin{aligned}
x \cdot (x \backslash y) &\longrightarrow y && (1)\\
(x/y) \cdot y &\longrightarrow x && (2)\\
x \backslash (x \cdot y) &\longrightarrow y && (3)\\
(x \cdot y)/y &\longrightarrow x && (4)\\
(x/y) \backslash x &\longrightarrow y && (5)\\
y/(x \backslash y) &\longrightarrow x && (6)\\
x \cdot x &\longrightarrow x && (7)\\
y/y &\longrightarrow y && (8)\\
y \backslash y &\longrightarrow y && (9)
\end{aligned}
$$

There is the following canonical systems for unipotent quasigroups:

$$
\begin{aligned}
x \cdot (x \backslash y) &\longrightarrow y && (1)\\
(x/y) \cdot y &\longrightarrow x && (2)\\
x \backslash (x \cdot y) &\longrightarrow y && (3)\\
(x \cdot y)/y &\longrightarrow x && (4)\\
(x/y) \backslash x &\longrightarrow y && (5)\\
y/(x \backslash y) &\longrightarrow x && (6)\\
x \cdot x &\longrightarrow 1 && (7)\\
1/y &\longrightarrow y && (8)\\
y \backslash 1 &\longrightarrow y && (9)
\end{aligned}
$$

There is the following canonical system for semisymmetric quasigroups:

$$
\begin{aligned}
x \backslash y &\longrightarrow x/y && (1)\\
x \cdot y &\longrightarrow y/x && (2)\\
x/(y/x) &\longrightarrow y && (3)\\
(x/y)/x &\longrightarrow y && (4)
\end{aligned}
$$

2.4 Trace

The operation $\cdot$ is represented by $*$ and e by E resp.; \, /, 1 remain the same.

```
RRL-> read test/2a.eq
Equations read in are:
 1. (X * (X \ Y)) == Y  [USER, 1]
 2. ((X / Y) * Y) == X  [USER, 2]
 3. (X \ (X * Y)) == Y  [USER, 3]
 4. ((X * Y) / Y) == X  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [3] (X \ (X * Y)) ---> Y  [USER, 3]

Adding Rule:   [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [3] (X \ (X * Y)) ---> Y  [USER, 3]
```

```
Computing critical pairs with:
    [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [6] (Y / (X \ Y)) ---> X  [1, 4]

Computing critical pairs with:
    [6] (Y / (X \ Y)) ---> X  [1, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]

Time used                           =   0.18 sec
Number of rules generated           = 6
Number of rules retained            = 6
Number of critical pairs            = 12
Time spent in normalization         =   0.02 sec  (11.0 percent of time)
Time spent in unification           =   0.01 sec   (5.0 percent of time)
Time spent in ordering              =   0.02 sec   (11.0 percent of time)
Time spent in simplifying the rules =   0.04 sec   (22.0 percent of time)
Total time (including ´undo´ action) = 20.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2b.eq
New constant set is: { E }
Equations read in are:
 1. (E * X) == X  [USER, 5]
 2. (X * E) == X  [USER, 6]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 2 -----

Adding Rule:   [7] (E * X) ---> X  [USER, 5]

Adding Rule:   [8] (X * E) ---> X  [USER, 6]

Computing critical pairs with:
    [8] (X * E) ---> X  [USER, 6]

Adding Rule:   [9] (X / E) ---> X  [8, 4]

Adding Rule:  [10] (X \ X) ---> E  [8, 3]

Computing critical pairs with:
```

```
   [10] (X \ X) ---> E  [8, 3]

Computing critical pairs with:
    [9] (X / E) ---> X  [8, 4]

Computing critical pairs with:
    [7] (E * X) ---> X  [USER, 5]

Adding Rule:  [11] (Y / Y) ---> E  [7, 4]

Adding Rule:  [12] (E \ Y) ---> Y  [7, 3]

Computing critical pairs with:
   [12] (E \ Y) ---> Y  [7, 3]

Computing critical pairs with:
   [11] (Y / Y) ---> E  [7, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]
  [7] (E * X) ---> X  [USER, 5]
  [8] (X * E) ---> X  [USER, 6]
  [9] (X / E) ---> X  [8, 4]
 [10] (X \ X) ---> E  [8, 3]
 [11] (Y / Y) ---> E  [7, 4]
 [12] (E \ Y) ---> Y  [7, 3]

Time used                           =   0.38 sec
Number of rules generated           = 12
Number of rules retained            = 12
Number of critical pairs            = 31
Time spent in normalization         =   0.04 sec  (10.0 percent of time)
Time spent in unification           =   0.06 sec   (15.0 percent of time)
Time spent in ordering              =   0.03 sec   (7.0 percent of time)
Time spent in simplifying the rules =   0.08 sec   (21.0 percent of time)
Total time (including 'undo' action) = 24.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2a.eq
Equations read in are:
 1. (X * (X \ Y)) == Y  [USER, 1]
 2. ((X / Y) * Y) == X  [USER, 2]
 3. (X \ (X * Y)) == Y  [USER, 3]
 4. ((X * Y) / Y) == X  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
```

```
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [3] (X \ (X * Y)) ---> Y  [USER, 3]

Adding Rule:   [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [3] (X \ (X * Y)) ---> Y  [USER, 3]

Computing critical pairs with:
    [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [6] (Y / (X \ Y)) ---> X  [1, 4]

Computing critical pairs with:
    [6] (Y / (X \ Y)) ---> X  [1, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]

Time used                           =   0.12 sec
Number of rules generated           = 6
Number of rules retained            = 6
Number of critical pairs            = 12
Time spent in normalization         =   0.00 sec  (0.0 percent of time)
Time spent in unification           =   0.02 sec   (16.0 percent of time)
Time spent in ordering              =   0.02 sec   (16.0 percent of time)
Time spent in simplifying the rules =   0.02 sec   (16.0 percent of time)
Total time (including 'undo' action) = 15.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2c.eq
Equations read in are:
 1. (X * X) == X  [USER, 5]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
```

```
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 2 -----

Adding Rule:   [7] (X * X) ---> X  [USER, 5]

Computing critical pairs with:
    [7] (X * X) ---> X  [USER, 5]

Adding Rule:   [8] (Y / Y) ---> Y  [7, 4]

Adding Rule:   [9] (Y \ Y) ---> Y  [7, 3]

Computing critical pairs with:
    [9] (Y \ Y) ---> Y  [7, 3]

Computing critical pairs with:
    [8] (Y / Y) ---> Y  [7, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]
  [7] (X * X) ---> X  [USER, 5]
  [8] (Y / Y) ---> Y  [7, 4]
  [9] (Y \ Y) ---> Y  [7, 3]

Time used                          =   0.21 sec
Number of rules generated          = 9
Number of rules retained           = 9
Number of critical pairs           = 18
Time spent in normalization        =   0.01 sec  (4.0 percent of time)
Time spent in unification          =   0.03 sec   (14.0 percent of time)
Time spent in ordering             =   0.03 sec   (14.0 percent of time)
Time spent in simplifying the rules  =   0.05 sec   (23.0 percent of time)
Total time (including 'undo' action) = 11.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2a.eq
Equations read in are:
 1. (X * (X \ Y)) == Y  [USER, 1]
 2. ((X / Y) * Y) == X  [USER, 2]
 3. (X \ (X * Y)) == Y  [USER, 3]
 4. ((X * Y) / Y) == X  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
```

```
----- Step 1 -----

Adding Rule:   [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [3] (X \ (X * Y)) ---> Y  [USER, 3]

Adding Rule:   [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [3] (X \ (X * Y)) ---> Y  [USER, 3]

Computing critical pairs with:
    [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [6] (Y / (X \ Y)) ---> X  [1, 4]

Computing critical pairs with:
    [6] (Y / (X \ Y)) ---> X  [1, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]

Time used                         =   0.16 sec
Number of rules generated         = 6
Number of rules retained          = 6
Number of critical pairs          = 12
Time spent in normalization       =   0.00 sec  (0.0 percent of time)
Time spent in unification         =   0.02 sec   (12.0 percent of time)
Time spent in ordering            =   0.02 sec   (12.0 percent of time)
Time spent in simplifying the rules  =   0.03 sec   (18.0 percent of time)
Total time (including 'undo' action) = 18.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2d.eq
New constant set is: { 1 }
Equations read in are:
 1. (X * X) == 1  [USER, 5]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
```

```
RRL-> akb
----- Step 2 -----

Adding Rule:   [7] (X * X) ---> 1  [USER, 5]

Computing critical pairs with:
    [7] (X * X) ---> 1  [USER, 5]

Adding Rule:   [8] (1 / Y) ---> Y  [7, 4]

Adding Rule:   [9] (Y \ 1) ---> Y  [7, 3]

Computing critical pairs with:
    [9] (Y \ 1) ---> Y  [7, 3]

Computing critical pairs with:
    [8] (1 / Y) ---> Y  [7, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]
  [7] (X * X) ---> 1  [USER, 5]
  [8] (1 / Y) ---> Y  [7, 4]
  [9] (Y \ 1) ---> Y  [7, 3]

Time used                           =   0.26 sec
Number of rules generated           = 9
Number of rules retained            = 9
Number of critical pairs            = 20
Time spent in normalization         =   0.00 sec  (0.0 percent of time)
Time spent in unification           =   0.04 sec   (15.0 percent of time)
Time spent in ordering              =   0.03 sec   (11.0 percent of time)
Time spent in simplifying the rules =   0.05 sec   (19.0 percent of time)
Total time (including 'undo' action) = 14.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2a.eq
Equations read in are:
 1. (X * (X \ Y)) == Y  [USER, 1]
 2. ((X / Y) * Y) == X  [USER, 2]
 3. (X \ (X * Y)) == Y  [USER, 3]
 4. ((X * Y) / Y) == X  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----
```

```
Adding Rule:   [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [3] (X \ (X * Y)) ---> Y  [USER, 3]

Adding Rule:   [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [4] ((X * Y) / Y) ---> X  [USER, 4]

Computing critical pairs with:
    [3] (X \ (X * Y)) ---> Y  [USER, 3]

Computing critical pairs with:
    [2] ((X / Y) * Y) ---> X  [USER, 2]

Adding Rule:   [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [5] ((X / Y) \ X) ---> Y  [2, 3]

Computing critical pairs with:
    [1] (X * (X \ Y)) ---> Y  [USER, 1]

Adding Rule:   [6] (Y / (X \ Y)) ---> X  [1, 4]

Computing critical pairs with:
    [6] (Y / (X \ Y)) ---> X  [1, 4]

Your system is canonical.

  [1] (X * (X \ Y)) ---> Y  [USER, 1]
  [2] ((X / Y) * Y) ---> X  [USER, 2]
  [3] (X \ (X * Y)) ---> Y  [USER, 3]
  [4] ((X * Y) / Y) ---> X  [USER, 4]
  [5] ((X / Y) \ X) ---> Y  [2, 3]
  [6] (Y / (X \ Y)) ---> X  [1, 4]

Time used                          =   0.92 sec
Number of rules generated          = 6
Number of rules retained           = 6
Number of critical pairs           = 12
Time spent in normalization        =   0.01 sec  (1.0 percent of time)
Time spent in unification          =   0.01 sec   (1.0 percent of time)
Time spent in ordering             =   0.03 sec   (3.0 percent of time)
Time spent in simplifying the rules =   0.80 sec   (86.0 percent of time)
Total time (including 'undo' action) = 93.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/2e.eq
Equations read in are:
 1. (X * (Y * X)) == Y  [USER, 5]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 2 -----
```

```
Adding Rule:   [7] (X * (Y * X)) ---> Y  [USER, 5]

Computing critical pairs with:
    [7] (X * (Y * X)) ---> Y  [USER, 5]

Adding Rule:   [8] (Y / (Y * X)) ---> X  [7, 4]

Trying to orient equation:
  (X \ Y) == (Y * X)  [7, 3]

  To prove:  (X \ Y)  >  (Y * X)
        or:  (X \ Y)  <  (Y * X)
  Here are some precedence suggestions:
       1.  \ > *      2.  * > \
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, \ > *, is added.

----- Step 3 -----

Adding Rule:   [9] (X \ Y) ---> (Y * X)  [7, 3]

  Deleting rule:  [1] (X * (X \ Y)) ---> Y  [USER, 1]

  Deleting rule:  [3] (X \ (X * Y)) ---> Y  [USER, 3]

  Deleting rule:  [5] ((X / Y) \ X) ---> Y  [2, 3]

  Deleting rule:  [6] (Y / (X \ Y)) ---> X  [1, 4]

Adding Rule:  [10] (X * (X / Y)) ---> Y  [DELETED, 5]

Adding Rule:  [11] ((X * Y) * X) ---> Y  [DELETED, 3]

Trying to orient equation:
  (X * X1) == (X1 / X)  [2, 7]

  To prove:  (X * X1)  >  (X1 / X)
        or:  (X * X1)  <  (X1 / X)
  Here are some precedence suggestions:
       1.  * > /      2.  / > *
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, * > /, is added.

----- Step 4 -----

Adding Rule:  [12] (X * X1) ---> (X1 / X)  [2, 7]

  Deleting rule:  [2] ((X / Y) * Y) ---> X  [USER, 2]

  Deleting rule:  [4] ((X * Y) / Y) ---> X  [USER, 4]

  Deleting rule:  [7] (X * (Y * X)) ---> Y  [USER, 5]

  Deleting rule:  [8] (Y / (Y * X)) ---> X  [7, 4]

  The right hand side is reducible:
      [9] (X \ Y) ---> (Y * X)  [7, 3]
```

```
  Deleting rule: [10] (X * (X / Y)) ---> Y  [DELETED, 5]

  Deleting rule: [11] ((X * Y) * X) ---> Y  [DELETED, 3]

Adding Rule:  [13] (X / (Y / X)) ---> Y  [DELETED, 11]

Adding Rule:  [14] ((X / Y) / X) ---> Y  [DELETED, 10]

Computing critical pairs with:
   [12] (X * X1) ---> (X1 / X)  [2, 7]

Computing critical pairs with:
    [9] (X \ Y) ---> (X / Y)  [7, 3]

Computing critical pairs with:
   [14] ((X / Y) / X) ---> Y  [DELETED, 10]

Computing critical pairs with:
   [13] (X / (Y / X)) ---> Y  [DELETED, 11]

Your system is canonical.

  [9] (X \ Y) ---> (X / Y)  [7, 3]
 [12] (X * X1) ---> (X1 / X)  [2, 7]
 [13] (X / (Y / X)) ---> Y  [DELETED, 11]
 [14] ((X / Y) / X) ---> Y  [DELETED, 10]

Time used                           =   1.22 sec
Number of rules generated           = 14
Number of rules retained            = 4
Number of critical pairs            = 19
Time spent in normalization         =   0.03 sec  (2.0 percent of time)
Time spent in unification           =   0.01 sec   (0.0 percent of time)
Time spent in ordering              =   0.13 sec   (10.0 percent of time)
Time spent in simplifying the rules =   0.92 sec   (75.0 percent of time)
Total time (including 'undo' action) = 31.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

3 Semigroups and Monoids

3.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 3.1 *A semigroup SG is a groupoid with the following axiom:*

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \tag{1}$$

Definition 3.2 *A monoid MO is a semigroup with unit 1, i.e. there hold additionally the following axioms:*

$$\begin{aligned} x \cdot 1 &= x & (2) \\ 1 \cdot x &= x & (3) \end{aligned}$$

Definition 3.3 *An idempotent semigroup IS is a semigroup with the following axiom:*

$$x \cdot x = x \tag{2}$$

3.2 Example

A model of a semigroup is the set of all functions $M \rightarrow M$, where M is an arbitrary but fixed set. The $\cdot$ operation is the composition of functions. Additional this is even a monoid, where 1 is the identity function.

A model of an idempotent semigroup is the set of all integers with the minimum (or maximum) operation. This (both) semigroup(s) is (are) not a monoid.

3.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical system for semigroups:

$$x \cdot (y \cdot z) \longrightarrow (x \cdot y) \cdot z \tag{1}$$

There is the following canonical system for monoids:

$$\begin{aligned} x \cdot (y \cdot z) &\longrightarrow (x \cdot y) \cdot z & (1) \\ x \cdot 1 &\longrightarrow x & (2) \\ 1 \cdot x &\longrightarrow x & (3) \end{aligned}$$

No canonical system for idempotent semigroups is presented in [Hul80]. RRL reports its inability to complete this system.

3.4 Trace

The operation $\cdot$ is represented by $*$, 1 remains the same.

```
RRL-> read test/3a.eq
Equations read in are:
 1. ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 2 -----

Processing all postponed equations ...

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 3 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Operator, *, given status: rl

----- Step 4 -----

Adding Rule:   [1] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 1]

Computing critical pairs with:
    [1] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 1]
```

```
Your system is canonical.

  [1] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 1]

Time used                           =   0.17 sec
Number of rules generated           = 1
Number of rules retained            = 1
Number of critical pairs            = 1
Time spent in normalization         =   0.00 sec  (0.0 percent of time)
Time spent in unification           =   0.00 sec   (0.0 percent of time)
Time spent in ordering              =   0.13 sec   (76.0 percent of time)
Time spent in simplifying the rules  =   0.01 sec   (5.0 percent of time)
Total time (including 'undo' action) = 18.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/3b.eq
New constant set is: { 1 }
Equations read in are:
 1. (X * 1) == X  [USER, 2]
 2. (1 * X) == X  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 5 -----

Adding Rule:   [2] (X * 1) ---> X  [USER, 2]

Adding Rule:   [3] (1 * X) ---> X  [USER, 3]

Computing critical pairs with:
    [3] (1 * X) ---> X  [USER, 3]

Computing critical pairs with:
    [2] (X * 1) ---> X  [USER, 2]

Your system is canonical.

  [1] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 1]
  [2] (X * 1) ---> X  [USER, 2]
  [3] (1 * X) ---> X  [USER, 3]

Time used                           =   0.21 sec
Number of rules generated           = 3
Number of rules retained            = 3
Number of critical pairs            = 5
Time spent in normalization         =   0.00 sec  (0.0 percent of time)
Time spent in unification           =   0.01 sec   (4.0 percent of time)
Time spent in ordering              =   0.14 sec   (66.0 percent of time)
Time spent in simplifying the rules  =   0.01 sec   (4.0 percent of time)
Total time (including 'undo' action) = 5.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
```

```
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/3c.eq
Equations read in are:
 1. ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]
 2. (X * X) == X  [USER, 2]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> status * lr
Operator, *, given status: LR

----- Step 1 -----

Adding Rule:   [1] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 1]

Adding Rule:   [2] (X * X) ---> X  [USER, 2]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 2 -----

Computing critical pairs with:
    [2] (X * X) ---> X  [USER, 2]

Computing critical pairs with:
    [1] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 1]

Adding Rule:   [3] (X * (Y * (X * Y))) ---> (X * Y)  [2, 1]

Adding Rule:   [4] (Y * (Y * Z)) ---> (Y * Z)  [2, 1]

Computing critical pairs with:
    [4] (Y * (Y * Z)) ---> (Y * Z)  [2, 1]

Adding Rule:   [5] (X * (Y * (X * (Y * Z)))) ---> (X * (Y * Z))  [1, 4]

Computing critical pairs with:
    [3] (X * (Y * (X * Y))) ---> (X * Y)  [2, 1]

Adding Rule:   [6] (X * (Y * (Y1 * (X * (Y * Y1))))) ---> (X * (Y * Y1))  [1, 3]

Computing critical pairs with:
    [5] (X * (Y * (X * (Y * Z)))) ---> (X * (Y * Z))  [1, 4]
```

```
Adding Rule:   [7] (X * (Y * (Y1 * (X * (Y * (Y1 * Z)))))) ---> (X * (Y * (Y1 * Z)))  [1, 5]

Computing critical pairs with:
    [6] (X * (Y * (Y1 * (X * (Y * Y1))))) ---> (X * (Y * Y1))  [1, 3]

Adding Rule:   [8] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2))))))) ---> (X * (Y * (Y1 * Y2)))
  [1, 6]

Computing critical pairs with:
    [7] (X * (Y * (Y1 * (X * (Y * (Y1 * Z)))))) ---> (X * (Y * (Y1 * Z)))  [1, 5]

Adding Rule:   [9] (X * (Y * (Z * (Y * (X * (Y * Z)))))) ---> (X * (Y * Z))  [2, 7]

Adding Rule:  [10] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 7]

Adding Rule:  [11] (X * (Y * (Y1 * (Y2 * (Z * (X * (Y * (Y1 * (Y2 * Z))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [7, 6]

Computing critical pairs with:
    [9] (X * (Y * (Z * (Y * (X * (Y * Z)))))) ---> (X * (Y * Z))  [2, 7]

Adding Rule:  [12] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * Z)))))))) --->
(X * (Y * (Y1 * Z)))  [1, 9]

Adding Rule:  [13] (X * (X1 * (Y * (Z * (X1 * (Y * (X * (X1 * (Y * Z))))))))) --->
(X * (X1 * (Y * Z)))  [1, 9]

Adding Rule:  [14] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * Y2)))))))) --->
(X * (Y * (X1 * Y2)))  [1, 9]

Adding Rule:  [15] (X * (Y * (Z * (Y * (X * (Y * (Z * Z1))))))) --->
(X * (Y * (Z * Z1)))  [9, 1]

Adding Rule:  [16] (X * (Y * (Y1 * (Y2 * (Z * (Y * (X * (Y * (Y1 * (Y2 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [7, 9]

Computing critical pairs with:
   [15] (X * (Y * (Z * (Y * (X * (Y * (Z * Z1))))))) ---> (X * (Y * (Z * Z1)))  [9, 1]

Adding Rule:  [17] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * (Z * Z2))))))))) --->
(X * (Y * (Y1 * (Z * Z2))))  [1, 15]

Adding Rule:  [18] (X * (X1 * (Y * (Z * (X1 * (Y * (X * (X1 * (Y * (Z * Z2)))))))))) --->
(X * (X1 * (Y * (Z * Z2))))  [1, 15]

Adding Rule:  [19] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 15]

Computing critical pairs with:
    [8] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2))))))) ---> (X * (Y * (Y1 * Y2)))  [1, 6]

Adding Rule:  [20] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (X * (Y * (Y1 * (Y2 * (Y3 * Z))))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 8]

Computing critical pairs with:
   [14] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * Y2)))))))) ---> (X * (Y * (X1 * Y2)))  [1, 9]

Adding Rule:  [21] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (X * (Y * (Y1 * (X2 * Y3)))))))))) --->
(X * (Y * (Y1 * (X2 * Y3))))  [1, 14]

Adding Rule:  [22] (X * (X1 * (Y * (X2 * (Y3 * (X1 * (Y * (X * (X1 * (Y * (X2 * Y3))))))))))) --->
(X * (X1 * (Y * (X2 * Y3))))  [1, 14]

Adding Rule:  [23] (X * (Y * (X1 * (Y2 * (Y3 * (Z * (Y * (X * (Y * (X1 * (Y2 * (Y3 * Z
```

```
)))))))))))) ---> (X * (Y * (X1 * (Y2 * (Y3 * Z)))))  [7, 14]

Computing critical pairs with:
   [12] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * Z)))))))) ---> (X * (Y * (Y1 * Z)))  [1, 9]

Adding Rule:  [24] (X * (Y * (Y1 * (Y2 * (Z * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 12]

Adding Rule:  [25] (X * (Y * (X1 * (Y2 * (Z * (X1 * (Y2 * (X * (Y * (X1 * (Y2 * Z))))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 12]

Adding Rule:  [26] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (Y1 * (X * (Y * (Y1 * (Y2 * (Y3 * Z
)))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 12]

Computing critical pairs with:
   [10] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))) ---> (X * (Y * (Y1 * (Y2 * Z))))  [1, 7]

Adding Rule:  [27] (X * (Y * (Y1 * (Y2 * (Y3 * (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [1, 10]

Adding Rule:  [28] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z
))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 8]

Adding Rule:  [29] (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * (Z * (Y * (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * Z
)))))))))))))) ---> (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 14]

Adding Rule:  [30] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (Y1 * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z
)))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 12]

Computing critical pairs with:
   [19] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 15]

Adding Rule:  [31] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * Z))))))))) --->
(X * (Y * (Y1 * Z)))  [2, 19]

Adding Rule:  [32] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (X * (Y * (Y1 * (X2 * (Y3 * Z))))))))))) --->
(X * (Y * (Y1 * (X2 * (Y3 * Z)))))  [1, 19]

Adding Rule:  [33] (X * (X1 * (Y * (X2 * (Y3 * (X1 * (Y * (X * (X1 * (Y * (X2 * (Y3 * Z
)))))))))))) ---> (X * (X1 * (Y * (X2 * (Y3 * Z)))))  [1, 19]

Adding Rule:  [34] (X * (Y * (X1 * (Y2 * (Y3 * (Y * (X * (Y * (X1 * (Y2 * (Y3 * Z))))))))))) --->
(X * (Y * (X1 * (Y2 * (Y3 * Z)))))  [1, 19]

Computing critical pairs with:
   [31] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * Z))))))))) ---> (X * (Y * (Y1 * Z)))  [2, 19]

Adding Rule:  [35] (X * (Y * (Y1 * (Y2 * (Z * (Y2 * (Y1 * (X * (Y * (Y1 * (Y2 * Z))))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 31]

Adding Rule:  [36] (X * (X1 * (Y * (Y2 * (Z * (Y2 * (X1 * (Y * (X * (X1 * (Y * (Y2 * Z
)))))))))))) ---> (X * (X1 * (Y * (Y2 * Z))))  [1, 31]

Adding Rule:  [37] (X * (Y * (X1 * (Y2 * (Z * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z
)))))))))))) ---> (X * (Y * (X1 * (Y2 * Z))))  [1, 31]

Adding Rule:  [38] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (Y * (X * (Y * (Y1 * (X2 * Y3))))))))))) --->
(X * (Y * (Y1 * (X2 * Y3))))  [1, 31]

Adding Rule:  [39] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Z * Z2)))))))))) --->
(X * (Y * (Y1 * (Z * Z2))))  [31, 1]

Adding Rule:  [40] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Y2 * (Y3 * Z
))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 31]
```

```
Adding Rule:  [41] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * Y1)))))))) ---> (X * (Y * Y1))  [31, 7]

Adding Rule:  [42] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z
))))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 31]

Computing critical pairs with:
   [41] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * Y1)))))))) ---> (X * (Y * Y1))  [31, 7]

Adding Rule:  [43] (X * (Y * (Y1 * (Y2 * (Y1 * (X * (Y * (Y2 * (X * (Y * (Y1 * Y2))))))))))) --->
(X * (Y * (Y1 * Y2)))  [1, 41]

Adding Rule:  [44] (X * (X1 * (Y * (Y2 * (X1 * (Y * (X * (Y2 * (X * (X1 * (Y * Y2))))))))))) --->
(X * (X1 * (Y * Y2)))  [1, 41]

Adding Rule:  [45] (X * (Y * (X1 * (Y2 * (Y * (X * (X1 * (Y2 * (X * (Y * (X1 * Y2))))))))))) --->
(X * (Y * (X1 * Y2)))  [1, 41]

Adding Rule:  [46] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * (Y1 * Z))))))))) --->
(X * (Y * (Y1 * Z)))  [41, 1]

Adding Rule:  [47] (X * (Y * (Y1 * (Y * (X * (Y1 * (Y * (Y1 * (X * (Y * Y1)))))))))) --->
(X * (Y * Y1))  [3, 41]

Adding Rule:  [48] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (Y * (Y1 * (Z * (X * (Y * (Y1 * Z
))))))))))))) ---> (X * (Y * (Y1 * Z)))  [5, 41]

Adding Rule:  [49] (X * (Y * (Y1 * (Y2 * (Y * (X * (Y1 * (Y2 * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2
)))))))))))))) ---> (X * (Y * (Y1 * Y2)))  [6, 41]

Adding Rule:  [50] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (X * (Y * (Y1 * Z)))))))))) --->
(X * (Y * (Y1 * Z)))  [7, 41]

Adding Rule:  [51] (X * (Y * (Y1 * (Y2 * (Z * (Y * (X * (Y1 * (Y2 * (Y * (Y1 * (Y2 * (Z *
(X * (Y * (Y1 * (Y2 * Z))))))))))))))))) ---> (X * (Y * (Y1 * (Y2 * Z))))  [7, 41]

Adding Rule:  [52] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (Z * (Y1 * (Y * (Y1 * (Z * (X * (Y * (Y1 * Z
))))))))))))))) ---> (X * (Y * (Y1 * Z)))  [9, 41]

Adding Rule:  [53] (X * (Y * (Z * (Z1 * (Y * (X * (Z * (Y * (X * (Y * (Z * Z1))))))))))) --->
(X * (Y * (Z * Z1)))  [15, 41]

WARNING: More than 50 new rules generated without any change of the precedence.I undo it to last interaction.
----- Return to Step 1 -----

Computing critical pairs with:
    [2] (X * X) ---> X  [USER, 2]

Computing critical pairs with:
    [1] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 1]

Adding Rule:   [3] (X * (Y * (X * Y))) ---> (X * Y)  [2, 1]

Adding Rule:   [4] (Y * (Y * Z)) ---> (Y * Z)  [2, 1]

Computing critical pairs with:
    [4] (Y * (Y * Z)) ---> (Y * Z)  [2, 1]

Adding Rule:   [5] (X * (Y * (X * (Y * Z)))) ---> (X * (Y * Z))  [1, 4]

Computing critical pairs with:
    [3] (X * (Y * (X * Y))) ---> (X * Y)  [2, 1]

Adding Rule:   [6] (X * (Y * (Y1 * (X * (Y * Y1))))) ---> (X * (Y * Y1))  [1, 3]
```

```
Computing critical pairs with:
    [5] (X * (Y * (X * (Y * Z)))) ---> (X * (Y * Z))  [1, 4]

Adding Rule:   [7] (X * (Y * (Y1 * (X * (Y * (Y1 * Z)))))) ---> (X * (Y * (Y1 * Z)))  [1, 5]

Computing critical pairs with:
    [6] (X * (Y * (Y1 * (X * (Y * Y1))))) ---> (X * (Y * Y1))  [1, 3]

Adding Rule:   [8] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2))))))) ---> (X * (Y * (Y1 * Y2)))  [1, 6]

Computing critical pairs with:
    [7] (X * (Y * (Y1 * (X * (Y * (Y1 * Z)))))) ---> (X * (Y * (Y1 * Z)))  [1, 5]

Adding Rule:   [9] (X * (Y * (Z * (Y * (X * (Y * Z)))))) ---> (X * (Y * Z))  [2, 7]

Adding Rule:  [10] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 7]

Adding Rule:  [11] (X * (Y * (Y1 * (Y2 * (Z * (X * (Y * (Y1 * (Y2 * Z))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [7, 6]

Computing critical pairs with:
    [9] (X * (Y * (Z * (Y * (X * (Y * Z)))))) ---> (X * (Y * Z))  [2, 7]

Adding Rule:  [12] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * Z)))))))) ---> (X * (Y * (Y1 * Z)))  [1, 9]

Adding Rule:  [13] (X * (X1 * (Y * (Z * (X1 * (Y * (X * (X1 * (Y * Z))))))))) --->
(X * (X1 * (Y * Z)))  [1, 9]

Adding Rule:  [14] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * Y2)))))))) --->
(X * (Y * (X1 * Y2)))  [1, 9]

Adding Rule:  [15] (X * (Y * (Z * (Y * (X * (Y * (Z * Z1))))))) ---> (X * (Y * (Z * Z1)))  [9, 1]

Adding Rule:  [16] (X * (Y * (Y1 * (Y2 * (Z * (Y * (X * (Y * (Y1 * (Y2 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [7, 9]

Computing critical pairs with:
   [15] (X * (Y * (Z * (Y * (X * (Y * (Z * Z1))))))) ---> (X * (Y * (Z * Z1)))  [9, 1]

Adding Rule:  [17] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * (Z * Z2))))))))) --->
(X * (Y * (Y1 * (Z * Z2))))  [1, 15]

Adding Rule:  [18] (X * (X1 * (Y * (Z * (X1 * (Y * (X * (X1 * (Y * (Z * Z2)))))))))) --->
(X * (X1 * (Y * (Z * Z2))))  [1, 15]

Adding Rule:  [19] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 15]

Computing critical pairs with:
    [8] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2))))))) ---> (X * (Y * (Y1 * Y2)))  [1, 6]

Adding Rule:  [20] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (X * (Y * (Y1 * (Y2 * (Y3 * Z))))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 8]

Computing critical pairs with:
   [14] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * Y2)))))))) ---> (X * (Y * (X1 * Y2)))  [1, 9]

Adding Rule:  [21] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (X * (Y * (Y1 * (X2 * Y3)))))))))) --->
(X * (Y * (Y1 * (X2 * Y3))))  [1, 14]

Adding Rule:  [22] (X * (X1 * (Y * (X2 * (Y3 * (X1 * (Y * (X * (X1 * (Y * (X2 * Y3))))))))))) --->
(X * (X1 * (Y * (X2 * Y3))))  [1, 14]

Adding Rule:  [23] (X * (Y * (X1 * (Y2 * (Y3 * (Z * (Y * (X * (Y * (X1 * (Y2 * (Y3 * Z)))))))))))) --->
```

```
(X * (Y * (X1 * (Y2 * (Y3 * Z)))))  [7, 14]

Computing critical pairs with:
   [12] (X * (Y * (Y1 * (Z * (Y1 * (X * (Y * (Y1 * Z)))))))) ---> (X * (Y * (Y1 * Z)))  [1, 9]

Adding Rule:  [24] (X * (Y * (Y1 * (Y2 * (Z * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 12]

Adding Rule:  [25] (X * (Y * (X1 * (Y2 * (Z * (X1 * (Y2 * (X * (Y * (X1 * (Y2 * Z))))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 12]

Adding Rule:  [26] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (Y1 * (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 12]

Computing critical pairs with:
   [10] (X * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * (Y2 * Z)))))))) ---> (X * (Y * (Y1 * (Y2 * Z))))  [1, 7]

Adding Rule:  [27] (X * (Y * (Y1 * (Y2 * (Y3 * (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [1, 10]

Adding Rule:  [28] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))))))))) --->
(X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 8]

Adding Rule:  [29] (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * (Z * (Y * (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * Z
)))))))))))))) ---> (X * (Y * (X1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 14]

Adding Rule:  [30] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (Y1 * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z
)))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 12]

Computing critical pairs with:
   [19] (X * (Y * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z))))))))) ---> (X * (Y * (X1 * (Y2 * Z))))  [1, 15]

Adding Rule:  [31] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * Z))))))))) --->
(X * (Y * (Y1 * Z)))  [2, 19]

Adding Rule:  [32] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (X * (Y * (Y1 * (X2 * (Y3 * Z))))))))))) --->
(X * (Y * (Y1 * (X2 * (Y3 * Z)))))  [1, 19]

Adding Rule:  [33] (X * (X1 * (Y * (X2 * (Y3 * (X1 * (Y * (X * (X1 * (Y * (X2 * (Y3 * Z)))))))))))) --->
(X * (X1 * (Y * (X2 * (Y3 * Z)))))  [1, 19]

Adding Rule:  [34] (X * (Y * (X1 * (Y2 * (Y3 * (Y * (X * (Y * (X1 * (Y2 * (Y3 * Z))))))))))) --->
(X * (Y * (X1 * (Y2 * (Y3 * Z)))))  [1, 19]

Computing critical pairs with:
   [31] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * Z))))))))) ---> (X * (Y * (Y1 * Z)))  [2, 19]

Adding Rule:  [35] (X * (Y * (Y1 * (Y2 * (Z * (Y2 * (Y1 * (X * (Y * (Y1 * (Y2 * Z))))))))))) --->
(X * (Y * (Y1 * (Y2 * Z))))  [1, 31]

Adding Rule:  [36] (X * (X1 * (Y * (Y2 * (Z * (Y2 * (X1 * (Y * (X * (X1 * (Y * (Y2 * Z)))))))))))) --->
(X * (X1 * (Y * (Y2 * Z))))  [1, 31]

Adding Rule:  [37] (X * (Y * (X1 * (Y2 * (Z * (X1 * (Y2 * (Y * (X * (Y * (X1 * (Y2 * Z)))))))))))) --->
(X * (Y * (X1 * (Y2 * Z))))  [1, 31]

Adding Rule:  [38] (X * (Y * (Y1 * (X2 * (Y3 * (Y1 * (Y * (X * (Y * (Y1 * (X2 * Y3))))))))))) --->
(X * (Y * (Y1 * (X2 * Y3))))  [1, 31]

Adding Rule:  [39] (X * (Y * (Y1 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Z * Z2)))))))))) --->
(X * (Y * (Y1 * (Z * Z2))))  [31, 1]

Adding Rule:  [40] (X * (Y * (Y1 * (Y2 * (Y3 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Y2 * (Y3 * Z
))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * Z)))))  [7, 31]
```

```
Adding Rule:  [41] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * Y1)))))))) --->
(X * (Y * Y1))  [31, 7]

Adding Rule:  [42] (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * (Z * (Y1 * (Y * (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z
)))))))))))))) ---> (X * (Y * (Y1 * (Y2 * (Y3 * (Y4 * Z))))))  [10, 31]

Computing critical pairs with:
   [41] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * Y1)))))))) ---> (X * (Y * Y1))  [31, 7]

Adding Rule:  [43] (X * (Y * (Y1 * (Y2 * (Y1 * (X * (Y * (Y2 * (X * (Y * (Y1 * Y2))))))))))) --->
(X * (Y * (Y1 * Y2)))  [1, 41]

Adding Rule:  [44] (X * (X1 * (Y * (Y2 * (X1 * (Y * (X * (Y2 * (X * (X1 * (Y * Y2))))))))))) --->
(X * (X1 * (Y * Y2)))  [1, 41]

Adding Rule:  [45] (X * (Y * (X1 * (Y2 * (Y * (X * (X1 * (Y2 * (X * (Y * (X1 * Y2))))))))))) --->
(X * (Y * (X1 * Y2)))  [1, 41]

Adding Rule:  [46] (X * (Y * (Y1 * (Y * (X * (Y1 * (X * (Y * (Y1 * Z))))))))) --->
(X * (Y * (Y1 * Z)))  [41, 1]

Adding Rule:  [47] (X * (Y * (Y1 * (Y * (X * (Y1 * (Y * (Y1 * (X * (Y * Y1)))))))))) --->
(X * (Y * Y1))  [3, 41]

Adding Rule:  [48] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (Y * (Y1 * (Z * (X * (Y * (Y1 * Z
))))))))))))) ---> (X * (Y * (Y1 * Z)))  [5, 41]

Adding Rule:  [49] (X * (Y * (Y1 * (Y2 * (Y * (X * (Y1 * (Y2 * (Y * (Y1 * (Y2 * (X * (Y * (Y1 * Y2
)))))))))))))) ---> (X * (Y * (Y1 * Y2)))  [6, 41]

Adding Rule:  [50] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (X * (Y * (Y1 * Z)))))))))) --->
(X * (Y * (Y1 * Z)))  [7, 41]

Adding Rule:  [51] (X * (Y * (Y1 * (Y2 * (Z * (Y * (X * (Y1 * (Y2 * (Y * (Y1 * (Y2 * (Z *
(X * (Y * (Y1 * (Y2 * Z))))))))))))))))) ---> (X * (Y * (Y1 * (Y2 * Z))))  [7, 41]

Adding Rule:  [52] (X * (Y * (Y1 * (Z * (Y * (X * (Y1 * (Z * (Y1 * (Y * (Y1 * (Z *
(X * (Y * (Y1 * Z))))))))))))))) ---> (X * (Y * (Y1 * Z)))  [9, 41]

Adding Rule:  [53] (X * (Y * (Z * (Z1 * (Y * (X * (Z * (Y * (X * (Y * (Z * Z1))))))))))) --->
(X * (Y * (Z * Z1)))  [15, 41]

WARNING: More than 50 new rules generated without any change of the precedence.I undo it to last interaction.
Sorry. I cannot complete your system.



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

4 Groups

4.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 4.1 *A group G is a groupoid together with a new unary operation* $^{-1}$*, i.e.* $^{-1} : S \to S$*, fulfilling the following axioms:*

$$\begin{aligned} 1 \cdot x &= x & (1)\\ x^{-1} \cdot x &= 1 & (2)\\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) & (3) \end{aligned}$$

4.2 Example

All permutations (i.e. bijections) of a fixed finite set are models of a group. The operation $\cdot$ is the composition of permutations and the inverse operation is just the inverse permutation.

4.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical systems for groups:

$$\begin{aligned} 1 \cdot x &\longrightarrow x & (1)\\ x^{-1} \cdot x &\longrightarrow 1 & (2)\\ x \cdot (y \cdot z) &\longrightarrow (x \cdot y) \cdot z & (3)\\ (x \cdot z^{-1}) \cdot z &\longrightarrow x & (4)\\ 1^{-1} &\longrightarrow 1 & (5)\\ z \cdot 1 &\longrightarrow z & (6)\\ (z^{-1})^{-1} &\longrightarrow z & (7)\\ z \cdot z^{-1} &\longrightarrow 1 & (8)\\ (x \cdot z) \cdot z^{-1} &\longrightarrow x & (9)\\ (x \cdot y)^{-1} &\longrightarrow y^{-1} \cdot x^{-1} & (10) \end{aligned}$$

4.4 Trace

The operation $\cdot$ is represented by $*$ and $^{-1}$ by I(_), resp., 1 remains the same.

```
RRL-> read test/4.eq
New constant set is: { 1 }
Equations read in are:
 1. (1 * X) == X  [USER, 1]
```

```
 2. (I(X) * X) == 1  [USER, 2]
 3. ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] (1 * X) ---> X  [USER, 1]

Adding Rule:   [2] (I(X) * X) ---> 1  [USER, 2]

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 2 -----

Computing critical pairs with:
    [1] (1 * X) ---> X  [USER, 1]

Computing critical pairs with:
    [2] (I(X) * X) ---> 1  [USER, 2]

Processing all postponed equations ...

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 3 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Operator, *, given status: rl
```

```
----- Step 4 -----

Adding Rule:   [3] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]

Computing critical pairs with:
    [3] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]

Adding Rule:   [4] ((X * 1) * Z) ---> (X * Z)  [1, 3]

Adding Rule:   [5] ((I((Y * Z)) * Y) * Z) ---> 1  [2, 3]

Adding Rule:   [6] ((X * I(Z)) * Z) ---> (X * 1)  [2, 3]

Computing critical pairs with:
    [4] ((X * 1) * Z) ---> (X * Z)  [1, 3]

Adding Rule:   [7] (I(1) * Z) ---> Z  [2, 4]

Computing critical pairs with:
    [7] (I(1) * Z) ---> Z  [2, 4]

Adding Rule:   [8] ((X * I(1)) * Z) ---> (X * Z)  [7, 3]

Computing critical pairs with:
    [8] ((X * I(1)) * Z) ---> (X * Z)  [7, 3]

Adding Rule:   [9] (I(I(1)) * Z) ---> Z  [2, 8]

Computing critical pairs with:
    [9] (I(I(1)) * Z) ---> Z  [2, 8]

Adding Rule:  [10] I(1) ---> 1  [2, 9]

  Deleting rule:  [7] (I(1) * Z) ---> Z  [2, 4]

  Deleting rule:  [8] ((X * I(1)) * Z) ---> (X * Z)  [7, 3]

  Deleting rule:  [9] (I(I(1)) * Z) ---> Z  [2, 8]

Computing critical pairs with:
   [10] I(1) ---> 1  [2, 9]

Computing critical pairs with:
    [6] ((X * I(Z)) * Z) ---> (X * 1)  [2, 3]

Adding Rule:  [11] (I(I(Z)) * 1) ---> Z  [2, 6]

Adding Rule:  [12] (((X * I((Y * Z))) * Y) * Z) ---> (X * 1)  [3, 6]

Adding Rule:  [13] ((X * I(I(Z))) * 1) ---> (X * Z)  [6, 6]

Computing critical pairs with:
   [11] (I(I(Z)) * 1) ---> Z  [2, 6]

Adding Rule:  [14] (I(I(Z)) * Z1) ---> (Z * Z1)  [11, 4]

  Deleting rule: [11] (I(I(Z)) * 1) ---> Z  [2, 6]

Adding Rule:  [15] (Z * 1) ---> Z  [DELETED, 11]

  Deleting rule:  [4] ((X * 1) * Z) ---> (X * Z)  [1, 3]

  The right hand side is reducible:
      [6] ((X * I(Z)) * Z) ---> (X * 1)  [2, 3]
```

```
  The right hand side is reducible:
     [12] (((X * I((Y * Z))) * Y) * Z) ---> (X * 1)  [3, 6]

  Deleting rule: [13] ((X * I(I(Z))) * 1) ---> (X * Z)  [6, 6]

Adding Rule:  [16] (X * I(I(Z))) ---> (X * Z)  [DELETED, 13]

Computing critical pairs with:
   [15] (Z * 1) ---> Z  [DELETED, 11]

Computing critical pairs with:
   [16] (X * I(I(Z))) ---> (X * Z)  [DELETED, 13]

Adding Rule:  [17] I(I(Z)) ---> Z  [1, 16]

  Deleting rule: [14] (I(I(Z)) * Z1) ---> (Z * Z1)  [11, 4]

  Deleting rule: [16] (X * I(I(Z))) ---> (X * Z)  [DELETED, 13]

Computing critical pairs with:
   [17] I(I(Z)) ---> Z  [1, 16]

Adding Rule:  [18] (Z * I(Z)) ---> 1  [17, 2]

Adding Rule:  [19] ((X * Z) * I(Z)) ---> X  [17, 6]

Computing critical pairs with:
   [18] (Z * I(Z)) ---> 1  [17, 2]

Computing critical pairs with:
   [19] ((X * Z) * I(Z)) ---> X  [17, 6]

Adding Rule:  [20] (((X * Y) * Z) * I((Y * Z))) ---> X  [3, 19]

Computing critical pairs with:
    [5] ((I((Y * Z)) * Y) * Z) ---> 1  [2, 3]

Adding Rule:  [21] (((I(((Y * Y1) * Z)) * Y) * Y1) * Z) ---> 1  [3, 5]

Adding Rule:  [22] (I((Y * I(Z))) * Y) ---> Z  [5, 6]

Adding Rule:  [23] (I((Y * Z)) * Y) ---> I(Z)  [5, 19]

  Deleting rule:  [5] ((I((Y * Z)) * Y) * Z) ---> 1  [2, 3]

  Deleting rule: [22] (I((Y * I(Z))) * Y) ---> Z  [5, 6]

Computing critical pairs with:
   [23] (I((Y * Z)) * Y) ---> I(Z)  [5, 19]

Adding Rule:  [24] ((I(((Y * Z) * Z1)) * Y) * Z) ---> I(Z1)  [3, 23]

  Deleting rule: [21] (((I(((Y * Y1) * Z)) * Y) * Y1) * Z) ---> 1  [3, 5]

Adding Rule:  [25] (I(((Y * Y1) * Z)) * Y) ---> I((Y1 * Z))  [3, 23]

  Deleting rule: [24] ((I(((Y * Z) * Z1)) * Y) * Z) ---> I(Z1)  [3, 23]

Adding Rule:  [26] ((X * I((Z * Z1))) * Z) ---> (X * I(Z1))  [23, 3]

  Deleting rule: [12] (((X * I((Y * Z))) * Y) * Z) ---> X  [3, 6]

Trying to orient equation:
  (I(Z) * Z1) == I((I(Z1) * Z))  [23, 6]
```

```
  To prove:  (I(Z) * Z1)  >  I((I(Z1) * Z))
        or:  (I(Z) * Z1)  <  I((I(Z1) * Z))
  Here are some precedence suggestions:
       1.  * > I      2.  I > *
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, * = I, is added.

----- Step 5 -----

Adding Rule:  [27] I((I(Z) * Z1)) ---> (I(Z1) * Z)  [23, 6]

Adding Rule:  [28] I((Z * Z1)) ---> (I(Z1) * I(Z))  [23, 19]

  Deleting rule: [20] (((X * Y) * Z) * I((Y * Z))) ---> X  [3, 19]

  Deleting rule: [23] (I((Y * Z)) * Y) ---> I(Z)  [5, 19]

  Deleting rule: [25] (I(((Y * Y1) * Z)) * Y) ---> I((Y1 * Z))  [3, 23]

  Deleting rule: [26] ((X * I((Z * Z1))) * Z) ---> (X * I(Z1))  [23, 3]

  Deleting rule: [27] I((I(Z) * Z1)) ---> (I(Z1) * Z)  [23, 6]

Computing critical pairs with:
   [28] I((Z * Z1)) ---> (I(Z1) * I(Z))  [23, 19]

Your system is canonical.

  [1] (1 * X) ---> X  [USER, 1]
  [2] (I(X) * X) ---> 1  [USER, 2]
  [3] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]
  [6] ((X * I(Z)) * Z) ---> X  [2, 3]
 [10] I(1) ---> 1  [2, 9]
 [15] (Z * 1) ---> Z  [DELETED, 11]
 [17] I(I(Z)) ---> Z  [1, 16]
 [18] (Z * I(Z)) ---> 1  [17, 2]
 [19] ((X * Z) * I(Z)) ---> X  [17, 6]
 [28] I((Z * Z1)) ---> (I(Z1) * I(Z))  [23, 19]

Time used                            =   1.55 sec
Number of rules generated            = 28
Number of rules retained             = 10
Number of critical pairs             = 101
Time spent in normalization          =   0.37 sec  (23.0 percent of time)
Time spent in unification            =   0.13 sec   (8.0 percent of time)
Time spent in ordering               =   0.31 sec   (20.0 percent of time)
Time spent in simplifying the rules  =   0.39 sec   (25.0 percent of time)
Total time (including 'undo' action) = 158.0 sec


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

5 (l,r) Systems

5.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 5.1 *A (l,r) system LR is a groupoid together with a new unary operation* $^{-1}$*, i.e.* $^{-1}: S \to S$*, fulfilling the following axioms:*

$$\begin{aligned} 1 \cdot x &= x & (1)\\ x \cdot x^{-1} &= 1 & (2)\\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) & (3) \end{aligned}$$

The left identity like in groups remained but now there exist right inverses instead of left inverses. All together there would be 4 combinations.

5.2 Example

We can take the same model like that for groups.

5.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical systems for (l,r) systems:

$$\begin{aligned} 1 \cdot x &\longrightarrow x & (1)\\ x \cdot x^{-1} &\longrightarrow 1 & (2)\\ 1^{-1} &\longrightarrow 1 & (3)\\ x \cdot (y \cdot z) &\longrightarrow (x \cdot y) \cdot z & (4)\\ (x \cdot 1) \cdot z &\longrightarrow x \cdot z & (5)\\ (x \cdot y) \cdot y^{-1} &\longrightarrow x \cdot 1 & (6)\\ (x^{-1})^{-1} &\longrightarrow x \cdot 1 & (7)\\ (x^{-1} \cdot x) \cdot z &\longrightarrow z & (8)\\ ((x \cdot y^{-1}) \cdot y) \cdot z &\longrightarrow x \cdot z & (9)\\ x^{-1} \cdot 1 &\longrightarrow x^{-1} & (10)\\ (x \cdot y^{-1}) \cdot 1 &\longrightarrow x \cdot y^{-1} & (11)\\ (x \cdot y)^{-1} &\longrightarrow y^{-1} \cdot x^{-1} & (12) \end{aligned}$$

In [Hul80] another canonical system is given, but this does not mean a problem, since there might be several canonical systems.

5.4 Trace

The operation $\cdot$ is represented by $*$ and $^{-1}$ by I(_), resp., 1 remains the same.

```
RRL-> read test/5.eq
New constant set is: { 1 }
Equations read in are:
 1. (1 * X) == X  [USER, 1]
 2. (X * I(X)) == 1  [USER, 2]
 3. ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] (1 * X) ---> X  [USER, 1]

Adding Rule:   [2] (X * I(X)) ---> 1  [USER, 2]

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 2 -----

Computing critical pairs with:
    [1] (1 * X) ---> X  [USER, 1]

Computing critical pairs with:
    [2] (X * I(X)) ---> 1  [USER, 2]

Adding Rule:   [3] I(1) ---> 1  [1, 2]

Computing critical pairs with:
    [3] I(1) ---> 1  [1, 2]

Processing all postponed equations ...

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 3 -----

Trying to orient equation:
```

```
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Operator, *, given status: rl

----- Step 4 -----

Adding Rule:   [4] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]

Computing critical pairs with:
    [4] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]

Adding Rule:   [5] ((X * 1) * Z) ---> (X * Z)  [1, 4]

Adding Rule:   [6] ((X * Y) * I(Y)) ---> (X * 1)  [2, 4]

Computing critical pairs with:
    [5] ((X * 1) * Z) ---> (X * Z)  [1, 4]

Adding Rule:   [7] (X * I((X * 1))) ---> 1  [2, 5]

Computing critical pairs with:
    [7] (X * I((X * 1))) ---> 1  [2, 5]

Adding Rule:   [8] ((X * Y) * I((Y * 1))) ---> (X * 1)  [7, 4]

Computing critical pairs with:
    [6] ((X * Y) * I(Y)) ---> (X * 1)  [2, 4]

Trying to orient equation:
  I(I(X)) == (X * 1)  [2, 6]

  To prove:  I(I(X))  >  (X * 1)
        or:  I(I(X))  <  (X * 1)
  Here are some precedence suggestions:
       1.  I > *      2.  * > I
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, I = *, is added.

----- Step 5 -----

Adding Rule:   [9] I(I(X)) ---> (X * 1)  [2, 6]

Adding Rule:  [10] (((X * Y) * Z) * I((Y * Z))) ---> (X * 1)  [4, 6]

Computing critical pairs with:
    [9] I(I(X)) ---> (X * 1)  [2, 6]

Adding Rule:  [11] I((X * 1)) ---> (I(X) * 1)  [9, 9]

  Deleting rule:  [7] (X * I((X * 1))) ---> 1  [2, 5]

  Deleting rule:  [8] ((X * Y) * I((Y * 1))) ---> (X * 1)  [7, 4]

Adding Rule:  [12] ((I(X) * X) * 1) ---> 1  [9, 2]
```

```
Adding Rule:  [13] (((X * I(X1)) * X1) * 1) ---> (X * 1)  [9, 6]

Computing critical pairs with:
   [11] I((X * 1)) ---> (I(X) * 1)  [9, 9]

Computing critical pairs with:
   [12] ((I(X) * X) * 1) ---> 1  [9, 2]

Adding Rule:  [14] (I((I(X) * X)) * 1) ---> 1  [12, 11]

Adding Rule:  [15] (((I((Y * Z)) * Y) * Z) * 1) ---> 1  [4, 12]

Adding Rule:  [16] ((I(X) * X) * Z) ---> Z  [12, 5]

  Deleting rule: [12] ((I(X) * X) * 1) ---> 1  [9, 2]

Computing critical pairs with:
   [16] ((I(X) * X) * Z) ---> Z  [12, 5]

Adding Rule:  [17] I((I(X) * X)) ---> 1  [2, 16]

  Deleting rule: [14] (I((I(X) * X)) * 1) ---> 1  [12, 11]

Adding Rule:  [18] (((I((Y * Z)) * Y) * Z) * Z1) ---> Z1  [4, 16]

  Deleting rule: [15] (((I((Y * Z)) * Y) * Z) * 1) ---> 1  [4, 12]

Adding Rule:  [19] (((X * I(X1)) * X1) * Z) ---> (X * Z)  [16, 4]

  Deleting rule: [13] (((X * I(X1)) * X1) * 1) ---> (X * 1)  [9, 6]

Adding Rule:  [20] (I(X) * 1) ---> I(X)  [6, 16]

  The right hand side is reducible:
     [11] I((X * 1)) ---> (I(X) * 1)  [9, 9]

Computing critical pairs with:
   [20] (I(X) * 1) ---> I(X)  [6, 16]

Adding Rule:  [21] ((X * I(X1)) * 1) ---> (X * I(X1))  [20, 4]

Computing critical pairs with:
   [17] I((I(X) * X)) ---> 1  [2, 16]

Adding Rule:  [22] I(((I((Y * Z)) * Y) * Z)) ---> 1  [4, 17]

Computing critical pairs with:
   [21] ((X * I(X1)) * 1) ---> (X * I(X1))  [20, 4]

Computing critical pairs with:
   [19] (((X * I(X1)) * X1) * Z) ---> (X * Z)  [16, 4]

Adding Rule:  [23] (X * I(((X * I(X1)) * X1))) ---> 1  [2, 19]

Adding Rule:  [24] I(((X * I(X1)) * X1)) ---> I(X)  [19, 11]

  Deleting rule: [23] (X * I(((X * I(X1)) * X1))) ---> 1  [2, 19]

Adding Rule:  [25] ((((X * I((Y * Z))) * Y) * Z) * Z1) ---> (X * Z1)  [4, 19]

Computing critical pairs with:
   [24] I(((X * I(X1)) * X1)) ---> I(X)  [19, 11]

Adding Rule:  [26] I((((X * I((Y * Z))) * Y) * Z)) ---> I(X)  [4, 24]
```

```
Computing critical pairs with:
   [22] I(((I((Y * Z)) * Y) * Z)) ---> 1  [4, 17]

Adding Rule:  [27] I((((I(((Y * Y1) * Z)) * Y) * Y1) * Z)) ---> 1  [4, 22]

Computing critical pairs with:
   [18] (((I((Y * Z)) * Y) * Z) * Z1) ---> Z1  [4, 16]

Adding Rule:  [28] ((((I(((Y * Y1) * Z)) * Y) * Y1) * Z) * Z2) ---> Z2  [4, 18]

Adding Rule:  [29] ((I((Y * Z)) * Y) * 1) ---> I(Z)  [6, 18]

Adding Rule:  [30] ((I((Y * I(X))) * Y) * I(X)) ---> 1  [21, 18]

Adding Rule:  [31] I((I((Y * I(X))) * Y)) ---> I(X)  [18, 24]

Adding Rule:  [32] ((I((Y * I(X))) * Y) * Z) ---> (X * Z)  [18, 19]

  Deleting rule: [30] ((I((Y * I(X))) * Y) * I(X)) ---> 1  [21, 18]

Computing critical pairs with:
   [31] I((I((Y * I(X))) * Y)) ---> I(X)  [18, 24]

Adding Rule:  [33] I((I((Y * X)) * Y)) ---> (X * 1)  [9, 31]

  Deleting rule: [31] I((I((Y * I(X))) * Y)) ---> I(X)  [18, 24]

Computing critical pairs with:
   [33] I((I((Y * X)) * Y)) ---> (X * 1)  [9, 31]

Adding Rule:  [34] I(((I(((Y * Z) * X)) * Y) * Z)) ---> (X * 1)  [4, 33]

Adding Rule:  [35] I((I(((Y * Y1) * Z)) * Y)) ---> ((Y1 * Z) * 1)  [4, 33]

Adding Rule:  [36] I(((X * I((Y * X))) * Y)) ---> 1  [33, 17]

Adding Rule:  [37] (((X * I((Y * X))) * Y) * Z) ---> Z  [33, 16]

Adding Rule:  [38] I((((X * X1) * I((Y * X1))) * Y)) ---> I(X)  [33, 24]

Adding Rule:  [39] I((X * I((X1 * X)))) ---> (X1 * 1)  [33, 33]

Adding Rule:  [40] ((((X * X1) * I((Y * X1))) * Y) * Z) ---> (X * Z)  [33, 19]

Computing critical pairs with:
   [39] I((X * I((X1 * X)))) ---> (X1 * 1)  [33, 33]

Adding Rule:  [41] (X * I((X1 * X))) ---> I(X1)  [39, 9]

  Deleting rule: [36] I(((X * I((Y * X))) * Y)) ---> 1  [33, 17]

  Deleting rule: [37] (((X * I((Y * X))) * Y) * Z) ---> Z  [33, 16]

  Deleting rule: [39] I((X * I((X1 * X)))) ---> (X1 * 1)  [33, 33]

Computing critical pairs with:
   [41] (X * I((X1 * X))) ---> I(X1)  [39, 9]

Adding Rule:  [42] ((Y * Z) * I(((X * Y) * Z))) ---> I(X)  [4, 41]

Adding Rule:  [43] ((X * Y) * I((X1 * Y))) ---> (X * I(X1))  [41, 4]

  Deleting rule: [10] (((X * Y) * Z) * I((Y * Z))) ---> (X * 1)  [4, 6]
```

```
  Deleting rule: [38] I((((X * X1) * I((Y * X1))) * Y)) ---> I(X)  [33, 24]

  Deleting rule: [40] ((((X * X1) * I((Y * X1))) * Y) * Z) ---> (X * Z)  [33, 19]

  Deleting rule: [42] ((Y * Z) * I(((X * Y) * Z))) ---> I(X)  [4, 41]

Adding Rule:  [44] I((X * Y)) ---> (I(Y) * I(X))  [6, 41]

  Deleting rule: [11] I((X * 1)) ---> I(X)  [9, 9]

  Deleting rule: [17] I((I(X) * X)) ---> 1  [2, 16]

  Deleting rule: [18] (((I((Y * Z)) * Y) * Z) * Z1) ---> Z1  [4, 16]

  Deleting rule: [22] I(((I((Y * Z)) * Y) * Z)) ---> 1  [4, 17]

  Deleting rule: [24] I(((X * I(X1)) * X1)) ---> I(X)  [19, 11]

  Deleting rule: [25] ((((X * I((Y * Z))) * Y) * Z) * Z1) ---> (X * Z1)  [4, 19]

  Deleting rule: [26] I((((X * I((Y * Z))) * Y) * Z)) ---> I(X)  [4, 24]

  Deleting rule: [27] I((((I(((Y * Y1) * Z)) * Y) * Y1) * Z)) ---> 1  [4, 22]

  Deleting rule: [28] ((((I(((Y * Y1) * Z)) * Y) * Y1) * Z) * Z2) ---> Z2  [4, 18]

  Deleting rule: [29] ((I((Y * Z)) * Y) * 1) ---> I(Z)  [6, 18]

  Deleting rule: [32] ((I((Y * I(X))) * Y) * Z) ---> (X * Z)  [18, 19]

  Deleting rule: [33] I((I((Y * X)) * Y)) ---> (X * 1)  [9, 31]

  Deleting rule: [34] I(((I(((Y * Z) * X)) * Y) * Z)) ---> (X * 1)  [4, 33]

  Deleting rule: [35] I((I(((Y * Y1) * Z)) * Y)) ---> ((Y1 * Z) * 1)  [4, 33]

  Deleting rule: [41] (X * I((X1 * X))) ---> I(X1)  [39, 9]

  Deleting rule: [43] ((X * Y) * I((X1 * Y))) ---> (X * I(X1))  [41, 4]

Computing critical pairs with:
   [44] I((X * Y)) ---> (I(Y) * I(X))  [6, 41]

Your system is canonical.

  [1] (1 * X) ---> X  [USER, 1]
  [2] (X * I(X)) ---> 1  [USER, 2]
  [3] I(1) ---> 1  [1, 2]
  [4] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 3]
  [5] ((X * 1) * Z) ---> (X * Z)  [1, 4]
  [6] ((X * Y) * I(Y)) ---> (X * 1)  [2, 4]
  [9] I(I(X)) ---> (X * 1)  [2, 6]
 [16] ((I(X) * X) * Z) ---> Z  [12, 5]
 [19] (((X * I(X1)) * X1) * Z) ---> (X * Z)  [16, 4]
 [20] (I(X) * 1) ---> I(X)  [6, 16]
 [21] ((X * I(X1)) * 1) ---> (X * I(X1))  [20, 4]
 [44] I((X * Y)) ---> (I(Y) * I(X))  [6, 41]

Time used                          =   5.15 sec
Number of rules generated          = 44
Number of rules retained           = 12
Number of critical pairs           = 264
Time spent in normalization        =   1.70 sec  (33.0 percent of time)
Time spent in unification          =   0.37 sec   (7.0 percent of time)
Time spent in ordering             =   0.47 sec   (9.0 percent of time)
```

```
Time spent in simplifying the rules  =    1.82 sec    (35.0 percent of time)
Total time (including ´undo´ action) = 522.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/5.eq
New constant set is: { 1 }
Equations read in are:
 1. (1 * X) == X  [USER, 1]
 2. (X * I(X)) == 1  [USER, 2]
 3. ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
Adding Rule:   [1] (1 * X) ---> X  [USER, 1]

Adding Rule:   [2] (X * I(X)) ---> 1  [USER, 2]

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> status * lr
Operator, *, given status: LR

----- Step 1 -----

Adding Rule:   [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 2 -----

Computing critical pairs with:
    [1] (1 * X) ---> X  [USER, 1]

Computing critical pairs with:
    [2] (X * I(X)) ---> 1  [USER, 2]

Adding Rule:   [4] I(1) ---> 1  [1, 2]

Computing critical pairs with:
    [4] I(1) ---> 1  [1, 2]
```

```
Computing critical pairs with:
    [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]

Adding Rule:   [5] (X * (Y * I((X * Y)))) ---> 1  [2, 3]

Adding Rule:   [6] (X * (I(X) * Z)) ---> Z  [2, 3]

Computing critical pairs with:
    [6] (X * (I(X) * Z)) ---> Z  [2, 3]

Trying to orient equation:
  (X * 1) == I(I(X))  [2, 6]

  To prove:  (X * 1)  >  I(I(X))
        or:  (X * 1)  <  I(I(X))
  Here are some precedence suggestions:
       1.  * > I      2.  I > *
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, * = I, is added.

----- Step 3 -----

Adding Rule:   [7] I(I(X)) ---> (X * 1)  [2, 6]

Adding Rule:   [8] (X * (Y * (I((X * Y)) * Z))) ---> Z  [3, 6]

Computing critical pairs with:
    [7] I(I(X)) ---> (X * 1)  [2, 6]

Adding Rule:   [9] I((X * 1)) ---> (I(X) * 1)  [7, 7]

Adding Rule:  [10] (I(X) * (X * 1)) ---> 1  [7, 2]

Adding Rule:  [11] (I(X) * (X * Z)) ---> Z  [7, 6]

  Deleting rule: [10] (I(X) * (X * 1)) ---> 1  [7, 2]

Computing critical pairs with:
    [9] I((X * 1)) ---> (I(X) * 1)  [7, 7]

Trying to orient equation:
  I((X * (Y * 1))) == (I((X * Y)) * 1)  [3, 9]

  To prove:  I((X * (Y * 1)))  >  (I((X * Y)) * 1)
        or:  I((X * (Y * 1)))  <  (I((X * Y)) * 1)
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

Computing critical pairs with:
   [11] (I(X) * (X * Z)) ---> Z  [7, 6]

Adding Rule:  [12] (I(X) * 1) ---> I(X)  [2, 11]

  The right hand side is reducible:
      [9] I((X * 1)) ---> (I(X) * 1)  [7, 7]
```

```
Adding Rule:  [13] (I((X * Y)) * (X * (Y * Z))) ---> Z  [3, 11]

Computing critical pairs with:
   [12] (I(X) * 1) ---> I(X)  [2, 11]

Computing critical pairs with:
    [5] (X * (Y * I((X * Y)))) ---> 1  [2, 3]

Adding Rule:  [14] (X * (Y * (Y1 * I((X * (Y * Y1)))))) ---> 1  [3, 5]

Adding Rule:  [15] (Y * I((I(X) * Y))) ---> (X * 1)  [5, 6]

Adding Rule:  [16] I((I(Y) * Y)) ---> 1  [11, 5]

Adding Rule:  [17] (Y * I((X * Y))) ---> I(X)  [5, 11]

  Deleting rule:  [5] (X * (Y * I((X * Y)))) ---> 1  [2, 3]

  Deleting rule: [15] (Y * I((I(X) * Y))) ---> (X * 1)  [5, 6]

Computing critical pairs with:
   [16] I((I(Y) * Y)) ---> 1  [11, 5]

Computing critical pairs with:
   [17] (Y * I((X * Y))) ---> I(X)  [5, 11]

Adding Rule:  [18] (X * (Y * I((X1 * (X * Y))))) ---> I(X1)  [3, 17]

  Deleting rule: [14] (X * (Y * (Y1 * I((X * (Y * Y1)))))) ---> 1  [3, 5]

Adding Rule:  [19] (Y * I((X * (Y1 * Y)))) ---> I((X * Y1))  [3, 17]

  Deleting rule: [18] (X * (Y * I((X1 * (X * Y))))) ---> I(X1)  [3, 17]

Adding Rule:  [20] (X * (I((X1 * X)) * Z)) ---> (I(X1) * Z)  [17, 3]

  Deleting rule:  [8] (X * (Y * (I((X * Y)) * Z))) ---> Z  [3, 6]

Adding Rule:  [21] I((X * I(X1))) ---> (X1 * I(X))  [17, 6]

Adding Rule:  [22] I((X * X1)) ---> (I(X1) * I(X))  [17, 11]

  Deleting rule:  [9] I((X * 1)) ---> I(X)  [7, 7]

  Deleting rule: [13] (I((X * Y)) * (X * (Y * Z))) ---> Z  [3, 11]

  Deleting rule: [16] I((I(Y) * Y)) ---> 1  [11, 5]

  Deleting rule: [17] (Y * I((X * Y))) ---> I(X)  [5, 11]

  Deleting rule: [19] (Y * I((X * (Y1 * Y)))) ---> I((X * Y1))  [3, 17]

  Deleting rule: [20] (X * (I((X1 * X)) * Z)) ---> (I(X1) * Z)  [17, 3]

  Deleting rule: [21] I((X * I(X1))) ---> (X1 * I(X))  [17, 6]

Computing critical pairs with:
   [22] I((X * X1)) ---> (I(X1) * I(X))  [17, 11]

Processing all postponed equations ...

Your system is canonical.

  [1] (1 * X) ---> X  [USER, 1]
  [2] (X * I(X)) ---> 1  [USER, 2]
```

```
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] I(1) ---> 1  [1, 2]
  [6] (X * (I(X) * Z)) ---> Z  [2, 3]
  [7] I(I(X)) ---> (X * 1)  [2, 6]
 [11] (I(X) * (X * Z)) ---> Z  [7, 6]
 [12] (I(X) * 1) ---> I(X)  [2, 11]
 [22] I((X * X1)) ---> (I(X1) * I(X))  [17, 11]

Time used                           =   1.21 sec
Number of rules generated           = 22
Number of rules retained            = 9
number of critical pairs            = 89
time spent in normalization         =   0.39 sec  (32.0 percent of time)
time spent in unification           =   0.07 sec   (5.0 percent of time)
time spent in ordering              =   1.22 sec   (100.0 percent of time)
time spent in simplifying the rules  =   0.33 sec   (27.0 percent of time)
total time (including 'undo' action) = 124.0 sec



type add, akb, auto, break, clean, delete, grammar, history, init, kb, list,
     load, log, makerule, narrow, norm, option, operator, prove, quit, read,
     refute, save, stats, suffice, undo, unlog, write or help.
rrl-> quit
```

6 Commutative Semigroups and Monoids

6.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 6.1 *A commutative semigroup CS is a semigroup where besides*

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \tag{1}$$

also

$$x \cdot y = y \cdot x \tag{2}$$

holds.

Definition 6.2 *A commutative monoid CM is a commutative semigroup with the additional axiom:*

$$1 \cdot x = x \tag{3}$$

Definition 6.3 *An idempotent commutative semigroup ICS is a commutative semigroup with the additional axiom:*

$$x \cdot x = x \tag{3}$$

6.2 Example

A model of a commutative semigroup are the positive integers with addition. If we add zero then we will get a model of a commutative monoid, where 1 is just zero. A model of an idempotent commutative semigroup is the set of integers again with the min (or max) operation.

6.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

In the following we assume that there is an AC rewrite engine buildt into the rewrite mechanism, i.e. the systems performs associative and commutative rewritings automatically.

Then in the case of commutative semigroups it is only left to state that $\cdot$ is ac (associative and commutative). In the other examples of this section the same is true.

There is the following canonical system for commutative monoids (with $\cdot$ ac):

$$1 \cdot x \longrightarrow x \tag{1}$$

There is the following canonical system for idempotent commutative semigroups (with $\cdot$ ac):

$$x \cdot x \longrightarrow x \tag{1}$$

6.4 Trace

The operation $\cdot$ is represented by $*$, 1 remains the same.

```
RRL-> read test/6a.eq
Equations read in are:
 1. ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]
 2. (X * Y) == (Y * X)  [USER, 2]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 2 -----

----- Step 3 -----

´*´ has the associative and commutative property now:
    (X * Y) == (Y * X)  [USER, 2]

Processing all postponed equations ...

Your system is canonical.


Time used                          =   0.06 sec
Number of rules generated          = 0
Number of rules retained           = 0
Number of critical pairs           = 0
Number of unblocked critical pairs  = 0
Time spent in normalization        =   0.01 sec  (16.0 percent of time)
Time spent in unification          =   0.00 sec   (0.0 percent of time)
Time spent in ordering             =   0.05 sec   (83.0 percent of time)
Time spent in simplifying the rules  =   0.00 sec   (0.0 percent of time)
Time spent in blocking             =   0.00 sec   (0.0 percent of time)
Total time (including ´undo´ action) = 6.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/6b.eq
New constant set is: { 1 }
Equations read in are:
 1. (1 * X) == X  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
```

```
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 4 -----

Adding Rule:   [1] (X * 1) ---> X  [USER, 3]

Your system is canonical.

  [1] (X * 1) ---> X  [USER, 3]

Time used                          =   0.11 sec
Number of rules generated          = 1
Number of rules retained           = 1
Number of critical pairs           = 3
Number of unblocked critical pairs  = 0
Time spent in normalization         =   0.01 sec  (9.0 percent of time)
Time spent in unification           =   0.03 sec   (27.0 percent of time)
Time spent in ordering              =   0.05 sec   (45.0 percent of time)
Time spent in simplifying the rules  =   0.01 sec   (9.0 percent of time)
Time spent in blocking              =   0.00 sec   (0.0 percent of time)
Total time (including ´undo´ action) = 6.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/6a.eq
Equations read in are:
 1. ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]
 2. (X * Y) == (Y * X)  [USER, 2]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/6c.eq
Equations read in are:
 1. (X * X) == X  [USER, 3]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 1]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
```

```
The equation is postponed.

----- Step 2 -----

----- Step 3 -----

´*´ has the associative and commutative property now:
    (X * Y) == (Y * X)  [USER, 2]

Adding Rule:   [1] (X * X) ---> X  [USER, 3]

Processing all postponed equations ...

Your system is canonical.

  [1] (X * X) ---> X  [USER, 3]

Time used                           =   0.18 sec
Number of rules generated           = 1
Number of rules retained            = 1
Number of critical pairs            = 3
Number of unblocked critical pairs  = 0
Time spent in normalization         =   0.02 sec  (11.0 percent of time)
Time spent in unification           =   0.07 sec   (38.0 percent of time)
Time spent in ordering              =   0.06 sec   (33.0 percent of time)
Time spent in simplifying the rules =   0.01 sec   (5.0 percent of time)
Time spent in blocking              =   0.00 sec   (0.0 percent of time)
Total time (including ´undo´ action) = 18.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

7 Abelian Groups

7.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S. Here we will denote the operation $\cdot$ used in groupoids with $+$.

Definition 7.1 *An Abelian group AG is a groupoid with an additional unary operation $-$, i.e. $- : S \to S$ fulfilling the following axioms:*

$$
\begin{aligned}
x + y &= y + x && (1)\\
(x + y) + z &= x + (y + z) && (2)\\
x + 0 &= x && (3)\\
x + (-x) &= 0 && (4)
\end{aligned}
$$

7.2 Example

A model of an Abelian group is the set of all integers with plus as the + operation and the negation as the inverse operation. 0 is represented by zero.

7.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical system for Abelian groups (with + ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(z)) &\longrightarrow z && (4)\\
-(x + y) &\longrightarrow -(x) + -(y) && (5)
\end{aligned}
$$

In [Hul80] also a case with orienting the last rule from right to left is considered. In that case the result is the following canonical system (+ ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(z)) &\longrightarrow z && (4)\\
z + -(z + y) &\longrightarrow -(y) && (5)\\
-(z) + -(y) &\longrightarrow -(z + y) && (6)\\
-(z + -(y)) &\longrightarrow y + -(z) && (7)
\end{aligned}
$$

7.4 Trace

The operations + and − remain the same, also 0.

```
RRL-> read test/7.eq
New constant set is: { 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

'+' has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]

----- Step 3 -----

'+' has the associative and commutative property now:
    (Z + (X + Y)) == (X + (Y + Z))  [USER, 2]

Adding Rule:   [1] (X + 0) ---> X  [USER, 3]

Adding Rule:   [2] (X + -(X)) ---> 0  [USER, 4]

Adding Rule:   [3] -(0) ---> 0  [2, 1]

Adding Rule:   [4] -(-(Z)) ---> Z  [2, 2]

Adding Rule:   [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]

Trying to orient equation:
  (-(Z) + -(Z1)) == -((Z + Z1))  [2, 5]

  To prove:  (-(Z) + -(Z1))  >  -((Z + Z1))
        or:  (-(Z) + -(Z1))  <  -((Z + Z1))
  Here are some precedence suggestions:
       1.  + > -       2.  - > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, + = -, is added.

----- Step 4 -----

Adding Rule:   [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]

  Deleting rule:  [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]
```

```
Time used                          =   2.52 sec
Number of rules generated          = 6
Number of rules retained           = 5
Number of critical pairs           = 51
Number of unblocked critical pairs  = 29
Time spent in normalization        =   0.92 sec  (36.0 percent of time)
Time spent in unification          =   0.81 sec   (32.0 percent of time)
Time spent in ordering             =   0.12 sec   (4.0 percent of time)
Time spent in simplifying the rules =   0.07 sec   (2.0 percent of time)
Time spent in blocking             =   0.18 sec   (7.0 percent of time)
Total time (including 'undo' action) = 256.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/7.eq
New constant set is: { 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> kb
----- Step 1 -----

----- Step 2 -----

'+' has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]

----- Step 3 -----

'+' has the associative and commutative property now:
    (Z + (X + Y)) == (X + (Y + Z))  [USER, 2]

Adding Rule:   [1] (X + 0) ---> X  [USER, 3]

Adding Rule:   [2] (X + -(X)) ---> 0  [USER, 4]

Adding Rule:   [3] -(0) ---> 0  [2, 1]

Adding Rule:   [4] -(-(Z)) ---> Z  [2, 2]

Adding Rule:   [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]

Trying to orient equation:
  (-(Z) + -(Z1)) == -((Z + Z1))  [2, 5]

  To prove:  (-(Z) + -(Z1))  >  -((Z + Z1))
        or:  (-(Z) + -(Z1))  <  -((Z + Z1))
  Here are some precedence suggestions:
```

```
      1.  + > -      2.  - > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 4 -----

Adding Rule:   [6] (-(Z) + -(Z1)) ---> -((Z + Z1))  [2, 5]

Adding Rule:   [7] -((Z + -(Z1))) ---> (Z1 + -(Z))  [2, 5]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]
  [6] (-(Z) + -(Z1)) ---> -((Z + Z1))  [2, 5]
  [7] -((Z + -(Z1))) ---> (Z1 + -(Z))  [2, 5]

Time used                            =   7.51 sec
Number of rules generated            = 7
Number of rules retained             = 7
Number of critical pairs             = 114
Number of unblocked critical pairs   = 44
Time spent in normalization          =   3.94 sec  (52.0 percent of time)
Time spent in unification            =   1.66 sec   (22.0 percent of time)
Time spent in ordering               =   0.10 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   0.11 sec   (1.0 percent of time)
Time spent in blocking               =   0.62 sec   (8.0 percent of time)
Total time (including 'undo' action) = 754.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

8 Rings

8.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 8.1 *A ring R is an Abelian group with the axioms:*

$$\begin{aligned} x+y &= y+x && (1)\\ (x+y)+z &= x+(y+z) && (2)\\ x+0 &= x && (3)\\ x+(-x) &= 0 && (4) \end{aligned}$$

and one additional binary operation $*$, i.e. $* : S \times S \to S$ fulfilling furthermore the axioms:

$$\begin{aligned} x*(y+z) &= (x*y)+(x*z) && (5)\\ (x+y)*z &= (x*z)+(y*z) && (6) \end{aligned}$$

Definition 8.2 *A ring RU with unit 1 is a ring fulfilling the additional axioms:*

$$\begin{aligned} 1*x &= x && (7)\\ x*1 &= x && (8) \end{aligned}$$

Definition 8.3 *An associative ring ARU with unit 1 is a ring with unit 1 fulfilling the additional axiom:*

$$(x*y)*z = x*(y*z) \qquad (9)$$

Definition 8.4 *An associative and commutative ring ACRU with unit 1 is an accociative ring with unit 1 fulfilling the additional axiom:*

$$x*y = y*x \qquad (10)$$

8.2 Example

A model of an associative and commutative ring with unit consists of the integers with addition, negation and multiplication. 0 is zero and 1 is one.

8.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical system for rings (+ ac):

$$\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(x)) &\longrightarrow x && (4)\\
-(z + y) &\longrightarrow -(z) + -(y) && (5)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (6)\\
(x + y) * z &\longrightarrow (x * z) + (y * z) && (7)\\
x * 0 &\longrightarrow 0 && (8)\\
0 * z &\longrightarrow 0 && (9)\\
x * -(z) &\longrightarrow -(x * z) && (10)\\
-(z) * y &\longrightarrow -(z * y) && (11)
\end{aligned}$$

There is the following canonical system for rings with unit (+ ac):

$$\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(z)) &\longrightarrow z && (4)\\
-(z + y) &\longrightarrow -(z) + -(y) && (5)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (6)\\
(x + y) * z &\longrightarrow (x * z) + (y * z) && (7)\\
x * 0 &\longrightarrow 0 && (8)\\
0 * z &\longrightarrow 0 && (9)\\
x * -(z) &\longrightarrow -(x * z) && (10)\\
-(z) * y &\longrightarrow -(z * y) && (11)\\
1 * x &\longrightarrow x && (12)\\
x * 1 &\longrightarrow x && (13)
\end{aligned}$$

There is the following canonical system for associative rings with unit (+ ac):

$$\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(z)) &\longrightarrow z && (4)\\
-(z + y) &\longrightarrow -(z) + -(y) && (5)
\end{aligned}$$

$$
\begin{aligned}
x * (y + z) &\longrightarrow (x * y) + (x * z) && (6)\\
(x + y) * z &\longrightarrow (x * z) + (y * z) && (7)\\
x * 0 &\longrightarrow 0 && (8)\\
0 * z &\longrightarrow 0 && (9)\\
x * -(z) &\longrightarrow -(x * z) && (10)\\
-(z) * y &\longrightarrow -(z * y) && (11)\\
1 * x &\longrightarrow x && (12)\\
x * 1 &\longrightarrow x && (13)\\
x * (y * z) &\longrightarrow (x * y) * z && (14)
\end{aligned}
$$

There is the following canonical system for associative and commutative rings with unit ($+$ and $*$ ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
-(0) &\longrightarrow 0 && (3)\\
-(-(z)) &\longrightarrow z && (4)\\
-(x + y) &\longrightarrow -(x) + -(y) && (5)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (6)\\
x * 0 &\longrightarrow 0 && (7)\\
x * -(z) &\longrightarrow -(x * z) && (8)\\
x * 1 &\longrightarrow x && (9)
\end{aligned}
$$

8.4 Trace

The operations $+$, $-$ and $*$ remain the same, also 0 and 1.

```
RRL-> read test/8a.eq
New constant set is: { 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

'+' has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]
```

```
----- Step 3 -----

´+´ has the associative and commutative property now:
    (Z + (X + Y)) == (X + (Y + Z))  [USER, 2]

Adding Rule:   [1] (X + 0) ---> X  [USER, 3]

Adding Rule:   [2] (X + -(X)) ---> 0  [USER, 4]

Adding Rule:   [3] -(0) ---> 0  [2, 1]

Adding Rule:   [4] -(-(Z)) ---> Z  [2, 2]

Adding Rule:   [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]

Trying to orient equation:
  (-(Z) + -(Z1)) == -((Z + Z1))  [2, 5]

  To prove:  (-(Z) + -(Z1))  >  -((Z + Z1))
        or:  (-(Z) + -(Z1))  <  -((Z + Z1))
  Here are some precedence suggestions:
       1.  + > -      2.  - > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, + = -, is added.

----- Step 4 -----

Adding Rule:   [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]

  Deleting rule:  [5] (Z + -((Z + Z1))) ---> -(Z1)  [2, 2]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]

Time used                          =   2.47 sec
Number of rules generated          = 6
Number of rules retained           = 5
Number of critical pairs           = 51
Number of unblocked critical pairs = 29
Time spent in normalization        =   0.84 sec  (34.0 percent of time)
Time spent in unification          =   0.79 sec   (31.0 percent of time)
Time spent in ordering             =   0.12 sec   (4.0 percent of time)
Time spent in simplifying the rules =   0.07 sec   (2.0 percent of time)
Time spent in blocking             =   0.21 sec   (8.0 percent of time)
Total time (including ´undo´ action) = 249.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/8b.eq
Equations read in are:
 1. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 5]
 2. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 6]
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 5 -----

Trying to orient equation:
  (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 5]

  To prove:  (X * (Y + Z))  >  ((X * Y) + (X * Z))
        or:  (X * (Y + Z))  <  ((X * Y) + (X * Z))
  Here are some precedence suggestions:
       1.  * > +      2.  + > *
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, * > +, is added.

----- Step 6 -----

Adding Rule:   [7] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 5]

Adding Rule:   [8] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 6]

Adding Rule:   [9] ((X * Z) + (X * 0)) ---> (X * Z)  [1, 7]

Adding Rule:  [10] ((Z * Z1) + (0 * Z1)) ---> (Z * Z1)  [1, 8]

Adding Rule:  [11] ((X * Z) + (X * -(Z))) ---> (X * 0)  [2, 7]

Adding Rule:  [12] ((Z * Z1) + (-(Z) * Z1)) ---> (0 * Z1)  [2, 8]

Adding Rule:  [13] (-((X * Z)) + -((X * 0))) ---> -((X * Z))  [9, 6]

Adding Rule:  [14] (-((Z * Z1)) + -((0 * Z1))) ---> -((Z * Z1))  [10, 6]

Adding Rule:  [15] ((X * (X1 * Z)) + (X * (X1 * 0))) ---> (X * (X1 * Z))  [9, 7]

Adding Rule:  [16] (((X * Z) * Z1) + ((X * 0) * Z1)) ---> ((X * Z) * Z1)  [9, 8]

Adding Rule:  [17] ((X * (Z * Z1)) + (X * (0 * Z1))) ---> (X * (Z * Z1))  [10, 7]

Adding Rule:  [18] (((Z * Z1) * Z2) + ((0 * Z1) * Z2)) ---> ((Z * Z1) * Z2)  [10, 8]

Adding Rule:  [19] (-((X * Z)) + -((X * -(Z)))) ---> -((X * 0))  [11, 6]

Adding Rule:  [20] (-((Z * Z1)) + -((-(Z) * Z1))) ---> -((0 * Z1))  [12, 6]

Adding Rule:  [21] (-((X * 0)) + (X * Z)) ---> (X * Z)  [9, 2]

Adding Rule:  [22] (X * 0) ---> 0  [9, 2]

  Deleting rule:  [9] ((X * Z) + (X * 0)) ---> (X * Z)  [1, 7]

  The right hand side is reducible:
     [11] ((X * Z) + (X * -(Z))) ---> (X * 0)  [2, 7]

  Deleting rule: [13] (-((X * Z)) + -((X * 0))) ---> -((X * Z))  [9, 6]

  Deleting rule: [15] ((X * (X1 * Z)) + (X * (X1 * 0))) ---> (X * (X1 * Z))  [9, 7]

  Deleting rule: [16] (((X * Z) * Z1) + ((X * 0) * Z1)) ---> ((X * Z) * Z1)  [9, 8]

  The right hand side is reducible:
```

```
    [19] (-((X * Z)) + -((X * -(Z)))) ---> -((X * 0))  [11, 6]

  Deleting rule: [21] (-((X * 0)) + (X * Z)) ---> (X * Z)  [9, 2]

Adding Rule:  [23] (-((0 * Z)) + (Z1 * Z)) ---> (Z1 * Z)  [10, 2]

Adding Rule:  [24] (0 * Z) ---> 0  [10, 2]

  Deleting rule: [10] ((Z * Z1) + (0 * Z1)) ---> (Z * Z1)  [1, 8]

  The right hand side is reducible:
    [12] ((Z * Z1) + (-(Z) * Z1)) ---> (0 * Z1)  [2, 8]

  Deleting rule: [14] (-((Z * Z1)) + -((0 * Z1))) ---> -((Z * Z1))  [10, 6]

  Deleting rule: [17] ((X * (Z * Z1)) + (X * (0 * Z1))) ---> (X * (Z * Z1))  [10, 7]

  Deleting rule: [18] (((Z * Z1) * Z2) + ((0 * Z1) * Z2)) ---> ((Z * Z1) * Z2)  [10, 8]

  The right hand side is reducible:
    [20] (-((Z * Z1)) + -((-(Z) * Z1))) ---> -((0 * Z1))  [12, 6]

  Deleting rule: [23] (-((0 * Z)) + (Z1 * Z)) ---> (Z1 * Z)  [10, 2]

Adding Rule:  [25] ((X * (X1 * Z)) + (X * (X1 * -(Z)))) ---> 0  [11, 7]

Adding Rule:  [26] (((X * Z) * Z1) + ((X * -(Z)) * Z1)) ---> 0  [11, 8]

Adding Rule:  [27] ((X * (Z * Z1)) + (X * (-(Z) * Z1))) ---> 0  [12, 7]

Adding Rule:  [28] (((Z * Z1) * Z2) + ((-(Z) * Z1) * Z2)) ---> 0  [12, 8]

Adding Rule:  [29] (X * -(Z)) ---> -((X * Z))  [11, 2]

  Deleting rule: [11] ((X * Z) + (X * -(Z))) ---> 0  [2, 7]

  Deleting rule: [19] (-((X * Z)) + -((X * -(Z)))) ---> 0  [11, 6]

  Deleting rule: [25] ((X * (X1 * Z)) + (X * (X1 * -(Z)))) ---> 0  [11, 7]

  Deleting rule: [26] (((X * Z) * Z1) + ((X * -(Z)) * Z1)) ---> 0  [11, 8]

Adding Rule:  [30] (-(Z) * Z1) ---> -((Z * Z1))  [12, 2]

  Deleting rule: [12] ((Z * Z1) + (-(Z) * Z1)) ---> 0  [2, 8]

  Deleting rule: [20] (-((Z * Z1)) + -((-(Z) * Z1))) ---> 0  [12, 6]

  Deleting rule: [27] ((X * (Z * Z1)) + (X * (-(Z) * Z1))) ---> 0  [12, 7]

  Deleting rule: [28] (((Z * Z1) * Z2) + ((-(Z) * Z1) * Z2)) ---> 0  [12, 8]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]
  [7] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 5]
  [8] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 6]
 [22] (X * 0) ---> 0  [9, 2]
 [24] (0 * Z) ---> 0  [10, 2]
 [29] (X * -(Z)) ---> -((X * Z))  [11, 2]
 [30] (-(Z) * Z1) ---> -((Z * Z1))  [12, 2]
```

```
Time used                          =  18.89 sec
Number of rules generated          = 30
Number of rules retained           = 11
Number of critical pairs           = 186
Number of unblocked critical pairs  = 49
Time spent in normalization        =  10.37 sec  (54.0 percent of time)
Time spent in unification          =   2.23 sec   (11.0 percent of time)
Time spent in ordering             =   0.38 sec   (2.0 percent of time)
Time spent in simplifying the rules =   2.48 sec   (13.0 percent of time)
Time spent in blocking             =   1.66 sec   (8.0 percent of time)
Total time (including 'undo' action) = 1651.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/8c.eq
New constant set is: { 1 }
Equations read in are:
 1. (1 * X) == X  [USER, 7]
 2. (X * 1) == X  [USER, 8]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 7 -----

Adding Rule:  [31] (1 * X) ---> X  [USER, 7]

Adding Rule:  [32] (X * 1) ---> X  [USER, 8]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]
  [7] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 5]
  [8] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 6]
 [22] (X * 0) ---> 0  [9, 2]
 [24] (0 * Z) ---> 0  [10, 2]
 [29] (X * -(Z)) ---> -((X * Z))  [11, 2]
 [30] (-(Z) * Z1) ---> -((Z * Z1))  [12, 2]
 [31] (1 * X) ---> X  [USER, 7]
 [32] (X * 1) ---> X  [USER, 8]

Time used                          =  19.04 sec
Number of rules generated          = 32
Number of rules retained           = 13
Number of critical pairs           = 195
Number of unblocked critical pairs  = 49
Time spent in normalization        =  10.41 sec  (54.0 percent of time)
Time spent in unification          =   2.23 sec   (11.0 percent of time)
Time spent in ordering             =   0.38 sec   (1.0 percent of time)
Time spent in simplifying the rules =   2.52 sec   (13.0 percent of time)
Time spent in blocking             =   1.67 sec   (8.0 percent of time)
Total time (including 'undo' action) = 19.0 sec
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/8d.eq
Equations read in are:
 1. ((X * Y) * Z) == (X * (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 8 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 9]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 9 -----

Processing all postponed equations ...

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 9]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 10 -----

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 9]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Operator, *, given status: rl

----- Step 11 -----

Adding Rule:  [33] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 9]

Your system is canonical.
```

```
  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]
  [7] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 5]
  [8] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 6]
 [22] (X * 0) ---> 0  [9, 2]
 [24] (0 * Z) ---> 0  [10, 2]
 [29] (X * -(Z)) ---> -((X * Z))  [11, 2]
 [30] (-(Z) * Z1) ---> -((Z * Z1))  [12, 2]
 [31] (1 * X) ---> X  [USER, 7]
 [32] (X * 1) ---> X  [USER, 8]
 [33] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 9]

Time used                            =  19.61 sec
Number of rules generated            = 33
Number of rules retained             = 14
Number of critical pairs             = 209
Number of unblocked critical pairs   = 49
Time spent in normalization          =  10.64 sec  (54.0 percent of time)
Time spent in unification            =   2.24 sec   (11.0 percent of time)
Time spent in ordering               =   0.60 sec   (3.0 percent of time)
Time spent in simplifying the rules  =   2.54 sec   (12.0 percent of time)
Time spent in blocking               =   1.67 sec   (8.0 percent of time)
Total time (including 'undo' action) = 64.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/8e.eq
Equations read in are:
 1. (X * Y) == (Y * X)  [USER, 10]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 12 -----

----- Step 13 -----

'*' has the associative and commutative property now:
    (X * Y) == (Y * X)  [USER, 10]
'*' has been given the status RL. Now, the status is cancelled.

  The LHS of Rule contains '*', which is AC now.
  [7] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 5]

  The LHS of Rule contains '*', which is AC now.
  [8] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 6]

  The LHS of Rule contains '*', which is AC now.
 [22] (X * 0) ---> 0  [9, 2]

  The LHS of Rule contains '*', which is AC now.
 [24] (0 * Z) ---> 0  [10, 2]

  The LHS of Rule contains '*', which is AC now.
 [29] (X * -(Z)) ---> -((X * Z))  [11, 2]

  The LHS of Rule contains '*', which is AC now.
```

```
 [30] (-(Z) * Z1) ---> -((Z * Z1))  [12, 2]

  The LHS of Rule contains ´*´, which is AC now.
 [31] (1 * X) ---> X  [USER, 7]

  The LHS of Rule contains ´*´, which is AC now.
 [32] (X * 1) ---> X  [USER, 8]

  The LHS of Rule contains ´*´, which is AC now.
 [33] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 9]

Adding Rule:  [34] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 7]

Adding Rule:  [35] (X * 0) ---> 0  [DELETED, 22]

Adding Rule:  [36] (X * -(Z)) ---> -((X * Z))  [DELETED, 29]

Adding Rule:  [37] (X * 1) ---> X  [DELETED, 31]

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] -(0) ---> 0  [2, 1]
  [4] -(-(Z)) ---> Z  [2, 2]
  [6] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 5]
 [34] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 7]
 [35] (X * 0) ---> 0  [DELETED, 22]
 [36] (X * -(Z)) ---> -((X * Z))  [DELETED, 29]
 [37] (X * 1) ---> X  [DELETED, 31]

Time used                           =  31.74 sec
Number of rules generated           = 37
Number of rules retained            = 9
Number of critical pairs            = 308
Number of unblocked critical pairs  = 96
Time spent in normalization         =  17.26 sec  (54.0 percent of time)
Time spent in unification           =   5.64 sec   (17.0 percent of time)
Time spent in ordering              =   0.64 sec   (2.0 percent of time)
Time spent in simplifying the rules =   2.64 sec   (8.0 percent of time)
Time spent in blocking              =   2.05 sec   (6.0 percent of time)
Total time (including ´undo´ action) = 1219.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

9 Anticommutative Rings and Lie Rings

9.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 9.1 *A Lie ring is a ring, i.e. fullfilling the axioms:*

$$
\begin{aligned}
x + y &= y + x && (1)\\
(x + y) + z &= x + (y + z) && (2)\\
x + 0 &= x && (3)\\
x + (-x) &= 0 && (4)\\
x * (y + z) &= (x * y) + (x * z) && (5)\\
(x + y) * z &= (x * z) + (y * z) && (6)
\end{aligned}
$$

together with a new binary operation [_,_], i.e. [_,_]:$S \times S \to S$ fullfilling the additional axioms:

$$
\begin{aligned}
[x, x] &= 0 && (7)\\
[[x, y], z] + [[y, z], x] + [[z, x], y] &= 0 && (8)
\end{aligned}
$$

9.2 Example

A model of a Lie ring is the three dimensional vector space over the real numbers with vector addition. The $*$ and the [_, _] operation coincide being the cross product.

9.3 A canonical System

In [Hul80] there is no canonical system for Lie rings. Although RRL terminates with a positive report obviously the added equations totally disappeared, since in the canonical system the [_,_] operation is missing. Thus the result is not correct.

9.4 Trace

The operation $+$ is represented by ADD, $-$ by INV, $*$ by MUL and [_, _] by LAW; 0 remains the same.

```
RRL-> read test/9a.eq
New constant set is: { 0 }
Equations read in are:
 1. ADD(X, Y) == ADD(Y, X)  [USER, 1]
 2. ADD(ADD(X, Y), Z) == ADD(X, ADD(Y, Z))  [USER, 2]
 3. ADD(X, 0) == X  [USER, 3]
 4. ADD(X, INV(X)) == 0  [USER, 4]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
```

```
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

'ADD' has the commutative property now:
    ADD(X, Y) == ADD(Y, X)  [USER, 1]

----- Step 3 -----

'ADD' has the associative and commutative property now:
    ADD(Z, ADD(X, Y)) == ADD(X, ADD(Y, Z))  [USER, 2]

Adding Rule:   [1] ADD(X, 0) ---> X  [USER, 3]

Adding Rule:   [2] ADD(X, INV(X)) ---> 0  [USER, 4]

Adding Rule:   [3] INV(0) ---> 0  [2, 1]

Adding Rule:   [4] INV(INV(Z)) ---> Z  [2, 2]

Adding Rule:   [5] ADD(Z, INV(ADD(Z, Z1))) ---> INV(Z1)  [2, 2]

Trying to orient equation:
  ADD(INV(Z), INV(Z1)) == INV(ADD(Z, Z1))  [2, 5]

  To prove:  ADD(INV(Z), INV(Z1))  >  INV(ADD(Z, Z1))
        or:  ADD(INV(Z), INV(Z1))  <  INV(ADD(Z, Z1))
  Here are some precedence suggestions:
       1.  ADD > INV      2.  INV > ADD
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, ADD = INV, is added.

----- Step 4 -----

Adding Rule:   [6] INV(ADD(Z, Z1)) ---> ADD(INV(Z), INV(Z1))  [2, 5]

  Deleting rule:  [5] ADD(Z, INV(ADD(Z, Z1))) ---> INV(Z1)  [2, 2]

Your system is canonical.

  [1] ADD(X, 0) ---> X  [USER, 3]
  [2] ADD(X, INV(X)) ---> 0  [USER, 4]
  [3] INV(0) ---> 0  [2, 1]
  [4] INV(INV(Z)) ---> Z  [2, 2]
  [6] INV(ADD(Z, Z1)) ---> ADD(INV(Z), INV(Z1))  [2, 5]

Time used                          =   2.57 sec
Number of rules generated          = 6
Number of rules retained           = 5
Number of critical pairs           = 51
Number of unblocked critical pairs  = 29
Time spent in normalization        =   0.96 sec  (37.0 percent of time)
Time spent in unification          =   0.79 sec   (30.0 percent of time)
Time spent in ordering             =   0.11 sec   (4.0 percent of time)
Time spent in simplifying the rules =   0.09 sec   (3.0 percent of time)
Time spent in blocking             =   0.19 sec   (7.0 percent of time)
Total time (including 'undo' action) = 259.0 sec
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/9b.eq
Equations read in are:
 1. MUL(X, ADD(Y, Z)) == ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]
 2. MUL(ADD(X, Y), Z) == ADD(MUL(X, Z), MUL(Y, Z))  [USER, 6]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 5 -----

Trying to orient equation:
  MUL(X, ADD(Y, Z)) == ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]

  To prove:  MUL(X, ADD(Y, Z))  >  ADD(MUL(X, Y), MUL(X, Z))
        or:  MUL(X, ADD(Y, Z))  <  ADD(MUL(X, Y), MUL(X, Z))
  Here are some precedence suggestions:
       1.  MUL > ADD      2.  ADD > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, MUL > ADD, is added.

----- Step 6 -----

Adding Rule:   [7] MUL(X, ADD(Y, Z)) ---> ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]

Adding Rule:   [8] MUL(ADD(X, Y), Z) ---> ADD(MUL(X, Z), MUL(Y, Z))  [USER, 6]

Adding Rule:   [9] ADD(MUL(X, Z), MUL(X, 0)) ---> MUL(X, Z)  [1, 7]

Adding Rule:  [10] ADD(MUL(Z, Z1), MUL(0, Z1)) ---> MUL(Z, Z1)  [1, 8]

Adding Rule:  [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> MUL(X, 0)  [2, 7]

Adding Rule:  [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> MUL(0, Z1)  [2, 8]

Adding Rule:  [13] ADD(INV(MUL(X, Z)), INV(MUL(X, 0))) ---> INV(MUL(X, Z))  [9, 6]

Adding Rule:  [14] ADD(INV(MUL(Z, Z1)), INV(MUL(0, Z1))) ---> INV(MUL(Z, Z1))  [10, 6]

Adding Rule:  [15] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, 0))) ---> MUL(X, MUL(X1, Z))  [9, 7]

Adding Rule:  [16] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, 0), Z1)) ---> MUL(MUL(X, Z), Z1)  [9, 8]

Adding Rule:  [17] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(0, Z1))) ---> MUL(X, MUL(Z, Z1))  [10, 7]

Adding Rule:  [18] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(0, Z1), Z2)) ---> MUL(MUL(Z, Z1), Z2)  [10, 8]

Adding Rule:  [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> INV(MUL(X, 0))  [11, 6]

Adding Rule:  [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> INV(MUL(0, Z1))  [12, 6]

Adding Rule:  [21] ADD(INV(MUL(X, 0)), MUL(X, Z)) ---> MUL(X, Z)  [9, 2]

Adding Rule:  [22] MUL(X, 0) ---> 0  [9, 2]

  Deleting rule:  [9] ADD(MUL(X, Z), MUL(X, 0)) ---> MUL(X, Z)  [1, 7]

  The right hand side is reducible:
```

```
    [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> MUL(X, 0)  [2, 7]

  Deleting rule: [13] ADD(INV(MUL(X, Z)), INV(MUL(X, 0))) ---> INV(MUL(X, Z))  [9, 6]

  Deleting rule: [15] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, 0))) ---> MUL(X, MUL(X1, Z))  [9, 7]

  Deleting rule: [16] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, 0), Z1)) ---> MUL(MUL(X, Z), Z1)  [9, 8]

  The right hand side is reducible:
    [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> INV(MUL(X, 0))  [11, 6]

  Deleting rule: [21] ADD(INV(MUL(X, 0)), MUL(X, Z)) ---> MUL(X, Z)  [9, 2]

Adding Rule:  [23] ADD(INV(MUL(0, Z)), MUL(Z1, Z)) ---> MUL(Z1, Z)  [10, 2]

Adding Rule:  [24] MUL(0, Z) ---> 0  [10, 2]

  Deleting rule: [10] ADD(MUL(Z, Z1), MUL(0, Z1)) ---> MUL(Z, Z1)  [1, 8]

  The right hand side is reducible:
    [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> MUL(0, Z1)  [2, 8]

  Deleting rule: [14] ADD(INV(MUL(Z, Z1)), INV(MUL(0, Z1))) ---> INV(MUL(Z, Z1))  [10, 6]

  Deleting rule: [17] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(0, Z1))) ---> MUL(X, MUL(Z, Z1))  [10, 7]

  Deleting rule: [18] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(0, Z1), Z2)) ---> MUL(MUL(Z, Z1), Z2)  [10, 8]

  The right hand side is reducible:
    [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> INV(MUL(0, Z1))  [12, 6]

  Deleting rule: [23] ADD(INV(MUL(0, Z)), MUL(Z1, Z)) ---> MUL(Z1, Z)  [10, 2]

Adding Rule:  [25] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, INV(Z)))) ---> 0  [11, 7]

Adding Rule:  [26] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, INV(Z)), Z1)) ---> 0  [11, 8]

Adding Rule:  [27] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(INV(Z), Z1))) ---> 0  [12, 7]

Adding Rule:  [28] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(INV(Z), Z1), Z2)) ---> 0  [12, 8]

Adding Rule:  [29] MUL(X, INV(Z)) ---> INV(MUL(X, Z))  [11, 2]

  Deleting rule: [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> 0  [2, 7]

  Deleting rule: [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> 0  [11, 6]

  Deleting rule: [25] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, INV(Z)))) ---> 0  [11, 7]

  Deleting rule: [26] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, INV(Z)), Z1)) ---> 0  [11, 8]

Adding Rule:  [30] MUL(INV(Z), Z1) ---> INV(MUL(Z, Z1))  [12, 2]

  Deleting rule: [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> 0  [2, 8]

  Deleting rule: [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> 0  [12, 6]

  Deleting rule: [27] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(INV(Z), Z1))) ---> 0  [12, 7]

  Deleting rule: [28] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(INV(Z), Z1), Z2)) ---> 0  [12, 8]

Your system is canonical.

  [1] ADD(X, 0) ---> X  [USER, 3]
  [2] ADD(X, INV(X)) ---> 0  [USER, 4]
```

```
  [3] INV(0) ---> 0  [2, 1]
  [4] INV(INV(Z)) ---> Z  [2, 2]
  [6] INV(ADD(Z, Z1)) ---> ADD(INV(Z), INV(Z1))  [2, 5]
  [7] MUL(X, ADD(Y, Z)) ---> ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]
  [8] MUL(ADD(X, Y), Z) ---> ADD(MUL(X, Z), MUL(Y, Z))  [USER, 6]
 [22] MUL(X, 0) ---> 0  [9, 2]
 [24] MUL(0, Z) ---> 0  [10, 2]
 [29] MUL(X, INV(Z)) ---> INV(MUL(X, Z))  [11, 2]
 [30] MUL(INV(Z), Z1) ---> INV(MUL(Z, Z1))  [12, 2]

Time used                           =  17.87 sec
Number of rules generated           = 30
Number of rules retained            = 11
Number of critical pairs            = 186
Number of unblocked critical pairs  = 49
Time spent in normalization         =  10.54 sec  (58.0 percent of time)
Time spent in unification           =   2.25 sec   (12.0 percent of time)
Time spent in ordering              =   1.40 sec   (7.0 percent of time)
Time spent in simplifying the rules =   1.33 sec   (7.0 percent of time)
Time spent in blocking              =   0.65 sec   (3.0 percent of time)
Total time (including 'undo' action) = 1534.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/9c.eq
Equations read in are:
 1. LAW(Z, Z) == 0  [USER, 7]
 2. ADD(ADD(LAW(LAW(X, Y), Z), LAW(LAW(Y, Z), X)), LAW(LAW(Z, X), Y)) == 0  [USER, 8]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 7 -----

Adding Rule:  [31] LAW(Z, Z) ---> 0  [USER, 7]

Adding Rule:  [32] ADD(LAW(LAW(X, Y), Z), LAW(LAW(Y, Z), X), LAW(LAW(Z, X), Y)) ---> 0  [USER, 8]

Adding Rule:  [33] ADD(LAW(LAW(Y, LAW(X, Y)), X), LAW(LAW(LAW(X, Y), X), Y)) ---> 0  [31, 32]

Adding Rule:  [34] ADD(LAW(0, Z), LAW(LAW(Y, Z), Y), LAW(LAW(Z, Y), Y)) ---> 0  [31, 32]

Adding Rule:  [35] ADD(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y), Y)) ---> 0  [31, 33]

Adding Rule:  [36] ADD(INV(LAW(LAW(Y, 0), Y)), INV(LAW(LAW(0, Y), Y))) ---> 0  [35, 6]

Adding Rule:  [37] ADD(MUL(X, LAW(LAW(Y, 0), Y)), MUL(X, LAW(LAW(0, Y), Y))) ---> 0  [35, 7]

Adding Rule:  [38] ADD(MUL(LAW(LAW(Y, 0), Y), Z), MUL(LAW(LAW(0, Y), Y), Z)) ---> 0  [35, 8]

Adding Rule:  [39] INV(LAW(LAW(0, Y), Y)) ---> LAW(LAW(Y, 0), Y)  [35, 2]

  Deleting rule: [36] ADD(INV(LAW(LAW(Y, 0), Y)), INV(LAW(LAW(0, Y), Y))) ---> 0  [35, 6]

Adding Rule:  [40] INV(LAW(LAW(Y, 0), Y)) ---> LAW(LAW(0, Y), Y)  [35, 2]

Adding Rule:  [41] INV(MUL(X, LAW(LAW(0, Y), Y))) ---> MUL(X, LAW(LAW(Y, 0), Y))  [39, 29]

Adding Rule:  [42] INV(MUL(LAW(LAW(0, Y), Y), Z)) ---> MUL(LAW(LAW(Y, 0), Y), Z)  [39, 30]
```

```
Adding Rule:  [43] INV(MUL(X, LAW(LAW(Y, 0), Y))) ---> MUL(X, LAW(LAW(0, Y), Y))  [40, 29]

Adding Rule:  [44] INV(MUL(LAW(LAW(Y, 0), Y), Z)) ---> MUL(LAW(LAW(0, Y), Y), Z)  [40, 30]

Adding Rule:  [45] INV(MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))) --->
MUL(X, MUL(X1, LAW(LAW(Y, 0), Y)))  [41, 29]

Adding Rule:  [46] INV(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)) --->
MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z)  [41, 30]

Adding Rule:  [47] INV(MUL(X, MUL(LAW(LAW(0, Y), Y), Z))) --->
MUL(X, MUL(LAW(LAW(Y, 0), Y), Z))  [42, 29]

Adding Rule:  [48] INV(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)) --->
MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)  [42, 30]

Adding Rule:  [49] INV(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y)))) --->
MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))  [43, 29]

Adding Rule:  [50] INV(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z)) --->
MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)  [43, 30]

Adding Rule:  [51] INV(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z))) --->
MUL(X, MUL(LAW(LAW(0, Y), Y), Z))  [44, 29]

Adding Rule:  [52] INV(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)) --->
MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)  [44, 30]

Adding Rule:  [53] ADD(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))) --->
0  [45, 2]

Adding Rule:  [54] INV(MUL(X, MUL(X1, MUL(X2, LAW(LAW(0, Y), Y))))) --->
MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y))))  [45, 29]

Adding Rule:  [55] INV(MUL(MUL(X, MUL(X1, LAW(LAW(0, Y), Y))), Z)) --->
MUL(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), Z)  [45, 30]

Adding Rule:  [56] ADD(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)) --->
0  [46, 2]

Adding Rule:  [57] INV(MUL(X, MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z))) --->
MUL(X, MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z))  [46, 29]

Adding Rule:  [58] INV(MUL(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z), Z1)) --->
MUL(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), Z1)  [46, 30]

Adding Rule:  [59] ADD(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), MUL(X, MUL(LAW(LAW(0, Y), Y), Z))) --->
0  [47, 2]

Adding Rule:  [60] INV(MUL(X, MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)))) --->
MUL(X, MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)))  [47, 29]

Adding Rule:  [61] INV(MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1)) --->
MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1)  [47, 30]

Adding Rule:  [62] ADD(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)) --->
0  [48, 2]

Adding Rule:  [63] INV(MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1))) --->
MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1))  [48, 29]

Adding Rule:  [64] INV(MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2)) --->
MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2)  [48, 30]

Adding Rule:  [65] INV(MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y))))) --->
```

```
MUL(X, MUL(X1, MUL(X2, LAW(LAW(0, Y), Y))))  [49, 29]

Adding Rule:  [66] INV(MUL(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), Z)) --->
MUL(MUL(X, MUL(X1, LAW(LAW(0, Y), Y))), Z)  [49, 30]

Adding Rule:  [67] INV(MUL(X, MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z))) --->
MUL(X, MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z))  [50, 29]

Adding Rule:  [68] INV(MUL(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), Z1)) --->
MUL(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z), Z1)  [50, 30]

Adding Rule:  [69] INV(MUL(X, MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)))) --->
MUL(X, MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)))  [51, 29]

Adding Rule:  [70] INV(MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1)) --->
MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1)  [51, 30]

Adding Rule:  [71] INV(MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1))) --->
MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1))  [52, 29]

Adding Rule:  [72] INV(MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2)) --->
MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2)  [52, 30]

Trying to orient equation:
  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  [42, 41]

  To prove:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 8 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [43, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 9 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [44, 41]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.
```

```
----- Step 10 -----

Trying to orient equation:
  MUL(LAW(LAW(O, Y), Y), LAW(LAW(Y1, O), Y1)) == MUL(LAW(LAW(Y, O), Y), LAW(LAW(O, Y1), Y1))  [44, 43]

  To prove:  MUL(LAW(LAW(O, Y), Y), LAW(LAW(Y1, O), Y1))  >  MUL(LAW(LAW(Y, O), Y), LAW(LAW(O, Y1), Y1))
        or:  MUL(LAW(LAW(O, Y), Y), LAW(LAW(Y1, O), Y1))  <  MUL(LAW(LAW(Y, O), Y), LAW(LAW(O, Y1), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 11 -----

Adding Rule:  [73] ADD(MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, O), Y)))),
MUL(X, MUL(X1, MUL(X2, LAW(LAW(O, Y), Y))))) ---> O  [54, 2]

Adding Rule:  [74] INV(MUL(X, MUL(X1, MUL(X2, MUL(X3, LAW(LAW(O, Y), Y)))))) --->
MUL(X, MUL(X1, MUL(X2, MUL(X3, LAW(LAW(Y, O), Y)))))  [54, 29]

Adding Rule:  [75] INV(MUL(MUL(X, MUL(X1, MUL(X2, LAW(LAW(O, Y), Y)))), Z)) --->
MUL(MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, O), Y)))), Z)  [54, 30]

Adding Rule:  [76] ADD(MUL(MUL(X, MUL(X1, LAW(LAW(Y, O), Y))), Z),
MUL(MUL(X, MUL(X1, LAW(LAW(O, Y), Y))), Z)) ---> O  [55, 2]

Adding Rule:  [77] INV(MUL(X, MUL(MUL(X1, MUL(X2, LAW(LAW(O, Y), Y))), Z))) --->
MUL(X, MUL(MUL(X1, MUL(X2, LAW(LAW(Y, O), Y))), Z))  [55, 29]

Adding Rule:  [78] INV(MUL(MUL(MUL(X, MUL(X1, LAW(LAW(O, Y), Y))), Z), Z2)) --->
MUL(MUL(MUL(X, MUL(X1, LAW(LAW(Y, O), Y))), Z), Z2)  [55, 30]

Adding Rule:  [79] ADD(MUL(X, MUL(MUL(X1, LAW(LAW(Y, O), Y)), Z)),
MUL(X, MUL(MUL(X1, LAW(LAW(O, Y), Y)), Z))) ---> O  [57, 2]

Adding Rule:  [80] INV(MUL(X, MUL(X1, MUL(MUL(X2, LAW(LAW(O, Y), Y)), Z)))) --->
MUL(X, MUL(X1, MUL(MUL(X2, LAW(LAW(Y, O), Y)), Z)))  [57, 29]

Adding Rule:  [81] INV(MUL(MUL(X, MUL(MUL(X1, LAW(LAW(O, Y), Y)), Z)), Z2)) --->
MUL(MUL(X, MUL(MUL(X1, LAW(LAW(Y, O), Y)), Z)), Z2)  [57, 30]

Adding Rule:  [82] ADD(MUL(MUL(MUL(X, LAW(LAW(Y, O), Y)), Z), Z1),
MUL(MUL(MUL(X, LAW(LAW(O, Y), Y)), Z), Z1)) ---> O  [58, 2]

Adding Rule:  [83] INV(MUL(X, MUL(MUL(MUL(X1, LAW(LAW(O, Y), Y)), Z), Z2))) --->
MUL(X, MUL(MUL(MUL(X1, LAW(LAW(Y, O), Y)), Z), Z2))  [58, 29]

Adding Rule:  [84] INV(MUL(MUL(MUL(MUL(X, LAW(LAW(O, Y), Y)), Z), Z1), Z2)) --->
MUL(MUL(MUL(MUL(X, LAW(LAW(Y, O), Y)), Z), Z1), Z2)  [58, 30]

Adding Rule:  [85] ADD(MUL(X, MUL(X1, MUL(LAW(LAW(Y, O), Y), Z))),
MUL(X, MUL(X1, MUL(LAW(LAW(O, Y), Y), Z)))) ---> O  [60, 2]

Adding Rule:  [86] INV(MUL(X, MUL(X1, MUL(X2, MUL(LAW(LAW(O, Y), Y), Z))))) --->
MUL(X, MUL(X1, MUL(X2, MUL(LAW(LAW(Y, O), Y), Z))))  [60, 29]

Adding Rule:  [87] INV(MUL(MUL(X, MUL(X1, MUL(LAW(LAW(O, Y), Y), Z))), Z2)) --->
MUL(MUL(X, MUL(X1, MUL(LAW(LAW(Y, O), Y), Z))), Z2)  [60, 30]

Adding Rule:  [88] ADD(MUL(MUL(X, MUL(LAW(LAW(Y, O), Y), Z)), Z1),
MUL(MUL(X, MUL(LAW(LAW(O, Y), Y), Z)), Z1)) ---> O  [61, 2]
```

```
Adding Rule:  [89] INV(MUL(X, MUL(MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)), Z2))) --->
MUL(X, MUL(MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)), Z2))  [61, 29]

Adding Rule:  [90] INV(MUL(MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1), Z2)) --->
MUL(MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1), Z2)  [61, 30]

Adding Rule:  [91] ADD(MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)),
MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1))) ---> 0  [63, 2]

Adding Rule:  [92] INV(MUL(X, MUL(X1, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z2)))) --->
MUL(X, MUL(X1, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z2)))  [63, 29]

Adding Rule:  [93] INV(MUL(MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)), Z2)) --->
MUL(MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)), Z2)  [63, 30]

Adding Rule:  [94] ADD(MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2),
MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2)) ---> 0  [64, 2]

Adding Rule:  [95] INV(MUL(X, MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2))) --->
MUL(X, MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2))  [64, 29]

Adding Rule:  [96] INV(MUL(MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2), Z3)) --->
MUL(MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2), Z3)  [64, 30]

Adding Rule:  [97] INV(MUL(X, MUL(X1, MUL(X2, MUL(X3, LAW(LAW(Y, 0), Y)))))) --->
MUL(X, MUL(X1, MUL(X2, MUL(X3, LAW(LAW(0, Y), Y)))))  [65, 29]

Adding Rule:  [98] INV(MUL(MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y)))), Z)) --->
MUL(MUL(X, MUL(X1, MUL(X2, LAW(LAW(0, Y), Y)))), Z)  [65, 30]

Adding Rule:  [99] INV(MUL(X, MUL(MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y))), Z))) --->
MUL(X, MUL(MUL(X1, MUL(X2, LAW(LAW(0, Y), Y))), Z))  [66, 29]

Adding Rule:  [100] INV(MUL(MUL(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), Z), Z2)) --->
MUL(MUL(MUL(X, MUL(X1, LAW(LAW(0, Y), Y))), Z), Z2)  [66, 30]

Adding Rule:  [101] INV(MUL(X, MUL(X1, MUL(MUL(X2, LAW(LAW(Y, 0), Y)), Z)))) --->
MUL(X, MUL(X1, MUL(MUL(X2, LAW(LAW(0, Y), Y)), Z)))  [67, 29]

Adding Rule:  [102] INV(MUL(MUL(X, MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z)), Z2)) --->
MUL(MUL(X, MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z)), Z2)  [67, 30]

Adding Rule:  [103] INV(MUL(X, MUL(MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z), Z2))) --->
MUL(X, MUL(MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z), Z2))  [68, 29]

Adding Rule:  [104] INV(MUL(MUL(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), Z1), Z2)) --->
MUL(MUL(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z), Z1), Z2)  [68, 30]

Adding Rule:  [105] INV(MUL(X, MUL(X1, MUL(X2, MUL(LAW(LAW(Y, 0), Y), Z))))) --->
MUL(X, MUL(X1, MUL(X2, MUL(LAW(LAW(0, Y), Y), Z))))  [69, 29]

Adding Rule:  [106] INV(MUL(MUL(X, MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z))), Z2)) --->
MUL(MUL(X, MUL(X1, MUL(LAW(LAW(0, Y), Y), Z))), Z2)  [69, 30]

Adding Rule:  [107] INV(MUL(X, MUL(MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)), Z2))) --->
MUL(X, MUL(MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)), Z2))  [70, 29]

Adding Rule:  [108] INV(MUL(MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1), Z2)) --->
MUL(MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1), Z2)  [70, 30]

Adding Rule:  [109] INV(MUL(X, MUL(X1, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z2)))) --->
MUL(X, MUL(X1, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z2)))  [71, 29]

Adding Rule:  [110] INV(MUL(MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)), Z2)) --->
```

```
MUL(MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)), Z2)  [71, 30]

Adding Rule:  [111] INV(MUL(X, MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2))) --->
MUL(X, MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2))  [72, 29]

Adding Rule:  [112] INV(MUL(MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2), Z3)) --->
MUL(MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2), Z3)  [72, 30]

Adding Rule:  [113] ADD(LAW(0, Z), LAW(0, Z), LAW(0, Z)) ---> 0  [31, 34]

Adding Rule:  [114] ADD(INV(LAW(0, Z)), INV(LAW(0, Z)), INV(LAW(0, Z))) ---> 0  [113, 6]

Adding Rule:  [115] ADD(MUL(X, LAW(0, Z)), MUL(X, LAW(0, Z)), MUL(X, LAW(0, Z))) ---> 0  [113, 7]

Adding Rule:  [116] ADD(MUL(LAW(0, Z), Z1), MUL(LAW(0, Z), Z1), MUL(LAW(0, Z), Z1)) ---> 0  [113, 8]

Adding Rule:  [117] ADD(LAW(0, Z), LAW(0, Z)) ---> INV(LAW(0, Z))  [113, 2]

  Deleting rule: [113] ADD(LAW(0, Z), LAW(0, Z), LAW(0, Z)) ---> 0  [31, 34]

Adding Rule:  [118] ADD(INV(LAW(0, Z)), INV(LAW(0, Z))) ---> LAW(0, Z)  [117, 6]

  Deleting rule: [114] ADD(INV(LAW(0, Z)), INV(LAW(0, Z)), INV(LAW(0, Z))) ---> 0  [113, 6]

Adding Rule:  [119] ADD(MUL(X, LAW(0, Z)), MUL(X, LAW(0, Z))) ---> INV(MUL(X, LAW(0, Z)))  [117, 7]

  Deleting rule: [115] ADD(MUL(X, LAW(0, Z)), MUL(X, LAW(0, Z)), MUL(X, LAW(0, Z))) ---> 0  [113, 7]

Adding Rule:  [120] ADD(MUL(LAW(0, Z), Z1), MUL(LAW(0, Z), Z1)) ---> INV(MUL(LAW(0, Z), Z1))  [117, 8]

  Deleting rule: [116] ADD(MUL(LAW(0, Z), Z1), MUL(LAW(0, Z), Z1), MUL(LAW(0, Z), Z1)) ---> 0  [113, 8]

Adding Rule:  [121] ADD(INV(MUL(X, LAW(0, Z))), INV(MUL(X, LAW(0, Z)))) ---> MUL(X, LAW(0, Z))  [118, 7]

Adding Rule:  [122] ADD(INV(MUL(LAW(0, Z), Z1)), INV(MUL(LAW(0, Z), Z1))) ---> MUL(LAW(0, Z), Z1)  [118, 8]

Adding Rule:  [123] ADD(MUL(X, MUL(X1, LAW(0, Z))), MUL(X, MUL(X1, LAW(0, Z)))) --->
INV(MUL(X, MUL(X1, LAW(0, Z))))  [119, 7]

Adding Rule:  [124] ADD(MUL(MUL(X, LAW(0, Z)), Z1), MUL(MUL(X, LAW(0, Z)), Z1)) --->
INV(MUL(MUL(X, LAW(0, Z)), Z1))  [119, 8]

Adding Rule:  [125] ADD(MUL(X, MUL(LAW(0, Z), Z1)), MUL(X, MUL(LAW(0, Z), Z1))) --->
INV(MUL(X, MUL(LAW(0, Z), Z1)))  [120, 7]

Adding Rule:  [126] ADD(MUL(MUL(LAW(0, Z), Z1), Z2), MUL(MUL(LAW(0, Z), Z1), Z2)) --->
INV(MUL(MUL(LAW(0, Z), Z1), Z2))  [120, 8]

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1))) ==
  MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))  [45, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  >
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
        or:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  <
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 12 -----
```

```
Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  [44, 43]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [44, 41]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [43, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

Trying to orient equation:
  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  [42, 41]

  To prove:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1))) ==
  MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))  [45, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  >
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
        or:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  <
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
```

```
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto

WARNING: 4 equations have been postponed. I undo it.
----- Return to Step 11 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1))) ==
  MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))  [45, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  >
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
        or:  MUL(LAW(LAW(0, Y), Y), MUL(X, LAW(LAW(Y1, 0), Y1)))  <
             MUL(LAW(LAW(Y, 0), Y), MUL(X, LAW(LAW(0, Y1), Y1)))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 10 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  [44, 43]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 9 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [44, 41]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 8 -----

Trying to orient equation:
  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))  [43, 42]

  To prove:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(0, Y), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 7 -----

Trying to orient equation:
  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  [42, 41]

  To prove:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
```

```
       or:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
      1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 6 -----

Adding Rule:  [31] LAW(Z, Z) ---> 0  [USER, 7]

Adding Rule:  [32] ADD(ADD(LAW(LAW(X, Y), Z), LAW(LAW(Y, Z), X)), LAW(LAW(Z, X), Y)) ---> 0  [USER, 8]

Adding Rule:  [33] ADD(LAW(LAW(Y, LAW(X, Y)), X), LAW(LAW(LAW(X, Y), X), Y)) ---> 0  [31, 32]

Adding Rule:  [34] ADD(LAW(0, Z), LAW(LAW(Y, Z), Y), LAW(LAW(Z, Y), Y)) ---> 0  [31, 32]

Adding Rule:  [35] ADD(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y), Y)) ---> 0  [31, 33]

Adding Rule:  [36] ADD(INV(LAW(LAW(Y, 0), Y)), INV(LAW(LAW(0, Y), Y))) ---> 0  [35, 6]

Adding Rule:  [37] ADD(MUL(X, LAW(LAW(Y, 0), Y)), MUL(X, LAW(LAW(0, Y), Y))) ---> 0  [35, 7]

Adding Rule:  [38] ADD(MUL(LAW(LAW(Y, 0), Y), Z), MUL(LAW(LAW(0, Y), Y), Z)) ---> 0  [35, 8]

Adding Rule:  [39] INV(LAW(LAW(0, Y), Y)) ---> LAW(LAW(Y, 0), Y)  [35, 2]

  Deleting rule: [36] ADD(INV(LAW(LAW(Y, 0), Y)), INV(LAW(LAW(0, Y), Y))) ---> 0  [35, 6]

Adding Rule:  [40] INV(LAW(LAW(Y, 0), Y)) ---> LAW(LAW(0, Y), Y)  [35, 2]

Adding Rule:  [41] INV(MUL(X, LAW(LAW(0, Y), Y))) ---> MUL(X, LAW(LAW(Y, 0), Y))  [39, 29]

Adding Rule:  [42] INV(MUL(LAW(LAW(0, Y), Y), Z)) ---> MUL(LAW(LAW(Y, 0), Y), Z)  [39, 30]

Adding Rule:  [43] INV(MUL(X, LAW(LAW(Y, 0), Y))) ---> MUL(X, LAW(LAW(0, Y), Y))  [40, 29]

Adding Rule:  [44] INV(MUL(LAW(LAW(Y, 0), Y), Z)) ---> MUL(LAW(LAW(0, Y), Y), Z)  [40, 30]

Adding Rule:  [45] INV(MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))) ---> MUL(X, MUL(X1, LAW(LAW(Y, 0), Y)))  [41, 29]

Adding Rule:  [46] INV(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)) ---> MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z)  [41, 30]

Adding Rule:  [47] INV(MUL(X, MUL(LAW(LAW(0, Y), Y), Z))) ---> MUL(X, MUL(LAW(LAW(Y, 0), Y), Z))  [42, 29]

Adding Rule:  [48] INV(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)) ---> MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)  [42, 30]

Adding Rule:  [49] INV(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y)))) ---> MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))  [43, 29]

Adding Rule:  [50] INV(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z)) ---> MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)  [43, 30]

Adding Rule:  [51] INV(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z))) ---> MUL(X, MUL(LAW(LAW(0, Y), Y), Z))  [44, 29]

Adding Rule:  [52] INV(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1)) ---> MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)  [44, 30]

Adding Rule:  [53] ADD(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))),
MUL(X, MUL(X1, LAW(LAW(0, Y), Y)))) ---> 0  [45, 2]

Adding Rule:  [54] INV(MUL(X, MUL(X1, MUL(X2, LAW(LAW(0, Y), Y))))) --->
MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y))))  [45, 29]

Adding Rule:  [55] INV(MUL(MUL(X, MUL(X1, LAW(LAW(0, Y), Y))), Z)) --->
MUL(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), Z)  [45, 30]

Adding Rule:  [56] ADD(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z),
```

```
MUL(MUL(X, LAW(LAW(0, Y), Y)), Z)) ---> 0  [46, 2]

Adding Rule:  [57] INV(MUL(X, MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z))) --->
MUL(X, MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z))  [46, 29]

Adding Rule:  [58] INV(MUL(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z), Z1)) --->
MUL(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), Z1)  [46, 30]

Adding Rule:  [59] ADD(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)),
MUL(X, MUL(LAW(LAW(0, Y), Y), Z))) ---> 0  [47, 2]

Adding Rule:  [60] INV(MUL(X, MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)))) --->
MUL(X, MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)))  [47, 29]

Adding Rule:  [61] INV(MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1)) --->
MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1)  [47, 30]

Adding Rule:  [62] ADD(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1),
MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1)) ---> 0  [48, 2]

Adding Rule:  [63] INV(MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1))) --->
MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1))  [48, 29]

Adding Rule:  [64] INV(MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2)) --->
MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2)  [48, 30]

Adding Rule:  [65] INV(MUL(X, MUL(X1, MUL(X2, LAW(LAW(Y, 0), Y))))) --->
MUL(X, MUL(X1, MUL(X2, LAW(LAW(0, Y), Y))))  [49, 29]

Adding Rule:  [66] INV(MUL(MUL(X, MUL(X1, LAW(LAW(Y, 0), Y))), Z)) --->
MUL(MUL(X, MUL(X1, LAW(LAW(0, Y), Y))), Z)  [49, 30]

Adding Rule:  [67] INV(MUL(X, MUL(MUL(X1, LAW(LAW(Y, 0), Y)), Z))) --->
MUL(X, MUL(MUL(X1, LAW(LAW(0, Y), Y)), Z))  [50, 29]

Adding Rule:  [68] INV(MUL(MUL(MUL(X, LAW(LAW(Y, 0), Y)), Z), Z1)) --->
MUL(MUL(MUL(X, LAW(LAW(0, Y), Y)), Z), Z1)  [50, 30]

Adding Rule:  [69] INV(MUL(X, MUL(X1, MUL(LAW(LAW(Y, 0), Y), Z)))) --->
MUL(X, MUL(X1, MUL(LAW(LAW(0, Y), Y), Z)))  [51, 29]

Adding Rule:  [70] INV(MUL(MUL(X, MUL(LAW(LAW(Y, 0), Y), Z)), Z1)) --->
MUL(MUL(X, MUL(LAW(LAW(0, Y), Y), Z)), Z1)  [51, 30]

Adding Rule:  [71] INV(MUL(X, MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1))) --->
MUL(X, MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1))  [52, 29]

Adding Rule:  [72] INV(MUL(MUL(MUL(LAW(LAW(Y, 0), Y), Z), Z1), Z2)) --->
MUL(MUL(MUL(LAW(LAW(0, Y), Y), Z), Z1), Z2)  [52, 30]

Trying to orient equation:
  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1)) == MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))  [42, 41]

  To prove:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  >  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
        or:  MUL(LAW(LAW(Y, 0), Y), LAW(LAW(0, Y1), Y1))  <  MUL(LAW(LAW(0, Y), Y), LAW(LAW(Y1, 0), Y1))
  Here are some precedence suggestions:
       1.  MUL > LAW      2.  LAW > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
----- Return to Step 5 -----

Trying to orient equation:
  MUL(X, ADD(Y, Z)) == ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]
```

```
  To prove:  MUL(X, ADD(Y, Z))  >  ADD(MUL(X, Y), MUL(X, Z))
        or:  MUL(X, ADD(Y, Z))  <  ADD(MUL(X, Y), MUL(X, Z))
  Here are some precedence suggestions:
       1.  MUL > ADD      2.  ADD > MUL
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, MUL > ADD, is added.
Operator, ADD, given status: rl

----- Step 6 -----

Adding Rule:   [7] MUL(X, ADD(Y, Z)) ---> ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]

Adding Rule:   [8] MUL(ADD(X, Y), Z) ---> ADD(MUL(X, Z), MUL(Y, Z))  [USER, 6]

Adding Rule:   [9] ADD(MUL(X, Z), MUL(X, 0)) ---> MUL(X, Z)  [1, 7]

Adding Rule:  [10] ADD(MUL(Z, Z1), MUL(0, Z1)) ---> MUL(Z, Z1)  [1, 8]

Adding Rule:  [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> MUL(X, 0)  [2, 7]

Adding Rule:  [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> MUL(0, Z1)  [2, 8]

Adding Rule:  [13] ADD(INV(MUL(X, Z)), INV(MUL(X, 0))) ---> INV(MUL(X, Z))  [9, 6]

Adding Rule:  [14] ADD(INV(MUL(Z, Z1)), INV(MUL(0, Z1))) ---> INV(MUL(Z, Z1))  [10, 6]

Adding Rule:  [15] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, 0))) ---> MUL(X, MUL(X1, Z))  [9, 7]

Adding Rule:  [16] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, 0), Z1)) ---> MUL(MUL(X, Z), Z1)  [9, 8]

Adding Rule:  [17] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(0, Z1))) ---> MUL(X, MUL(Z, Z1))  [10, 7]

Adding Rule:  [18] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(0, Z1), Z2)) ---> MUL(MUL(Z, Z1), Z2)  [10, 8]

Adding Rule:  [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> INV(MUL(X, 0))  [11, 6]

Adding Rule:  [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> INV(MUL(0, Z1))  [12, 6]

Adding Rule:  [21] ADD(INV(MUL(X, 0)), MUL(X, Z)) ---> MUL(X, Z)  [9, 2]

Adding Rule:  [22] MUL(X, 0) ---> 0  [9, 2]

  Deleting rule:  [9] ADD(MUL(X, Z), MUL(X, 0)) ---> MUL(X, Z)  [1, 7]

  The right hand side is reducible:
     [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> MUL(X, 0)  [2, 7]

  Deleting rule: [13] ADD(INV(MUL(X, Z)), INV(MUL(X, 0))) ---> INV(MUL(X, Z))  [9, 6]

  Deleting rule: [15] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, 0))) ---> MUL(X, MUL(X1, Z))  [9, 7]

  Deleting rule: [16] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, 0), Z1)) ---> MUL(MUL(X, Z), Z1)  [9, 8]

  The right hand side is reducible:
     [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> INV(MUL(X, 0))  [11, 6]

  Deleting rule: [21] ADD(INV(MUL(X, 0)), MUL(X, Z)) ---> MUL(X, Z)  [9, 2]

Adding Rule:  [23] ADD(INV(MUL(0, Z)), MUL(Z1, Z)) ---> MUL(Z1, Z)  [10, 2]

Adding Rule:  [24] MUL(0, Z) ---> 0  [10, 2]
```

```
  Deleting rule: [10] ADD(MUL(Z, Z1), MUL(0, Z1)) ---> MUL(Z, Z1)  [1, 8]

  The right hand side is reducible:
     [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> MUL(0, Z1)  [2, 8]

  Deleting rule: [14] ADD(INV(MUL(Z, Z1)), INV(MUL(0, Z1))) ---> INV(MUL(Z, Z1))  [10, 6]

  Deleting rule: [17] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(0, Z1))) ---> MUL(X, MUL(Z, Z1))  [10, 7]

  Deleting rule: [18] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(0, Z1), Z2)) ---> MUL(MUL(Z, Z1), Z2)  [10, 8]

  The right hand side is reducible:
     [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> INV(MUL(0, Z1))  [12, 6]

  Deleting rule: [23] ADD(INV(MUL(0, Z)), MUL(Z1, Z)) ---> MUL(Z1, Z)  [10, 2]

Adding Rule:  [25] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, INV(Z)))) ---> 0  [11, 7]

Adding Rule:  [26] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, INV(Z)), Z1)) ---> 0  [11, 8]

Adding Rule:  [27] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(INV(Z), Z1))) ---> 0  [12, 7]

Adding Rule:  [28] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(INV(Z), Z1), Z2)) ---> 0  [12, 8]

Adding Rule:  [29] MUL(X, INV(Z)) ---> INV(MUL(X, Z))  [11, 2]

  Deleting rule: [11] ADD(MUL(X, Z), MUL(X, INV(Z))) ---> 0  [2, 7]

  Deleting rule: [19] ADD(INV(MUL(X, Z)), INV(MUL(X, INV(Z)))) ---> 0  [11, 6]

  Deleting rule: [25] ADD(MUL(X, MUL(X1, Z)), MUL(X, MUL(X1, INV(Z)))) ---> 0  [11, 7]

  Deleting rule: [26] ADD(MUL(MUL(X, Z), Z1), MUL(MUL(X, INV(Z)), Z1)) ---> 0  [11, 8]

Adding Rule:  [30] MUL(INV(Z), Z1) ---> INV(MUL(Z, Z1))  [12, 2]

  Deleting rule: [12] ADD(MUL(Z, Z1), MUL(INV(Z), Z1)) ---> 0  [2, 8]

  Deleting rule: [20] ADD(INV(MUL(Z, Z1)), INV(MUL(INV(Z), Z1))) ---> 0  [12, 6]

  Deleting rule: [27] ADD(MUL(X, MUL(Z, Z1)), MUL(X, MUL(INV(Z), Z1))) ---> 0  [12, 7]

  Deleting rule: [28] ADD(MUL(MUL(Z, Z1), Z2), MUL(MUL(INV(Z), Z1), Z2)) ---> 0  [12, 8]

Your system is canonical.

  [1] ADD(X, 0) ---> X  [USER, 3]
  [2] ADD(X, INV(X)) ---> 0  [USER, 4]
  [3] INV(0) ---> 0  [2, 1]
  [4] INV(INV(Z)) ---> Z  [2, 2]
  [6] INV(ADD(Z, Z1)) ---> ADD(INV(Z), INV(Z1))  [2, 5]
  [7] MUL(X, ADD(Y, Z)) ---> ADD(MUL(X, Y), MUL(X, Z))  [USER, 5]
  [8] MUL(ADD(X, Y), Z) ---> ADD(MUL(X, Z), MUL(Y, Z))  [USER, 6]
 [22] MUL(X, 0) ---> 0  [9, 2]
 [24] MUL(0, Z) ---> 0  [10, 2]
 [29] MUL(X, INV(Z)) ---> INV(MUL(X, Z))  [11, 2]
 [30] MUL(INV(Z), Z1) ---> INV(MUL(Z, Z1))  [12, 2]

Time used                            =  21.27 sec
Number of rules generated            = 30
Number of rules retained             = 11
Number of critical pairs             = 186
Number of unblocked critical pairs   = 49
Time spent in normalization          =   9.69 sec  (45.0 percent of time)
```

```
Time spent in unification            =   2.24 sec   (10.0 percent of time)
Time spent in ordering               =   0.44 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   2.69 sec   (12.0 percent of time)
Time spent in blocking               =   8.29 sec   (38.0 percent of time)
Total time (including ´undo´ action) = 22611.0 sec


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

10 A-modules

10.1 Axioms

Let x, y, z be arbitrary but fixed variables of type A, where A is an associative ring with unit, i.e. $A = (+, 0, *, 1)$ with the axioms:

$$\begin{aligned} x + y &= y + x && (1) \\ (x + y) + z &= x + (y + z) && (2) \\ x + 0 &= x && (3) \\ x + (-x) &= 0 && (4) \\ x * (y + z) &= (x * y) + (x * z) && (5) \\ (x + y) * z &= (x * z) + (y * z) && (6) \\ 1 * x &= x && (7) \\ x * 1 &= x && (8) \\ (x * y) * z &= x * (y * z) && (9) \end{aligned}$$

Let u, v, w be arbitrary but fixed variables of type M (introduced in the following).

Definition 10.1 *An A-module M is an algebraic structure based on an associative ring A with unit 1 and two binary operations*

- *an internal law denoted by* $\oplus$ *with* $\oplus : M \times M \to M$ *and*
- *an external law denoted by* $\cdot$ *with* $\cdot : A \times M \to M$,

such that M with $\oplus$ *is an Abelian group, i.e. it holds*

$$\begin{aligned} u \oplus v &= v \oplus u && (10) \\ (u \oplus v) \oplus w &= u \oplus (v \oplus w) && (11) \\ u \oplus O &= u && (12) \\ u \oplus i(u) &= O && (13) \end{aligned}$$

fulfilling the following additional axioms:

$$\begin{aligned} x \cdot (y \cdot u) &= (x * y) \cdot u && (14) \\ 1 \cdot u &= u && (15) \\ (x \cdot u) \oplus (y \cdot u) &= (x + y) \cdot u && (16) \\ x \cdot (u \oplus v) &= (x \cdot u) \oplus (x \cdot v) && (17) \end{aligned}$$

Definition 10.2 *An A-module over a commutative ring is an A-module with the additional property:*

$$x * y = y * x \qquad (18)$$

10.2 Example

A model of a module over a commutative ring is the three dimensional vector space over the real numbers. The internal law is the vector additon and the external law is the multiplication with a scalar.

10.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type A and let u, v, w be arbitrary but fixed variables of type M.

There is the following canonical system for A-modules (+ and $\oplus$ ac):

$$\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
1 * x &\longrightarrow x && (3)\\
x * 1 &\longrightarrow x && (4)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (5)\\
(x + y) * z &\longrightarrow (x * z) + (y * z) && (6)\\
-(0) &\longrightarrow 0 && (7)\\
-(-(z)) &\longrightarrow z && (8)\\
-(z + y) &\longrightarrow -(z) + -(y) && (9)\\
z * 0 &\longrightarrow 0 && (10)\\
0 * z &\longrightarrow 0 && (11)\\
x * -(z) &\longrightarrow -(x * z) && (12)\\
-(z) * y &\longrightarrow -(z * y) && (13)\\
x * (y * z) &\longrightarrow (x * y) * z && (14)\\
u \oplus O &\longrightarrow u && (15)\\
u \oplus i(u) &\longrightarrow O && (16)\\
i(O) &\longrightarrow O && (17)\\
i(i(u)) &\longrightarrow u && (18)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) && (19)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u && (20)\\
1 \cdot u &\longrightarrow u && (21)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) && (22)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) && (23)\\
x \cdot O &\longrightarrow O && (24)\\
0 \cdot u &\longrightarrow O && (25)\\
-(y) \cdot u &\longrightarrow i(y \cdot u) && (26)
\end{aligned}$$

$$x \cdot i(u) \longrightarrow i(x \cdot u) \quad (27)$$

There is the following canonical system for A-modules over a commutative ring (+, $*$ and $\oplus$ ac):

$$\begin{aligned}
x + 0 &\longrightarrow x & (1)\\
x + -(x) &\longrightarrow 0 & (2)\\
-(0) &\longrightarrow 0 & (3)\\
-(-(z)) &\longrightarrow z & (4)\\
-(z + y) &\longrightarrow -(z) + -(y) & (5)\\
(u \oplus O) &\longrightarrow u & (6)\\
(u \oplus i(u)) &\longrightarrow O & (7)\\
i(O) &\longrightarrow O & (8)\\
i(i(u)) &\longrightarrow u & (9)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) & (10)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u & (11)\\
1 \cdot u &\longrightarrow u & (12)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) & (13)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) & (14)\\
x \cdot O &\longrightarrow O & (15)\\
0 \cdot u &\longrightarrow O & (16)\\
-(y) \cdot u &\longrightarrow i(y \cdot u) & (17)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) & (18)\\
x * 1 &\longrightarrow x & (19)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) & (20)\\
z * 0 &\longrightarrow 0 & (21)\\
x * -(z) &\longrightarrow -(x * z) & (22)
\end{aligned}$$

10.4 Trace

The operation $\cdot$ is represented by @, $\oplus$ by #, i by I; +, $*$ and $-$ remain the same. Also 0, 1 and O remain the same. The output was edited, missing parts are indicated by "...".

```
RRL-> read test/10a.eq
New constant set is: { 1, 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]
 5. ((X * Y) * Z) == (X * (Y * Z))  [USER, 5]
```

```
 6. (1 * X) == X  [USER, 6]
 7. (X * 1) == X  [USER, 7]
 8. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 8]
 9. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...


Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]

Time used                            =  30.50 sec
Number of rules generated            = 50
Number of rules retained             = 14
Number of critical pairs             = 329
Number of unblocked critical pairs   = 84
Time spent in normalization          =  17.45 sec  (57.0 percent of time)
Time spent in unification            =   4.76 sec   (15.0 percent of time)
Time spent in ordering               =   0.62 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   3.02 sec   (9.0 percent of time)
Time spent in blocking               =   1.44 sec   (4.0 percent of time)
Total time (including ´undo´ action) = 3069.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/10b.eq
New constant set is: { 0 }
Equations read in are:
 1. (X # Y) == (Y # X)  [USER, 10]
 2. ((X # Y) # Z) == (X # (Y # Z))  [USER, 11]
 3. (X # 0) == X  [USER, 12]
 4. (X # I(X)) == 0  [USER, 13]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.
```

```
  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]

Time used                            =  34.58 sec
Number of rules generated            = 56
Number of rules retained             = 19
Number of critical pairs             = 380
Number of unblocked critical pairs   = 113
Time spent in normalization          =  19.51 sec  (56.0 percent of time)
Time spent in unification            =   5.53 sec   (15.0 percent of time)
Time spent in ordering               =   0.74 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   3.21 sec   (9.0 percent of time)
Time spent in blocking               =   1.61 sec   (4.0 percent of time)
Total time (including 'undo' action) = 418.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/10c.eq
Equations read in are:
 1. (X @ (Y @ Z)) == ((X * Y) @ Z)  [USER, 14]
 2. (1 @ X) == X  [USER, 15]
 3. ((X @ Z) # (Y @ Z)) == ((X + Y) @ Z)  [USER, 16]
 4. (X @ (Y # Z)) == ((X @ Y) # (X @ Z))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
```

```
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]
 [57] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [65] (X @ 0) ---> O  [53, 62]
 [72] (0 @ Z) ---> O  [66, 52]
 [77] (-(Y) @ Z) ---> I((Y @ Z))  [76, 57]
 [79] (X @ I(Z)) ---> I((X @ Z))  [58, 78]

Time used                          =  53.56 sec
Number of rules generated          = 79
Number of rules retained           = 27
Number of critical pairs           = 534
Number of unblocked critical pairs  = 149
Time spent in normalization        =  28.94 sec  (54.0 percent of time)
Time spent in unification          =   8.27 sec   (15.0 percent of time)
Time spent in ordering             =   1.10 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   5.86 sec   (10.0 percent of time)
Time spent in blocking             =   4.51 sec   (8.0 percent of time)
Total time (including ´undo´ action) = 6691.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/10d.eq
Equations read in are:
 1. (X * Y) == (Y * X)  [USER, 18]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]
 [57] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [65] (X @ 0) ---> O  [53, 62]
 [72] (0 @ Z) ---> O  [66, 52]
```

```
 [77] (-(Y) @ Z) ---> I((Y @ Z))  [76, 57]
 [79] (X @ I(Z)) ---> I((X @ Z))  [58, 78]
 [80] (X * 1) ---> X  [DELETED, 3]
 [81] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
 [82] (Z * 0) ---> 0  [DELETED, 32]
 [83] (X * -(Z)) ---> -((X * Z))  [DELETED, 48]

Time used                           =  68.90 sec
Number of rules generated           = 83
Number of rules retained            = 22
Number of critical pairs            = 633
Number of unblocked critical pairs  = 196
Time spent in normalization         =  38.10 sec  (55.0 percent of time)
Time spent in unification           =   9.61 sec   (13.0 percent of time)
Time spent in ordering              =   1.14 sec   (1.0 percent of time)
Time spent in simplifying the rules =   6.02 sec   (8.0 percent of time)
Time spent in blocking              =   4.87 sec   (7.0 percent of time)
Total time (including 'undo' action) = 1548.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

11 A-bimodules

11.1 Axioms

Let x, y, z be arbitrary but fixed variables of type A, where A is an associative ring with unit. Let u, v, w be arbitrary but fixed variables of type M.

Definition 11.1 *An A-bimodule M is an A-module together with a new binary operation*

- *which is another external law denoted by $\circ$ with $\circ : M \times A \to M$,*

such that the following axioms hold:

$$
\begin{aligned}
(u \circ x) \circ y &= u \circ (x * y) && (18)\\
u \circ 1 &= u && (19)\\
(u \circ x) \oplus (u \circ y) &= u \circ (x + y) && (20)\\
(u \oplus v) \circ x &= (u \circ x) \oplus (v \circ x) && (21)
\end{aligned}
$$

and additionally there is the following connection between both external laws:

$$(x \cdot u) \circ y = x \cdot (u \circ y) \tag{22}$$

Definition 11.2 *An A-bimodule over a commutative ring is an A-bimodule where additionally the following axiom holds:*

$$x * y = y * x \tag{23}$$

11.2 Example

A model of a bimodule is the three dimensional vector space over a skew field (e.g. the quaternions). The internal law is the vector addition and the external laws are the multiplication with a scalar – one coming in front, one coming at the end. But this is not a bimodule over a commutative ring, since the commutative law does normally not hold in a skew field.

11.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type A and let u, v, w be arbitrary but fixed variables of type M.

There is the following canonical system for A-bimodules (+ and $\oplus$ ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
1 * x &\longrightarrow x && (3)\\
x * 1 &\longrightarrow x && (4)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (5)\\
(x + y) * z &\longrightarrow (x * z) + (y * z) && (6)\\
-(0) &\longrightarrow 0 && (7)\\
-(-(z)) &\longrightarrow z && (8)\\
-(z + y) &\longrightarrow -(z) + -(y) && (9)\\
z * 0 &\longrightarrow 0 && (10)\\
0 * z &\longrightarrow 0 && (11)\\
x * -(z) &\longrightarrow -(x * z) && (12)\\
-(z) * y &\longrightarrow -(z * y) && (13)\\
x * (y * z) &\longrightarrow (x * y) * z && (14)\\
u \oplus O &\longrightarrow u && (15)\\
u \oplus i(u) &\longrightarrow O && (16)\\
i(O) &\longrightarrow O && (17)\\
i(i(u)) &\longrightarrow u && (18)\\
i(v \oplus w) &\longrightarrow i(v) \oplus i(w) && (19)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u && (20)\\
1 \cdot u &\longrightarrow u && (21)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) && (22)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) && (23)\\
x \cdot O &\longrightarrow O && (24)\\
0 \cdot u &\longrightarrow O && (25)\\
-(y) \cdot u &\longrightarrow i(y \cdot u) && (26)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) && (27)\\
u \circ (y * z) &\longrightarrow (u \circ y) \circ z && (28)\\
u \circ 1 &\longrightarrow u && (29)\\
u \circ (y + z) &\longrightarrow (u \circ y) \oplus (u \circ z) && (30)\\
(u \oplus v) \circ z &\longrightarrow (u \circ z) \oplus (v \circ z) && (31)\\
O \circ z &\longrightarrow O && (32)\\
u \circ 0 &\longrightarrow O && (33)\\
u \circ -(z) &\longrightarrow i(u \circ z) && (34)
\end{aligned}
$$

$$
\begin{aligned}
i(u) \circ z &\longrightarrow i(u \circ z) && (35)\\
x \cdot (u \circ z)) &\longrightarrow (x \cdot u) \circ z && (36)
\end{aligned}
$$

There is the following canonical system for A-bimodules over a commutative ring (+, $*$ and $\oplus$ ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x && (1)\\
x + -(x) &\longrightarrow 0 && (2)\\
x * 1 &\longrightarrow x && (3)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) && (4)\\
-(0) &\longrightarrow 0 && (5)\\
-(-(z)) &\longrightarrow z && (6)\\
-(z + y) &\longrightarrow -(z) + -(y) && (7)\\
z * 0 &\longrightarrow 0 && (8)\\
x * -(z) &\longrightarrow -(x * z) && (9)\\
u \oplus O &\longrightarrow u && (10)\\
u \oplus i(u) &\longrightarrow O && (11)\\
i(O) &\longrightarrow O && (12)\\
i(i(u)) &\longrightarrow u && (13)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) && (14)\\
1 \cdot u &\longrightarrow u && (15)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) && (16)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) && (17)\\
0 \cdot u &\longrightarrow O && (18)\\
x \cdot O &\longrightarrow O && (19)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) && (20)\\
-(z) \cdot u &\longrightarrow i(z \cdot u) && (21)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u && (22)\\
(u \circ y) \circ z &\longrightarrow u \circ (y * z) && (23)\\
u \circ 1 &\longrightarrow u && (24)\\
u \circ (y + z) &\longrightarrow (u \circ y) \oplus (u \circ z) && (25)\\
(u \oplus v) \circ z &\longrightarrow (u \circ z) \oplus (v \circ z) && (26)\\
O \circ z &\longrightarrow O && (27)\\
u \circ 0 &\longrightarrow O && (28)\\
u \circ -(y) &\longrightarrow i(u \circ y) && (29)\\
i(u) \circ z &\longrightarrow i(u \circ z) && (30)\\
(x \cdot u) \circ z &\longrightarrow x \cdot (u \circ z) && (31)
\end{aligned}
$$

11.4 Trace

The operation $\cdot$ is represented by @, $\oplus$ by #, $\circ$ by ^, i by I; $+$, $*$, $-$, 0, 1, O remain the same. The output was edited, missing parts are indicated by "...".

```
RRL-> read test/11a.eq
New constant set is: { 1, 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]
 5. ((X * Y) * Z) == (X * (Y * Z))  [USER, 5]
 6. (1 * X) == X  [USER, 6]
 7. (X * 1) == X  [USER, 7]
 8. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 8]
 9. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]

Time used                          =  30.54 sec
Number of rules generated          = 50
Number of rules retained           = 14
Number of critical pairs           = 329
Number of unblocked critical pairs  = 84
Time spent in normalization        =  17.30 sec  (56.0 percent of time)
Time spent in unification          =   4.79 sec   (15.0 percent of time)
Time spent in ordering             =   0.64 sec   (2.0 percent of time)
Time spent in simplifying the rules =   2.93 sec   (9.0 percent of time)
Time spent in blocking             =   1.38 sec   (4.0 percent of time)
Total time (including 'undo' action) = 3064.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11b.eq
New constant set is: { O }
Equations read in are:
 1. (X # Y) == (Y # X)  [USER, 10]
 2. ((X # Y) # Z) == (X # (Y # Z))  [USER, 11]
```

```
 3. (X # O) == X  [USER, 12]
 4. (X # I(X)) == O  [USER, 13]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]

Time used                          =  34.63 sec
Number of rules generated          = 56
Number of rules retained           = 19
Number of critical pairs           = 380
Number of unblocked critical pairs  = 113
Time spent in normalization        =  19.31 sec  (55.0 percent of time)
Time spent in unification          =   5.59 sec   (16.0 percent of time)
Time spent in ordering             =   0.74 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   3.13 sec   (9.0 percent of time)
Time spent in blocking             =   1.55 sec   (4.0 percent of time)
Total time (including 'undo' action) = 418.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11c.eq
Equations read in are:
 1. (X @ (Y @ Z)) == ((X * Y) @ Z)  [USER, 14]
 2. (1 @ X) == X  [USER, 15]
 3. ((X @ Z) # (Y @ Z)) == ((X + Y) @ Z)  [USER, 16]
 4. (X @ (Y # Z)) == ((X @ Y) # (X @ Z))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...
```

```
Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # 0) ---> X  [USER, 12]
 [52] (X # I(X)) ---> 0  [USER, 13]
 [53] I(0) ---> 0  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]
 [57] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [65] (X @ 0) ---> 0  [53, 62]
 [72] (0 @ Z) ---> 0  [66, 52]
 [77] (-(Y) @ Z) ---> I((Y @ Z))  [76, 57]
 [79] (X @ I(Z)) ---> I((X @ Z))  [58, 78]

Time used                           =  53.68 sec
Number of rules generated           = 79
Number of rules retained            = 27
Number of critical pairs            = 534
Number of unblocked critical pairs  = 149
Time spent in normalization         =  28.63 sec  (53.0 percent of time)
Time spent in unification           =   8.41 sec   (15.0 percent of time)
Time spent in ordering              =   1.10 sec   (2.0 percent of time)
Time spent in simplifying the rules =   5.79 sec   (10.0 percent of time)
Time spent in blocking              =   4.46 sec   (8.0 percent of time)
Total time (including 'undo' action) = 6692.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11d.eq
Equations read in are:
 1. ((X ^ Y) ^ Z) == (X ^ (Y * Z))  [USER, 18]
 2. (X ^ 1) == X  [USER, 19]
 3. ((X ^ Y) # (X ^ Z)) == (X ^ (Y + Z))  [USER, 20]
 4. ((X # Y) ^ Z) == ((X ^ Z) # (Y ^ Z))  [USER, 21]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
```

```
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]
 [57] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [65] (X @ O) ---> O  [53, 62]
 [72] (0 @ Z) ---> O  [66, 52]
 [77] (-(Y) @ Z) ---> I((Y @ Z))  [76, 57]
 [79] (X @ I(Z)) ---> I((X @ Z))  [58, 78]
 [80] (X ^ (Y * Z)) ---> ((X ^ Y) ^ Z)  [USER, 18]
 [81] (X ^ 1) ---> X  [USER, 19]
 [82] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 20]
 [83] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 21]
 [93] (O ^ Z) ---> O  [92, 85]
 [101] (Z ^ 0) ---> O  [91, 52]
 [110] (X ^ -(Z)) ---> I((X ^ Z))  [12, 108]
 [112] (I(X) ^ Z) ---> I((X ^ Z))  [111, 54]

Time used                          =  77.47 sec
Number of rules generated          = 112
Number of rules retained           = 35
Number of critical pairs           = 722
Number of unblocked critical pairs  = 180
Time spent in normalization         =  38.67 sec  (49.0 percent of time)
Time spent in unification           =  11.10 sec   (14.0 percent of time)
Time spent in ordering              =   2.79 sec   (3.0 percent of time)
Time spent in simplifying the rules  =   8.67 sec   (11.0 percent of time)
Time spent in blocking              =   4.97 sec   (6.0 percent of time)
Total time (including 'undo' action) = 2411.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11e.eq
Equations read in are:
 1. ((X @ Y) ^ Z) == (X @ (Y ^ Z))  [USER, 22]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.
```

```
  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] I(O) ---> O  [52, 51]
 [54] I(I(Z)) ---> Z  [52, 52]
 [56] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 55]
 [57] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [65] (X @ O) ---> O  [53, 62]
 [72] (0 @ Z) ---> O  [66, 52]
 [77] (-(Y) @ Z) ---> I((Y @ Z))  [76, 57]
 [79] (X @ I(Z)) ---> I((X @ Z))  [58, 78]
 [80] (X ^ (Y * Z)) ---> ((X ^ Y) ^ Z)  [USER, 18]
 [81] (X ^ 1) ---> X  [USER, 19]
 [82] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 20]
 [83] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 21]
 [93] (O ^ Z) ---> O  [92, 85]
 [101] (Z ^ 0) ---> O  [91, 52]
 [110] (X ^ -(Z)) ---> I((X ^ Z))  [12, 108]
 [112] (I(X) ^ Z) ---> I((X ^ Z))  [111, 54]
 [113] (X @ (Y ^ Z)) ---> ((X @ Y) ^ Z)  [USER, 22]

Time used                          =  78.03 sec
Number of rules generated          = 113
Number of rules retained           = 36
Number of critical pairs           = 736
Number of unblocked critical pairs  = 180
Time spent in normalization         =  38.93 sec  (49.0 percent of time)
Time spent in unification           =  11.12 sec   (14.0 percent of time)
Time spent in ordering              =   2.80 sec   (3.0 percent of time)
Time spent in simplifying the rules  =   8.72 sec   (11.0 percent of time)
Time spent in blocking              =   4.98 sec   (6.0 percent of time)
Total time (including 'undo' action) = 202.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> init
RRL is initialized.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11a.eq
New constant set is: { 1, 0 }
Equations read in are:
```

```
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]
 5. ((X * Y) * Z) == (X * (Y * Z))  [USER, 5]
 6. (1 * X) == X  [USER, 6]
 7. (X * 1) == X  [USER, 7]
 8. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 8]
 9. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11f.eq
Equations read in are:
 1. (X * Y) == (Y * X)  [USER, 10]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [7] (X * 1) ---> X  [DELETED, 3]
  [8] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
  [9] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [18] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [24] (Z * 0) ---> 0  [21, 12]
 [53] F1(Z) ---> 0  [22, 2]
 [55] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [54, 54]

Time used                          =  56.36 sec
Number of rules generated          = 55
Number of rules retained           = 10
Number of critical pairs           = 432
Number of unblocked critical pairs = 182
Time spent in normalization        =  34.77 sec  (61.0 percent of time)
Time spent in unification          =   7.06 sec   (12.0 percent of time)
Time spent in ordering             =   0.62 sec   (1.0 percent of time)
Time spent in simplifying the rules =   5.20 sec   (9.0 percent of time)
Time spent in blocking             =   3.18 sec   (5.0 percent of time)
Total time (including 'undo' action) = 5649.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11b.eq
New constant set is: { O }
Equations read in are:
 1. (X # Y) == (Y # X)  [USER, 11]
 2. ((X # Y) # Z) == (X # (Y # Z))  [USER, 12]
 3. (X # O) == X  [USER, 13]
 4. (X # I(X)) == O  [USER, 14]
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [7] (X * 1) ---> X  [DELETED, 3]
  [8] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
  [9] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [18] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [24] (Z * 0) ---> 0  [21, 12]
 [53] F1(Z) ---> 0  [22, 2]
 [55] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [54, 54]
 [56] (X # O) ---> X  [USER, 13]
 [57] (X # I(X)) ---> O  [USER, 14]
 [58] I(O) ---> O  [57, 56]
 [59] I(I(Z)) ---> Z  [57, 57]
 [61] I((Z # Z1)) ---> (I(Z) # I(Z1))  [57, 60]

Time used                          =  60.44 sec
Number of rules generated          = 61
Number of rules retained           = 15
Number of critical pairs           = 483
Number of unblocked critical pairs  = 211
Time spent in normalization        =  35.72 sec  (59.0 percent of time)
Time spent in unification          =   7.84 sec   (12.0 percent of time)
Time spent in ordering             =   0.73 sec   (1.0 percent of time)
Time spent in simplifying the rules =   5.35 sec   (8.0 percent of time)
Time spent in blocking             =   3.37 sec   (5.0 percent of time)
Total time (including 'undo' action) = 419.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11c.eq
Equations read in are:
 1. (X @ (Y @ Z)) == ((X * Y) @ Z)  [USER, 15]
 2. (1 @ X) == X  [USER, 16]
 3. ((X @ Z) # (Y @ Z)) == ((X + Y) @ Z)  [USER, 17]
 4. (X @ (Y # Z)) == ((X @ Y) # (X @ Z))  [USER, 18]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [7] (X * 1) ---> X  [DELETED, 3]
  [8] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
  [9] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
```

```
 [18] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [24] (Z * 0) ---> 0  [21, 12]
 [53] F1(Z) ---> 0  [22, 2]
 [55] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [54, 54]
 [56] (X # O) ---> X  [USER, 13]
 [57] (X # I(X)) ---> O  [USER, 14]
 [58] I(O) ---> O  [57, 56]
 [59] I(I(Z)) ---> Z  [57, 57]
 [61] I((Z # Z1)) ---> (I(Z) # I(Z1))  [57, 60]
 [62] (1 @ X) ---> X  [USER, 16]
 [63] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 17]
 [64] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 18]
 [79] (0 @ Z) ---> O  [69, 57]
 [87] (X @ O) ---> O  [65, 57]
 [90] (X @ I(Z)) ---> I((X @ Z))  [67, 57]
 [91] (-(Z) @ Z1) ---> I((Z @ Z1))  [68, 57]
 [92] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 15]

Time used                            =  84.00 sec
Number of rules generated            = 92
Number of rules retained             = 23
Number of critical pairs             = 699
Number of unblocked critical pairs   = 254
Time spent in normalization          =  47.87 sec  (56.0 percent of time)
Time spent in unification            =  11.57 sec   (13.0 percent of time)
Time spent in ordering               =   1.33 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   7.75 sec   (9.0 percent of time)
Time spent in blocking               =  11.47 sec   (13.0 percent of time)
Total time (including 'undo' action) = 18727.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11d.eq
Equations read in are:
 1. ((X ^ Y) ^ Z) == (X ^ (Y * Z))  [USER, 19]
 2. (X ^ 1) == X  [USER, 20]
 3. ((X ^ Y) # (X ^ Z)) == (X ^ (Y + Z))  [USER, 21]
 4. ((X # Y) ^ Z) == ((X ^ Z) # (Y ^ Z))  [USER, 22]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [7] (X * 1) ---> X  [DELETED, 3]
  [8] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
  [9] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [18] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [24] (Z * 0) ---> 0  [21, 12]
 [53] F1(Z) ---> 0  [22, 2]
 [55] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [54, 54]
 [56] (X # O) ---> X  [USER, 13]
 [57] (X # I(X)) ---> O  [USER, 14]
 [58] I(O) ---> O  [57, 56]
```

```
 [59] I(I(Z)) ---> Z  [57, 57]
 [61] I((Z # Z1)) ---> (I(Z) # I(Z1))  [57, 60]
 [62] (1 @ X) ---> X  [USER, 16]
 [63] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 17]
 [64] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 18]
 [79] (O @ Z) ---> 0  [69, 57]
 [87] (X @ 0) ---> 0  [65, 57]
 [90] (X @ I(Z)) ---> I((X @ Z))  [67, 57]
 [91] (-(Z) @ Z1) ---> I((Z @ Z1))  [68, 57]
 [92] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 15]
 [93] ((X ^ Y) ^ Z) ---> (X ^ (Y * Z))  [USER, 19]
 [94] (X ^ 1) ---> X  [USER, 20]
 [95] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 21]
 [96] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 22]
 [103] (0 ^ Z) ---> 0  [102, 93]
 [110] (Z ^ O) ---> 0  [101, 57]
 [116] (X ^ -(Y)) ---> I((X ^ Y))  [115, 93]
 [118] (I(X) ^ Z) ---> I((X ^ Z))  [94, 117]

Time used                           = 102.67 sec
Number of rules generated           = 118
Number of rules retained            = 31
Number of critical pairs            = 841
Number of unblocked critical pairs  = 285
Time spent in normalization         =  55.75 sec  (54.0 percent of time)
Time spent in unification           =  13.08 sec   (12.0 percent of time)
Time spent in ordering              =   1.60 sec   (1.0 percent of time)
Time spent in simplifying the rules =  11.45 sec   (11.0 percent of time)
Time spent in blocking              =  14.70 sec   (14.0 percent of time)
Total time (including 'undo' action) = 10521.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/11e.eq
Equations read in are:
 1. ((X @ Y) ^ Z) == (X @ (Y ^ Z))  [USER, 23]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + O) ---> X  [USER, 3]
  [2] (X + -(X)) ---> O  [USER, 4]
  [7] (X * 1) ---> X  [DELETED, 3]
  [8] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [DELETED, 5]
  [9] -(O) ---> O  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [18] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [24] (Z * O) ---> O  [21, 12]
 [53] F1(Z) ---> O  [22, 2]
 [55] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [54, 54]
 [56] (X # 0) ---> X  [USER, 13]
 [57] (X # I(X)) ---> 0  [USER, 14]
 [58] I(0) ---> O  [57, 56]
 [59] I(I(Z)) ---> Z  [57, 57]
 [61] I((Z # Z1)) ---> (I(Z) # I(Z1))  [57, 60]
```

```
 [62] (1 @ X) ---> X  [USER, 16]
 [63] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 17]
 [64] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 18]
 [79] (0 @ Z) ---> 0  [69, 57]
 [87] (X @ 0) ---> 0  [65, 57]
 [90] (X @ I(Z)) ---> I((X @ Z))  [67, 57]
 [91] (-(Z) @ Z1) ---> I((Z @ Z1))  [68, 57]
 [92] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 15]
 [93] ((X ^ Y) ^ Z) ---> (X ^ (Y * Z))  [USER, 19]
 [94] (X ^ 1) ---> X  [USER, 20]
 [95] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 21]
 [96] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 22]
 [103] (0 ^ Z) ---> 0  [102, 93]
 [110] (Z ^ 0) ---> 0  [101, 57]
 [116] (X ^ -(Y)) ---> I((X ^ Y))  [115, 93]
 [118] (I(X) ^ Z) ---> I((X ^ Z))  [94, 117]
 [119] ((X @ Y) ^ Z) ---> (X @ (Y ^ Z))  [USER, 23]

Time used                           = 103.25 sec
Number of rules generated           = 119
Number of rules retained            = 32
Number of critical pairs            = 855
Number of unblocked critical pairs  = 285
Time spent in normalization         =  56.03 sec  (54.0 percent of time)
Time spent in unification           =  13.09 sec   (12.0 percent of time)
Time spent in ordering              =   1.60 sec   (1.0 percent of time)
Time spent in simplifying the rules =  11.51 sec   (11.0 percent of time)
Time spent in blocking              =  14.71 sec   (14.0 percent of time)
Total time (including 'undo' action) = 72.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

12 A-rings

12.1 Axioms

Let x, y, z be arbitrary but fixed variables of type A, where A is an associative ring with unit. Let u, v, w be arbitrary but fixed variables of type M, where M is a ring.

Definition 12.1 *An A-ring M is an A-bimodule based on the associative ring A with unit, where M is a ring itself, i.e. with a new operation* $\otimes$ *with* $\otimes : M \times M \to M$ *fulfilling:*

$$u \otimes (v \oplus w) = (u \otimes v) \oplus (u \otimes w) \tag{23}$$
$$(u \oplus v) \otimes w = (u \otimes w) \oplus (v \otimes w) \tag{24}$$

Furthermore the following equations hold:

$$(x \cdot u) \otimes v = x \cdot (u \otimes v) \tag{25}$$
$$(u \circ x) \otimes v = u \otimes (x \cdot v) \tag{26}$$
$$u \otimes (v \circ x) = (u \otimes v) \circ x \tag{27}$$

12.2 Example

A model of an A-ring is the three dimensional vector space over the real numbers. The internal law is the vector addition and the external laws are the multiplication with scalars. The cross product forms a ring from the vector space. In that case the left and the right structure coincide.

12.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type A and let u, v, w be arbitrary but fixed variables of type M.

There is the following canonical system for A-rings (+ and $\oplus$ ac):

$$x + 0 \longrightarrow x \tag{1}$$
$$x + -(x) \longrightarrow 0 \tag{2}$$
$$1 * x \longrightarrow x \tag{3}$$
$$x * 1 \longrightarrow x \tag{4}$$
$$x * (y + z) \longrightarrow (x * y) + (x * z) \tag{5}$$
$$(x + y) * z \longrightarrow (x * z) + (y * z) \tag{6}$$
$$-(0) \longrightarrow 0 \tag{7}$$
$$-(-(z)) \longrightarrow z \tag{8}$$
$$-(x + y) \longrightarrow -(x) + -(y) \tag{9}$$

$$
\begin{aligned}
z * 0 &\longrightarrow 0 && (10)\\
0 * z &\longrightarrow 0 && (11)\\
x * -(z) &\longrightarrow -(x * z) && (12)\\
-(x) * y &\longrightarrow -(x * y) && (13)\\
x * (y * z) &\longrightarrow (x * y) * z && (14)\\
u \oplus O &\longrightarrow u && (15)\\
u \oplus i(u) &\longrightarrow O && (16)\\
i(O) &\longrightarrow O && (17)\\
i(i(u)) &\longrightarrow u && (18)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) && (19)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u && (20)\\
1 \cdot u &\longrightarrow u && (21)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) && (22)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) && (23)\\
x \cdot O &\longrightarrow O && (24)\\
0 \cdot u &\longrightarrow O && (25)\\
-(y) \cdot u &\longrightarrow i(y \cdot u) && (26)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) && (27)\\
u \circ (y * z) &\longrightarrow (u \circ y) \circ z && (28)\\
u \circ 1 &\longrightarrow u && (29)\\
u \circ (y + z) &\longrightarrow (u \circ y) \oplus (u \circ z) && (30)\\
(u \oplus v) \circ z &\longrightarrow (u \circ z) \oplus (v \circ z) && (31)\\
O \circ z &\longrightarrow O && (32)\\
u \circ 0 &\longrightarrow O && (33)\\
u \circ -(z) &\longrightarrow i(u \circ z) && (34)\\
i(u) \circ z &\longrightarrow i(u \circ z) && (35)\\
x \cdot (u \circ z) &\longrightarrow (x \cdot u) \circ z && (36)\\
u \otimes (v \oplus w) &\longrightarrow (u \otimes v) \oplus (u \otimes w) && (37)\\
(u \oplus v) \otimes w &\longrightarrow (u \otimes w) \oplus (v \otimes w) && (38)\\
u \otimes O &\longrightarrow O && (39)\\
O \otimes u &\longrightarrow O && (40)\\
u \otimes i(v) &\longrightarrow i(u \otimes v) && (41)\\
i(u) \otimes v &\longrightarrow i(u \otimes v) && (42)\\
x \cdot (v \otimes w) &\longrightarrow (x \cdot v) \otimes w && (43)\\
(u \otimes v) \circ z &\longrightarrow u \otimes (v \circ z) && (44)\\
(u \circ x) \otimes v &\longrightarrow u \otimes (x \cdot v) && (45)
\end{aligned}
$$

12.4 Trace

The operation $\cdot$ is represented by @, $\oplus$ by #, $\circ$ by ^, $\otimes$ by \$, i by I; $+$, $*$, $-$, 0, 1, O remain the same. The output was edited, missing parts are indicated by "...".

```
RRL-> read test/12a.eq
New constant set is: { 1, 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]
 5. ((X * Y) * Z) == (X * (Y * Z))  [USER, 5]
 6. (1 * X) == X  [USER, 6]
 7. (X * 1) == X  [USER, 7]
 8. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 8]
 9. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]

Time used                          =  30.32 sec
Number of rules generated          = 50
Number of rules retained           = 14
Number of critical pairs           = 329
Number of unblocked critical pairs  = 84
Time spent in normalization        =  17.25 sec  (56.0 percent of time)
Time spent in unification          =   4.78 sec   (15.0 percent of time)
Time spent in ordering             =   0.63 sec   (2.0 percent of time)
Time spent in simplifying the rules  =   3.02 sec   (9.0 percent of time)
Time spent in blocking             =   1.36 sec   (4.0 percent of time)
Total time (including 'undo' action) = 3048.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/12b.eq
New constant set is: { 0 }
Equations read in are:
 1. (X # Y) == (Y # X)  [USER, 10]
 2. ((X # Y) # Z) == (X # (Y # Z))  [USER, 11]
```

```
 3. (X # 0) == X  [USER, 12]
 4. (X # I(X)) == 0  [USER, 13]
 5. (X $ (Y # Z)) == ((X $ Y) # (X $ Z))  [USER, 14]
 6. ((X # Y) $ Z) == ((X $ Z) # (Y $ Z))  [USER, 15]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # 0) ---> X  [USER, 12]
 [52] (X # I(X)) ---> 0  [USER, 13]
 [53] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 14]
 [54] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 15]
 [55] I(0) ---> 0  [52, 51]
 [60] I(I(Z)) ---> Z  [52, 52]
 [62] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 61]
 [72] (X $ 0) ---> 0  [56, 52]
 [74] (0 $ Z) ---> 0  [57, 52]
 [79] (X $ I(Z)) ---> I((X $ Z))  [58, 52]
 [80] (I(Z) $ Z1) ---> I((Z $ Z1))  [59, 52]

Time used                          =  54.37 sec
Number of rules generated          = 80
Number of rules retained           = 25
Number of critical pairs           = 515
Number of unblocked critical pairs  = 133
Time spent in normalization        =  29.56 sec  (54.0 percent of time)
Time spent in unification          =   9.48 sec   (17.0 percent of time)
Time spent in ordering             =   1.05 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   5.11 sec   (9.0 percent of time)
Time spent in blocking             =   2.11 sec   (3.0 percent of time)
Total time (including 'undo' action) = 2427.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/12c.eq
Equations read in are:
 1. (X @ (Y @ Z)) == ((X * Y) @ Z)  [USER, 16]
 2. (1 @ X) == X  [USER, 17]
 3. ((X + Y) @ Z) == ((X @ Z) # (Y @ Z))  [USER, 18]
 4. (X @ (Y # Z)) == ((X @ Y) # (X @ Z))  [USER, 19]
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 14]
 [54] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 15]
 [55] I(O) ---> O  [52, 51]
 [60] I(I(Z)) ---> Z  [52, 52]
 [62] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 61]
 [72] (X $ O) ---> O  [56, 52]
 [74] (O $ Z) ---> O  [57, 52]
 [79] (X $ I(Z)) ---> I((X $ Z))  [58, 52]
 [80] (I(Z) $ Z1) ---> I((Z $ Z1))  [59, 52]
 [81] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 16]
 [82] (1 @ X) ---> X  [USER, 17]
 [83] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 18]
 [84] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 19]
 [91] (X @ O) ---> O  [90, 81]
 [99] (0 @ Z) ---> O  [89, 52]
 [106] (-(Y) @ Z) ---> I((Y @ Z))  [105, 81]
 [108] (X @ I(Z)) ---> I((X @ Z))  [82, 107]

Time used                         =  75.61 sec
Number of rules generated         = 108
Number of rules retained          = 33
Number of critical pairs          = 661
Number of unblocked critical pairs  = 164
Time spent in normalization       =  39.51 sec  (52.0 percent of time)
Time spent in unification         =  12.18 sec   (16.0 percent of time)
Time spent in ordering            =   1.48 sec   (1.0 percent of time)
Time spent in simplifying the rules  =  10.12 sec   (13.0 percent of time)
Time spent in blocking            =   2.56 sec   (3.0 percent of time)
Total time (including 'undo' action) = 2153.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/12d.eq
Equations read in are:
 1. ((X ^ Y) ^ Z) == (X ^ (Y * Z))  [USER, 20]
 2. (X ^ 1) == X  [USER, 21]
```

```
 3. (X ^ (Y + Z)) == ((X ^ Y) # (X ^ Z))  [USER, 22]
 4. ((X # Y) ^ Z) == ((X ^ Z) # (Y ^ Z))  [USER, 23]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # 0) ---> X  [USER, 12]
 [52] (X # I(X)) ---> 0  [USER, 13]
 [53] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 14]
 [54] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 15]
 [55] I(0) ---> 0  [52, 51]
 [60] I(I(Z)) ---> Z  [52, 52]
 [62] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 61]
 [72] (X $ 0) ---> 0  [56, 52]
 [74] (0 $ Z) ---> 0  [57, 52]
 [79] (X $ I(Z)) ---> I((X $ Z))  [58, 52]
 [80] (I(Z) $ Z1) ---> I((Z $ Z1))  [59, 52]
 [81] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 16]
 [82] (1 @ X) ---> X  [USER, 17]
 [83] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 18]
 [84] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 19]
 [91] (X @ 0) ---> 0  [90, 81]
 [99] (0 @ Z) ---> 0  [89, 52]
 [106] (-(Y) @ Z) ---> I((Y @ Z))  [105, 81]
 [108] (X @ I(Z)) ---> I((X @ Z))  [82, 107]
 [109] (X ^ (Y * Z)) ---> ((X ^ Y) ^ Z)  [USER, 20]
 [110] (X ^ 1) ---> X  [USER, 21]
 [111] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 22]
 [112] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 23]
 [122] (0 ^ Z) ---> 0  [121, 114]
 [132] (Z ^ 0) ---> 0  [120, 52]
 [143] (X ^ -(Z)) ---> I((X ^ Z))  [12, 141]
 [145] (I(X) ^ Z) ---> I((X ^ Z))  [144, 60]

Time used                           = 107.89 sec
Number of rules generated           = 145
Number of rules retained            = 41
Number of critical pairs            = 857
Number of unblocked critical pairs  = 195
Time spent in normalization         =  54.89 sec  (50.0 percent of time)
Time spent in unification           =  13.94 sec   (12.0 percent of time)
Time spent in ordering              =   1.95 sec   (1.0 percent of time)
Time spent in simplifying the rules =  17.30 sec   (16.0 percent of time)
Time spent in blocking              =   4.35 sec   (4.0 percent of time)
```

```
Total time (including 'undo' action) = 3257.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/12e.eq
Equations read in are:
 1. ((X @ Y) ^ Z) == (X @ (Y ^ Z))  [USER, 24]



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 14]
 [54] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 15]
 [55] I(O) ---> O  [52, 51]
 [60] I(I(Z)) ---> Z  [52, 52]
 [62] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 61]
 [72] (X $ O) ---> O  [56, 52]
 [74] (O $ Z) ---> O  [57, 52]
 [79] (X $ I(Z)) ---> I((X $ Z))  [58, 52]
 [80] (I(Z) $ Z1) ---> I((Z $ Z1))  [59, 52]
 [81] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 16]
 [82] (1 @ X) ---> X  [USER, 17]
 [83] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 18]
 [84] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 19]
 [91] (X @ O) ---> O  [90, 81]
 [99] (0 @ Z) ---> O  [89, 52]
 [106] (-(Y) @ Z) ---> I((Y @ Z))  [105, 81]
 [108] (X @ I(Z)) ---> I((X @ Z))  [82, 107]
 [109] (X ^ (Y * Z)) ---> ((X ^ Y) ^ Z)  [USER, 20]
 [110] (X ^ 1) ---> X  [USER, 21]
 [111] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 22]
 [112] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 23]
 [122] (O ^ Z) ---> O  [121, 114]
 [132] (Z ^ 0) ---> O  [120, 52]
 [143] (X ^ -(Z)) ---> I((X ^ Z))  [12, 141]
 [145] (I(X) ^ Z) ---> I((X ^ Z))  [144, 60]
 [146] (X @ (Y ^ Z)) ---> ((X @ Y) ^ Z)  [USER, 24]

Time used                          = 109.73 sec
```

```
Number of rules generated           = 146
Number of rules retained            = 42
Number of critical pairs            = 871
Number of unblocked critical pairs  = 195
Time spent in normalization         =  56.38 sec  (51.0 percent of time)
Time spent in unification           =  13.97 sec   (12.0 percent of time)
Time spent in ordering              =   1.96 sec   (1.0 percent of time)
Time spent in simplifying the rules =  17.37 sec   (15.0 percent of time)
Time spent in blocking              =   4.38 sec   (3.0 percent of time)
Total time (including 'undo' action) = 201.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/12f.eq
Equations read in are:
 1. ((X @ Y) $ Z) == (X @ (Y $ Z))  [USER, 25]
 2. (X $ (Y ^ Z)) == ((X $ Y) ^ Z)  [USER, 26]
 3. ((X ^ Y) $ Z) == (X $ (Y @ Z))  [USER, 27]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (1 * X) ---> X  [USER, 6]
  [4] (X * 1) ---> X  [USER, 7]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [6] ((X + Y) * Z) ---> ((X * Z) + (Y * Z))  [USER, 9]
  [7] -(0) ---> 0  [2, 1]
 [12] -(-(Z)) ---> Z  [2, 2]
 [25] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 13]
 [32] (Z * 0) ---> 0  [14, 2]
 [33] (0 * Z) ---> 0  [16, 2]
 [48] (X * -(Z)) ---> -((X * Z))  [10, 2]
 [49] (-(Z) * Z1) ---> -((Z * Z1))  [11, 2]
 [50] (X * (Y * Z)) ---> ((X * Y) * Z)  [USER, 5]
 [51] (X # O) ---> X  [USER, 12]
 [52] (X # I(X)) ---> O  [USER, 13]
 [53] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 14]
 [54] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 15]
 [55] I(O) ---> O  [52, 51]
 [60] I(I(Z)) ---> Z  [52, 52]
 [62] I((Z # Z1)) ---> (I(Z) # I(Z1))  [52, 61]
 [72] (X $ O) ---> O  [56, 52]
 [74] (O $ Z) ---> O  [57, 52]
 [79] (X $ I(Z)) ---> I((X $ Z))  [58, 52]
 [80] (I(Z) $ Z1) ---> I((Z $ Z1))  [59, 52]
 [81] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 16]
 [82] (1 @ X) ---> X  [USER, 17]
 [83] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 18]
 [84] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 19]
 [91] (X @ O) ---> O  [90, 81]
 [99] (0 @ Z) ---> O  [89, 52]
 [106] (-(Y) @ Z) ---> I((Y @ Z))  [105, 81]
 [108] (X @ I(Z)) ---> I((X @ Z))  [82, 107]
```

```
 [109] (X ^ (Y * Z)) ---> ((X ^ Y) ^ Z)  [USER, 20]
 [110] (X ^ 1) ---> X  [USER, 21]
 [111] (X ^ (Y + Z)) ---> ((X ^ Y) # (X ^ Z))  [USER, 22]
 [112] ((X # Y) ^ Z) ---> ((X ^ Z) # (Y ^ Z))  [USER, 23]
 [122] (0 ^ Z) ---> 0  [121, 114]
 [132] (Z ^ 0) ---> 0  [120, 52]
 [143] (X ^ -(Z)) ---> I((X ^ Z))  [12, 141]
 [145] (I(X) ^ Z) ---> I((X ^ Z))  [144, 60]
 [146] (X @ (Y ^ Z)) ---> ((X @ Y) ^ Z)  [USER, 24]
 [147] ((X @ Y) $ Z) ---> (X @ (Y $ Z))  [USER, 25]
 [148] (X $ (Y ^ Z)) ---> ((X $ Y) ^ Z)  [USER, 26]
 [149] ((X ^ Y) $ Z) ---> (X $ (Y @ Z))  [USER, 27]

Time used                          = 116.89 sec
Number of rules generated          = 149
Number of rules retained           = 45
Number of critical pairs           = 930
Number of unblocked critical pairs   = 195
Time spent in normalization        =  57.48 sec  (49.0 percent of time)
Time spent in unification          =  14.06 sec   (12.0 percent of time)
Time spent in ordering             =   2.12 sec   (1.0 percent of time)
Time spent in simplifying the rules =  17.58 sec   (15.0 percent of time)
Time spent in blocking             = 129.30 sec   (110.0 percent of time)
Total time (including 'undo' action) = 1047412.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

13 A-algebras

13.1 Axioms

Let x, y, z be arbitrary but fixed variables of type A, where A is an commutative and accociative ring with unit. Let u, v, w be arbitrary but fixed variables of type M, where M is an A-module over a commutative ring.

Definition 13.1 *An A-algebra M is an A-module over a commutative ring together with a new bilinear mapping $\otimes$, where $\otimes : M \times M \rightarrow M$, fulfilling the following equations:*

$$\begin{aligned} u \otimes (v \oplus w) &= (u \otimes v) \oplus (u \otimes w) && (19) \\ (u \oplus v) \otimes w &= (u \otimes w) \oplus (v \otimes w) && (20) \\ u \otimes (x \cdot v) &= x \cdot (u \otimes v) && (21) \\ (x \cdot u) \otimes v &= x \cdot (u \otimes v) && (22) \end{aligned}$$

Definition 13.2 *An associative A-algebra M is an A-algebra M, where $\otimes$ is associative, i.e.:*

$$(u \otimes v) \otimes w = u \otimes (v \otimes w) \tag{23}$$

13.2 Example

A model of an associative A-algebra is the ring of polynomials with one indeterminate over the real numbers. The internal law is the addition of polynomials, the external law is the multiplication of a polynomial with a scalar. Finally the multiplication of polynomials is the remaining operation.

13.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type A and let u, v, w be arbitrary but fixed variables of type M.

There is the following canonical system for A-algebras (+, $*$ and $\oplus$ ac):

$$\begin{aligned} x + 0 &\longrightarrow x && (1) \\ x + -(x) &\longrightarrow 0 && (2) \\ x * 1 &\longrightarrow x && (3) \\ x * (y + z) &\longrightarrow (x * y) + (x * z) && (4) \\ -(0) &\longrightarrow 0 && (5) \\ -(-(z)) &\longrightarrow z && (6) \\ -(x + y) &\longrightarrow -(x) + -(y) && (7) \end{aligned}$$

$$
\begin{aligned}
z * 0 &\longrightarrow 0 &\quad (8)\\
x * -(z) &\longrightarrow -(x * z) &\quad (9)\\
u \oplus O &\longrightarrow u &\quad (10)\\
u \oplus i(u) &\longrightarrow O &\quad (11)\\
i(O) &\longrightarrow O &\quad (12)\\
i(i(u)) &\longrightarrow u &\quad (13)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) &\quad (14)\\
1 \cdot u &\longrightarrow u &\quad (15)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) &\quad (16)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) &\quad (17)\\
0 \cdot u &\longrightarrow O &\quad (18)\\
x \cdot O &\longrightarrow O &\quad (19)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) &\quad (20)\\
-(z) \cdot u &\longrightarrow i(z \cdot u) &\quad (21)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u &\quad (22)\\
u \otimes (v \oplus w) &\longrightarrow (u \otimes v) \oplus (u \otimes w) &\quad (23)\\
(u \oplus v) \otimes w &\longrightarrow (u \otimes w) \oplus (v \otimes w) &\quad (24)\\
u \otimes (y \cdot v) &\longrightarrow y \cdot (u \otimes v) &\quad (25)\\
(x \cdot u) \otimes v &\longrightarrow x \cdot (u \otimes v) &\quad (26)\\
u \otimes O &\longrightarrow O &\quad (27)\\
O \otimes u &\longrightarrow O &\quad (28)\\
u \otimes i(v) &\longrightarrow i(u \otimes v) &\quad (29)\\
i(u) \otimes v &\longrightarrow i(u \otimes v) &\quad (30)\\
& &\quad (31)
\end{aligned}
$$

There is the following canonical system for associative A-algebras (+, $*$, $\oplus$ ac):

$$
\begin{aligned}
x + 0 &\longrightarrow x &\quad (1)\\
x + -(x) &\longrightarrow 0 &\quad (2)\\
x * 1 &\longrightarrow x &\quad (3)\\
x * (y + z) &\longrightarrow (x * y) + (x * z) &\quad (4)\\
-(0) &\longrightarrow 0 &\quad (5)\\
-(-(z)) &\longrightarrow z &\quad (6)\\
-(x + y) &\longrightarrow -(x) + -(y) &\quad (7)\\
z * 0 &\longrightarrow 0 &\quad (8)\\
x * -(z) &\longrightarrow -(x * z) &\quad (9)
\end{aligned}
$$

$$
\begin{aligned}
u \oplus O &\longrightarrow u & (10)\\
u \oplus i(u) &\longrightarrow O & (11)\\
i(O) &\longrightarrow O & (12)\\
i(i(u)) &\longrightarrow u & (13)\\
i(u \oplus v) &\longrightarrow i(u) \oplus i(v) & (14)\\
1 \cdot u &\longrightarrow u & (15)\\
(x + y) \cdot u &\longrightarrow (x \cdot u) \oplus (y \cdot u) & (16)\\
x \cdot (u \oplus v) &\longrightarrow (x \cdot u) \oplus (x \cdot v) & (17)\\
0 \cdot u &\longrightarrow O & (18)\\
x \cdot O &\longrightarrow O & (19)\\
x \cdot i(u) &\longrightarrow i(x \cdot u) & (20)\\
-(z) \cdot u &\longrightarrow i(z \cdot u) & (21)\\
x \cdot (y \cdot u) &\longrightarrow (x * y) \cdot u & (22)\\
u \otimes (v \oplus w) &\longrightarrow (u \otimes v) \oplus (u \otimes w) & (23)\\
(u \oplus v) \otimes w &\longrightarrow (u \otimes w) \oplus (v \otimes w) & (24)\\
u \otimes (y \cdot v) &\longrightarrow y \cdot (u \otimes v) & (25)\\
(x \cdot u) \otimes v &\longrightarrow x \cdot (u \otimes v) & (26)\\
u \otimes O &\longrightarrow O & (27)\\
O \otimes u &\longrightarrow O & (28)\\
u \otimes i(v) &\longrightarrow i(u \otimes v) & (29)\\
i(u) \otimes v &\longrightarrow i(u \otimes v) & (30)\\
u \otimes (v \otimes w) &\longrightarrow (u \otimes v) \otimes w & (31)
\end{aligned}
$$

13.4 Trace

The operation $\cdot$ is represented by @, $\oplus$ by #, $\otimes$ by \$, i by I; $+$, $*$, $-$, 0, 1, O remain the same. The output was edited, missing parts are indicated by "...".

```
RRL-> read test/13a.eq
New constant set is: { 1, 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. ((X + Y) + Z) == (X + (Y + Z))  [USER, 2]
 3. (X + 0) == X  [USER, 3]
 4. (X + -(X)) == 0  [USER, 4]
 5. (X * Y) == (Y * X)  [USER, 5]
 6. ((X * Y) * Z) == (X * (Y * Z))  [USER, 6]
 7. (1 * X) == X  [USER, 7]
 8. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 8]
 9. ((X + Y) * Z) == ((X * Z) + (Y * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
```

```
...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (X * 1) ---> X  [USER, 7]
  [4] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [5] -(0) ---> 0  [2, 1]
  [8] -(-(Z)) ---> Z  [2, 2]
 [14] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 9]
 [20] (Z * 0) ---> 0  [17, 8]
 [49] F1(Z) ---> 0  [18, 2]
 [51] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [50, 50]

Time used                           =  55.95 sec
Number of rules generated           = 51
Number of rules retained            = 10
Number of critical pairs            = 432
Number of unblocked critical pairs  = 182
Time spent in normalization         =  34.79 sec  (62.0 percent of time)
Time spent in unification           =   7.11 sec   (12.0 percent of time)
Time spent in ordering              =   0.56 sec   (1.0 percent of time)
Time spent in simplifying the rules =   2.93 sec   (5.0 percent of time)
Time spent in blocking              =   2.13 sec   (3.0 percent of time)
Total time (including 'undo' action) = 5617.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/13b.eq
New constant set is: { O }
Equations read in are:
 1. (X # Y) == (Y # X)  [USER, 10]
 2. ((X # Y) # Z) == (X # (Y # Z))  [USER, 11]
 3. (X # O) == X  [USER, 12]
 4. (X # I(X)) == O  [USER, 13]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (X * 1) ---> X  [USER, 7]
  [4] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [5] -(0) ---> 0  [2, 1]
  [8] -(-(Z)) ---> Z  [2, 2]
 [14] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 9]
 [20] (Z * 0) ---> 0  [17, 8]
 [49] F1(Z) ---> 0  [18, 2]
 [51] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [50, 50]
 [52] (X # O) ---> X  [USER, 12]
 [53] (X # I(X)) ---> O  [USER, 13]
 [54] I(O) ---> O  [53, 52]
 [55] I(I(Z)) ---> Z  [53, 53]
```

```
 [57] I((Z # Z1)) ---> (I(Z) # I(Z1))  [53, 56]

Time used                           =  58.90 sec
Number of rules generated           = 57
Number of rules retained            = 15
Number of critical pairs            = 483
Number of unblocked critical pairs  = 211
Time spent in normalization         =  35.71 sec  (60.0 percent of time)
Time spent in unification           =   7.87 sec   (13.0 percent of time)
Time spent in ordering              =   0.70 sec   (1.0 percent of time)
Time spent in simplifying the rules =   3.10 sec   (5.0 percent of time)
Time spent in blocking              =   2.32 sec   (3.0 percent of time)
Total time (including 'undo' action) = 306.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/13c.eq
Equations read in are:
 1. (X @ (Y @ Z)) == ((X * Y) @ Z)  [USER, 14]
 2. (1 @ X) == X  [USER, 15]
 3. ((X @ Z) # (Y @ Z)) == ((X + Y) @ Z)  [USER, 16]
 4. (X @ (Y # Z)) == ((X @ Y) # (X @ Z))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (X * 1) ---> X  [USER, 7]
  [4] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [5] -(0) ---> 0  [2, 1]
  [8] -(-(Z)) ---> Z  [2, 2]
 [14] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 9]
 [20] (Z * 0) ---> 0  [17, 8]
 [49] F1(Z) ---> 0  [18, 2]
 [51] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [50, 50]
 [52] (X # O) ---> X  [USER, 12]
 [53] (X # I(X)) ---> O  [USER, 13]
 [54] I(O) ---> O  [53, 52]
 [55] I(I(Z)) ---> Z  [53, 53]
 [57] I((Z # Z1)) ---> (I(Z) # I(Z1))  [53, 56]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [75] (0 @ Z) ---> O  [65, 53]
 [83] (X @ O) ---> O  [61, 53]
 [86] (X @ I(Z)) ---> I((X @ Z))  [63, 53]
 [87] (-(Z) @ Z1) ---> I((Z @ Z1))  [64, 53]
 [88] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]

Time used                           =  82.18 sec
Number of rules generated           = 88
Number of rules retained            = 23
Number of critical pairs            = 699
Number of unblocked critical pairs  = 254
```

```
Time spent in normalization         =  47.96 sec  (58.0 percent of time)
Time spent in unification           =  11.47 sec   (13.0 percent of time)
Time spent in ordering              =   1.27 sec   (1.0 percent of time)
Time spent in simplifying the rules =   5.44 sec   (6.0 percent of time)
Time spent in blocking              =   9.05 sec   (11.0 percent of time)
Total time (including 'undo' action) = 18327.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/13d.eq
Equations read in are:
 1. (X $ (Y # Z)) == ((X $ Y) # (X $ Z))  [USER, 18]
 2. ((X # Y) $ Z) == ((X $ Z) # (Y $ Z))  [USER, 19]
 3. (X $ (Y @ Z)) == (Y @ (X $ Z))  [USER, 20]
 4. ((X @ Y) $ Z) == (X @ (Y $ Z))  [USER, 21]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (X * 1) ---> X  [USER, 7]
  [4] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [5] -(0) ---> 0  [2, 1]
  [8] -(-(Z)) ---> Z  [2, 2]
 [14] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 9]
 [20] (Z * 0) ---> 0  [17, 8]
 [49] F1(Z) ---> 0  [18, 2]
 [51] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [50, 50]
 [52] (X # O) ---> X  [USER, 12]
 [53] (X # I(X)) ---> O  [USER, 13]
 [54] I(O) ---> O  [53, 52]
 [55] I(I(Z)) ---> Z  [53, 53]
 [57] I((Z # Z1)) ---> (I(Z) # I(Z1))  [53, 56]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [75] (0 @ Z) ---> O  [65, 53]
 [83] (X @ O) ---> O  [61, 53]
 [86] (X @ I(Z)) ---> I((X @ Z))  [63, 53]
 [87] (-(Z) @ Z1) ---> I((Z @ Z1))  [64, 53]
 [88] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [89] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 18]
 [90] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 19]
 [91] (X $ (Y @ Z)) ---> (Y @ (X $ Z))  [USER, 20]
 [92] ((X @ Y) $ Z) ---> (X @ (Y $ Z))  [USER, 21]
 [95] (X $ O) ---> O  [75, 91]
 [96] (O $ Z) ---> O  [75, 92]
 [103] (X $ I(Z)) ---> I((X $ Z))  [58, 99]
 [104] (I(Z) $ Z1) ---> I((Z $ Z1))  [58, 101]

Time used                           =  89.91 sec
Number of rules generated           = 104
Number of rules retained            = 31
Number of critical pairs            = 773
```

```
Number of unblocked critical pairs   = 262
Time spent in normalization          =  49.69 sec  (55.0 percent of time)
Time spent in unification            =  11.88 sec   (13.0 percent of time)
Time spent in ordering               =   1.52 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   7.91 sec   (8.0 percent of time)
Time spent in blocking               =   9.19 sec   (10.0 percent of time)
Total time (including ´undo´ action) = 787.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/13e.eq
Equations read in are:
 1. (X $ (Y $ Z)) == ((X $ Y) $ Z)  [USER, 22]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (X + 0) ---> X  [USER, 3]
  [2] (X + -(X)) ---> 0  [USER, 4]
  [3] (X * 1) ---> X  [USER, 7]
  [4] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 8]
  [5] -(0) ---> 0  [2, 1]
  [8] -(-(Z)) ---> Z  [2, 2]
 [14] -((Z + Z1)) ---> (-(Z) + -(Z1))  [2, 9]
 [20] (Z * 0) ---> 0  [17, 8]
 [49] F1(Z) ---> 0  [18, 2]
 [51] (XEX1 * -(Z)) ---> -((XEX1 * Z))  [50, 50]
 [52] (X # O) ---> X  [USER, 12]
 [53] (X # I(X)) ---> O  [USER, 13]
 [54] I(O) ---> O  [53, 52]
 [55] I(I(Z)) ---> Z  [53, 53]
 [57] I((Z # Z1)) ---> (I(Z) # I(Z1))  [53, 56]
 [58] (1 @ X) ---> X  [USER, 15]
 [59] ((X + Y) @ Z) ---> ((X @ Z) # (Y @ Z))  [USER, 16]
 [60] (X @ (Y # Z)) ---> ((X @ Y) # (X @ Z))  [USER, 17]
 [75] (0 @ Z) ---> O  [65, 53]
 [83] (X @ O) ---> O  [61, 53]
 [86] (X @ I(Z)) ---> I((X @ Z))  [63, 53]
 [87] (-(Z) @ Z1) ---> I((Z @ Z1))  [64, 53]
 [88] (X @ (Y @ Z)) ---> ((X * Y) @ Z)  [USER, 14]
 [89] (X $ (Y # Z)) ---> ((X $ Y) # (X $ Z))  [USER, 18]
 [90] ((X # Y) $ Z) ---> ((X $ Z) # (Y $ Z))  [USER, 19]
 [91] (X $ (Y @ Z)) ---> (Y @ (X $ Z))  [USER, 20]
 [92] ((X @ Y) $ Z) ---> (X @ (Y $ Z))  [USER, 21]
 [95] (X $ O) ---> O  [75, 91]
 [96] (O $ Z) ---> O  [75, 92]
 [103] (X $ I(Z)) ---> I((X $ Z))  [58, 99]
 [104] (I(Z) $ Z1) ---> I((Z $ Z1))  [58, 101]
 [105] (X $ (Y $ Z)) ---> ((X $ Y) $ Z)  [USER, 22]

Time used                            =  90.55 sec
Number of rules generated            = 105
Number of rules retained             = 32
Number of critical pairs             = 787
Number of unblocked critical pairs   = 262
```

```
Time spent in normalization         =  49.99 sec  (55.0 percent of time)
Time spent in unification           =  11.91 sec   (13.0 percent of time)
Time spent in ordering              =   1.62 sec   (1.0 percent of time)
Time spent in simplifying the rules =   7.95 sec   (8.0 percent of time)
Time spent in blocking              =   9.22 sec   (10.0 percent of time)
Total time (including ´undo´ action) = 80.0 sec


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

14 Lattices

14.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 14.1 *A lattice L is a set S with two binary operations* $\cup$ *and* $\cap$*, i.e.* $\cup : S \times S \to S$ *and* $\cap : S \times S \to S$ *fulfilling the following axioms:*

$$\begin{aligned} x \cap y &= y \cap x && (1)\\ x \cap (y \cap z) &= (x \cap y) \cap z && (2)\\ x \cup y &= y \cup x && (3)\\ x \cup (y \cup z) &= (x \cup y) \cup z && (4)\\ x \cap (x \cup y) &= x && (5)\\ x \cup (x \cap y) &= x && (6) \end{aligned}$$

14.2 Example

A model of a lattice is the powerset of an arbitrary but fixed finite set M. The operations are intersection and union.

14.3 A canonical System

In [Hul80] there is no canonical system for lattices. The previous test of the program did not terminate.

14.4 Trace

The operation $\cap$ is represented by $*$ and $\cup$ by $+$. The test was only performed a certain amount of time. One can see the two laws of idempotency during the run of the program:

$$\begin{aligned} x \cap x &= x\\ x \cup x &= x \end{aligned}$$

```
RRL-> read test/14.eq
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. (X * Y) == (Y * X)  [USER, 2]
 3. (X + (Y + Z)) == ((X + Y) + Z)  [USER, 3]
 4. (X * (Y * Z)) == ((X * Y) * Z)  [USER, 4]
 5. (X * (X + Y)) == X  [USER, 5]
 6. (X + (X * Y)) == X  [USER, 6]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
```

```
RRL-> option brake runtime 100



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

´+´ has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]

----- Step 3 -----

´*´ has the commutative property now:
    (X * Y) == (Y * X)  [USER, 2]

----- Step 4 -----

´+´ has the associative and commutative property now:
    (X + (Y + Z)) == (Z + (X + Y))  [USER, 3]

----- Step 5 -----

´*´ has the associative and commutative property now:
    (X * (Y * Z)) == (Z * (X * Y))  [USER, 4]

Adding Rule:   [1] (X * (X + Y)) ---> X  [USER, 5]

Adding Rule:   [2] (X + (X * Y)) ---> X  [USER, 6]

Adding Rule:   [3] (Y * Z * Z) ---> (Y * Z)  [2, 1]

Adding Rule:   [4] (Z * Z) ---> Z  [2, 1]

  Deleting rule:  [3] (Y * Z * Z) ---> (Y * Z)  [2, 1]

Adding Rule:   [5] (Y + Z + Z) ---> (Y + Z)  [1, 2]

Adding Rule:   [6] (Z + Z) ---> Z  [1, 2]

  Deleting rule:  [5] (Y + Z + Z) ---> (Y + Z)  [1, 2]

WARNING: Run time limit (100 seconds) has been reached.
Continue with a larger limit? (Y,N,YES,NO) no
```

15 Distributive Lattices

15.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 15.1 *A distributive lattice DL is a lattice with the additional axiom:*

$$x \cup (y \cap z) = (x \cup y) \cap (x \cup z) \tag{7}$$

15.2 Example

As a model of distributive lattice we can just take the same as in the previous chapter.

15.3 A canonical System

Let x, y, z be arbitrary but fixed variables of type S.

There is the following canonical system for distributive lattices ($\cap$ and $\cup$ ac):

$$\begin{aligned} x \cap (x \cup y) &\longrightarrow x && (1)\\ x \cup (y \cap z) &\longrightarrow (x \cup y) \cap (x \cup z) && (2)\\ x \cup x &\longrightarrow x && (3)\\ z \cap z &\longrightarrow z && (4) \end{aligned}$$

15.4 Trace

The operation $\cap$ is represented by $*$ and $\cup$ by $+$. Finally the other distributive law is proved, i.e.:

$$x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$$

```
RRL-> read test/15.eq
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. (X * Y) == (Y * X)  [USER, 2]
 3. (X + (Y + Z)) == ((X + Y) + Z)  [USER, 3]
 4. (X * (Y * Z)) == ((X * Y) * Z)  [USER, 4]
 5. (X * (X + Y)) == X  [USER, 5]
 6. (X + (X * Y)) == X  [USER, 6]
 7. (X + (Y * Z)) == ((X + Y) * (X + Z))  [USER, 7]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

'+' has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]
```

```
----- Step 3 -----

´*´ has the commutative property now:
    (X * Y) == (Y * X)  [USER, 2]

----- Step 4 -----

´+´ has the associative and commutative property now:
    (X + (Y + Z)) == (Z + (X + Y))  [USER, 3]

----- Step 5 -----

´*´ has the associative and commutative property now:
    (X * (Y * Z)) == (Z * (X * Y))  [USER, 4]

Adding Rule:   [1] (X * (X + Y)) ---> X  [USER, 5]

Adding Rule:   [2] (X + (X * Y)) ---> X  [USER, 6]

Trying to orient equation:
  (X + (Y * Z)) == ((X + Y) * (X + Z))  [USER, 7]

  To prove:  (X + (Y * Z))  >  ((X + Y) * (X + Z))
        or:  (X + (Y * Z))  <  ((X + Y) * (X + Z))
  Here are some precedence suggestions:
       1.  + > *      2.  * > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, + > *, is added.

----- Step 6 -----

Adding Rule:   [3] (X + (Y * Z)) ---> ((X + Y) * (X + Z))  [USER, 7]

  Deleting rule:  [2] (X + (X * Y)) ---> X  [USER, 6]

Adding Rule:   [4] ((X + X) * (X + Y)) ---> X  [DELETED, 2]

Adding Rule:   [5] (X + X) ---> X  [4, 1]

  Deleting rule:  [4] ((X + X) * (X + Y)) ---> X  [DELETED, 2]

Adding Rule:   [6] (Z * Z) ---> Z  [5, 1]

Your system is canonical.

  [1] (X * (X + Y)) ---> X  [USER, 5]
  [3] (X + (Y * Z)) ---> ((X + Y) * (X + Z))  [USER, 7]
  [5] (X + X) ---> X  [4, 1]
  [6] (Z * Z) ---> Z  [5, 1]

Time used                           =  35.15 sec
Number of rules generated           = 6
Number of rules retained            = 4
Number of critical pairs            = 87
Number of unblocked critical pairs  = 51
Time spent in normalization         =  31.96 sec  (90.0 percent of time)
Time spent in unification           =   1.13 sec   (3.0 percent of time)
Time spent in ordering              =   0.13 sec   (0.0 percent of time)
Time spent in simplifying the rules =   0.15 sec   (0.0 percent of time)
Time spent in blocking              =   0.39 sec   (1.0 percent of time)
Total time (including ´undo´ action) = 3517.0 sec
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> prove
Type your equation in the format:  L == R (if C)
Enter a ´]´ to exit when no equation is given.

x * (y + z) == (x * y) + (x * z)

Yes, it is equational theorem.


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

16 Boolean Algebras

16.1 Axioms

Let x, y, z be arbitrary but fixed variables of type S.

Definition 16.1 *A Boolean algebra BA is a distributive lattice fulfilling the additional axioms:*

$$x \cap 1 = x \tag{8}$$
$$x \cap \overline{x} = 0 \tag{9}$$
$$x \cup 0 = x \tag{10}$$
$$x \cup \overline{x} = 1 \tag{11}$$

16.2 Example

A model of a Boolean algebra is propositional logic with conjunction, disjunction and negation. 0 and 1 are represented as false and true, resp.

16.3 A canonical System

In [Hul80] there is no canonical system for Boolean algebra. Although RRL terminates with a positive report obviously the added equations totally disappeared, since in the canonical system the constants 0 and 1 are missing. Thus the result is not correct.

16.4 Trace

The operation $\cap$ is represented by $*$, $\cup$ by $+$ and $^{-}$ by $-$.

```
RRL-> read test/16a.eq
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. (X * Y) == (Y * X)  [USER, 2]
 3. (X + (Y + Z)) == ((X + Y) + Z)  [USER, 3]
 4. (X * (Y * Z)) == ((X * Y) * Z)  [USER, 4]
 5. (X * (X + Y)) == X  [USER, 5]
 6. (X + (X * Y)) == X  [USER, 6]
 7. (X + (Y * Z)) == ((X + Y) * (X + Z))  [USER, 7]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

----- Step 2 -----

'+' has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]

----- Step 3 -----
```

```
´*´ has the commutative property now:
    (X * Y) == (Y * X)  [USER, 2]

----- Step 4 -----

´+´ has the associative and commutative property now:
    (X + (Y + Z)) == (Z + (X + Y))  [USER, 3]

----- Step 5 -----

´*´ has the associative and commutative property now:
    (X * (Y * Z)) == (Z * (X * Y))  [USER, 4]

Adding Rule:   [1] (X * (X + Y)) ---> X  [USER, 5]

Adding Rule:   [2] (X + (X * Y)) ---> X  [USER, 6]

Trying to orient equation:
  (X + (Y * Z)) == ((X + Y) * (X + Z))  [USER, 7]

  To prove:  (X + (Y * Z))  >  ((X + Y) * (X + Z))
        or:  (X + (Y * Z))  <  ((X + Y) * (X + Z))
  Here are some precedence suggestions:
       1.  + > *      2.  * > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, + > *, is added.

----- Step 6 -----

Adding Rule:   [3] (X + (Y * Z)) ---> ((X + Y) * (X + Z))  [USER, 7]

  Deleting rule:  [2] (X + (X * Y)) ---> X  [USER, 6]

Adding Rule:   [4] ((X + X) * (X + Y)) ---> X  [DELETED, 2]

Adding Rule:   [5] (X + X) ---> X  [4, 1]

  Deleting rule:  [4] ((X + X) * (X + Y)) ---> X  [DELETED, 2]

Adding Rule:   [6] (Z * Z) ---> Z  [5, 1]

Your system is canonical.

  [1] (X * (X + Y)) ---> X  [USER, 5]
  [3] (X + (Y * Z)) ---> ((X + Y) * (X + Z))  [USER, 7]
  [5] (X + X) ---> X  [4, 1]
  [6] (Z * Z) ---> Z  [5, 1]

Time used                          =  35.16 sec
Number of rules generated          = 6
Number of rules retained           = 4
Number of critical pairs           = 87
Number of unblocked critical pairs  = 51
Time spent in normalization         =  31.90 sec  (90.0 percent of time)
Time spent in unification           =   1.14 sec   (3.0 percent of time)
Time spent in ordering              =   0.15 sec   (0.0 percent of time)
Time spent in simplifying the rules  =   0.16 sec   (0.0 percent of time)
Time spent in blocking              =   0.43 sec   (1.0 percent of time)
Total time (including ´undo´ action) = 3517.0 sec
```

```
Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/16b.eq
New constant set is: { 1, 0 }
Equations read in are:
 1. (X + 0) == X  [USER, 8]
 2. (X + -(X)) == 1  [USER, 9]
 3. (X * 1) == X  [USER, 10]
 4. (X * -(X)) == 0  [USER, 11]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 7 -----

Adding Rule:   [7] (X + 0) ---> X  [USER, 8]

Adding Rule:   [8] (X + -(X)) ---> 1  [USER, 9]

Adding Rule:   [9] (X * 1) ---> X  [USER, 10]

Adding Rule:  [10] (X * -(X)) ---> 0  [USER, 11]

Adding Rule:  [11] -(0) ---> 1  [8, 7]

Adding Rule:  [12] -(1) ---> 0  [10, 9]

Adding Rule:  [13] (Z * 0) ---> 0  [7, 1]

Adding Rule:  [14] (Y + 1) ---> 1  [9, 1]

Adding Rule:  [15] ((Y + -((Y * Z))) * (Z + -((Y * Z)))) ---> 1  [8, 3]

Adding Rule:  [16] ((X + Z) * (X + -(Z))) ---> X  [10, 3]

Adding Rule:  [17] ((Y + Z + -((Y * Z1))) * (Z1 + Z + -((Y * Z1)))) ---> 1  [8, 3]

Adding Rule:  [18] (Z * -((Y + Z))) ---> 0  [10, 1]

Adding Rule:  [19] ((X + Z) * (X + -((Y + Z)))) ---> X  [18, 3]

Adding Rule:  [20] (Z * -(((Y + Z) * (Z1 + Z)))) ---> 0  [3, 18]

Adding Rule:  [21] (Y * Z * -(((Y + Z1) * (Z + Z1)))) ---> 0  [3, 18]

Adding Rule:  [22] (Z + -(-(Z))) ---> Z  [8, 16]

Adding Rule:  [23] -(-(Z)) ---> Z  [8, 16]

  Deleting rule: [22] (Z + -(-(Z))) ---> Z  [8, 16]

Adding Rule:  [24] ((Y + Z) * (Z1 + Z) * (Z + -((Y * Z1)))) ---> Z  [3, 16]

Adding Rule:  [25] (-(Z) + -((Z1 + Z))) ---> -(Z)  [8, 16]

Adding Rule:  [26] (Z + -((Z1 + -(Z)))) ---> Z  [8, 16]

Adding Rule:  [27] (-(Z) * -((Z1 + Z))) ---> -((Z1 + Z))  [25, 1]

Adding Rule:  [28] (Z * -((Z1 + -(Z)))) ---> -((Z1 + -(Z)))  [23, 27]
```

```
Adding Rule:  [29] (-(Z) + -(((Y + Z) * (Z1 + Z)))) ---> -(Z)  [3, 25]

Adding Rule:  [30] (-((Y * Z)) + -(((Y + Z1) * (Z + Z1)))) ---> -((Y * Z))  [3, 25]

Adding Rule:  [31] ((Y + -((Z + -((Y * Z1))))) * (Z1 + -((Z + -((Y * Z1)))))) --->
                   (Y * Z1)  [26, 3]

Adding Rule:  [32] (Z + -(((Y + -(Z)) * (Z1 + -(Z))))) ---> Z  [3, 26]

Adding Rule:  [33] ((X + -(Z)) * (X + -((Z1 + Z)))) ---> (X + -((Z1 + Z)))  [27, 3]

Adding Rule:  [34] (-(Z) * -(((Y + Z) * (Z1 + Z)))) ---> -(((Y + Z) * (Z1 + Z)))  [3, 27]

Adding Rule:  [35] (-((Y * Z)) * -(((Y + Z1) * (Z + Z1)))) --->
                   -(((Y + Z1) * (Z + Z1)))  [3, 27]

Adding Rule:  [36] ((X + Z) * (X + -((Z1 + -(Z))))) ---> (X + -((Z1 + -(Z))))  [28, 3]

Adding Rule:  [37] (Z * -(((Y + -(Z)) * (Z1 + -(Z))))) --->
                   -(((Y + -(Z)) * (Z1 + -(Z))))  [3, 28]

Adding Rule:  [38] (Z * -((Z1 + -((Z2 * Z))))) ---> -((Z1 + -((Z2 * Z))))  [28, 6]

Adding Rule:  [39] (-((Z + Z1)) * (Z2 + -(Z1))) ---> -((Z + Z1))  [25, 1]

Adding Rule:  [40] (-((Z + -(Z1))) * (Z1 + Z2)) ---> -((Z + -(Z1)))  [23, 39]

Adding Rule:  [41] ((Z + -((Z1 + -(Z2)))) * (Z3 + Z + Z2)) ---> (Z + -((Z1 + -(Z2))))  [26, 1]

Adding Rule:  [42] ((Y + Z + -((Z1 + -((Y * Z2))))) * (Z2 + Z + -((Z1 + -((Y * Z2)))))) --->
                   ((Y + Z) * (Z2 + Z))  [26, 3]

Adding Rule:  [43] (Z * -((Z * Z1)) * -((Z2 + -(Z1)))) ---> 0  [28, 10]

Adding Rule:  [44] (-((Z + Z1)) * -((Z2 + -(Z1)))) ---> 0  [25, 18]

Adding Rule:  [45] (-((Z + -(Z1))) * (Z + -((Z2 + Z1)))) ---> 0  [25, 18]

Adding Rule:  [46] (-((Z + Z1)) * (Z + -((Z2 + -(Z1))))) ---> 0  [26, 18]

Adding Rule:  [47] -((Z + -((Z1 * Z)))) ---> 0  [18, 28]

Adding Rule:  [48] -((Z + Z1 + -((Z2 * Z1)))) ---> 0  [18, 28]

Adding Rule:  [49] (Z + -((Z1 * Z))) ---> 1  [47, 23]

  Deleting rule: [15] ((Y + -((Y * Z))) * (Z + -((Y * Z)))) ---> 1  [8, 3]

  Deleting rule: [17] ((Y + Z + -((Y * Z1))) * (Z1 + Z + -((Y * Z1)))) ---> 1  [8, 3]

  Deleting rule: [47] -((Z + -((Z1 * Z)))) ---> 0  [18, 28]

  Deleting rule: [48] -((Z + Z1 + -((Z2 * Z1)))) ---> 0  [18, 28]

Adding Rule:  [50] ((X + Z) * (X + -((Z1 + -((Z2 * Z)))))) --->
                   (X + -((Z1 + -((Z2 * Z)))))  [38, 3]

Adding Rule:  [51] (Z * -(((Y + -((Z1 * Z))) * (Z2 + -((Z1 * Z)))))) --->
                   -(((Y + -((Z1 * Z))) * (Z2 + -((Z1 * Z)))))  [3, 38]

Adding Rule:  [52] ((X + -((Z + Z1))) * (X + Z2 + -(Z1))) ---> (X + -((Z + Z1)))  [39, 3]

Adding Rule:  [53] (-(((Y + Z) * (Z1 + Z))) * (Z2 + -(Z))) --->
                   -(((Y + Z) * (Z1 + Z)))  [3, 39]
```

```
Adding Rule:  [54] (-(((Y + Z) * (Z1 + Z))) * (Z2 + -((Y * Z1)))) --->
                   -(((Y + Z) * (Z1 + Z)))  [3, 39]

Adding Rule:  [55] (-(((Y + -(Z)) * (Z1 + -(Z)))) * (Z + Z2)) --->
                   -(((Y + -(Z)) * (Z1 + -(Z))))  [3, 40]

Adding Rule:  [56] (-((Z + -((Y * Z1)))) * (Y + Z2) * (Z1 + Z2)) --->
                   -((Z + -((Y * Z1))))  [3, 40]

Adding Rule:  [57] (-(Z) + -((Z1 * Z))) ---> -((Z1 * Z))  [49, 16]

Adding Rule:  [58] (Z + -((Z1 * -(Z)))) ---> -((Z1 * -(Z)))  [49, 16]

Adding Rule:  [59] (-(Z) * -((Z1 * Z))) ---> -(Z)  [57, 1]

Adding Rule:  [60] (Z * -((Z1 * -(Z)))) ---> Z  [23, 59]

Adding Rule:  [61] ((Y + -((Z * -((Y * Z1))))) * (Z1 + -((Z * -((Y * Z1)))))) --->
                   -((Z * -((Y * Z1))))  [58, 3]

Adding Rule:  [62] (Z + -((Z1 * -((Z2 + Z))))) ---> -((Z1 * -((Z2 + Z))))  [58, 5]

WARNING: More than 50 new rules generated without any change of the precedence.I undo it to last interaction.
----- Return to Step 6 -----

Adding Rule:   [7] (X + 0) ---> X  [USER, 8]

Adding Rule:   [8] (X + -(X)) ---> 1  [USER, 9]

Adding Rule:   [9] (X * 1) ---> X  [USER, 10]

Adding Rule:  [10] (X * -(X)) ---> 0  [USER, 11]

Adding Rule:  [11] -(0) ---> 1  [8, 7]

Adding Rule:  [12] -(1) ---> 0  [10, 9]

Adding Rule:  [13] (Z * 0) ---> 0  [7, 1]

Adding Rule:  [14] (Y + 1) ---> 1  [9, 1]

Adding Rule:  [15] ((Y + -((Y * Z))) * (Z + -((Y * Z)))) ---> 1  [8, 3]

Adding Rule:  [16] ((X + Z) * (X + -(Z))) ---> X  [10, 3]

Adding Rule:  [17] ((Y + Z + -((Y * Z1))) * (Z1 + Z + -((Y * Z1)))) ---> 1  [8, 3]

Adding Rule:  [18] (Z * -((Y + Z))) ---> 0  [10, 1]

Adding Rule:  [19] ((X + Z) * (X + -((Y + Z)))) ---> X  [18, 3]

Adding Rule:  [20] (Z * -(((Y + Z) * (Z1 + Z)))) ---> 0  [3, 18]

Adding Rule:  [21] (Y * Z * -(((Y + Z1) * (Z + Z1)))) ---> 0  [3, 18]

Adding Rule:  [22] (Z + -(-(Z))) ---> Z  [8, 16]

Adding Rule:  [23] -(-(Z)) ---> Z  [8, 16]

  Deleting rule: [22] (Z + -(-(Z))) ---> Z  [8, 16]

Adding Rule:  [24] ((Y + Z) * (Z1 + Z) * (Z + -((Y * Z1)))) ---> Z  [3, 16]

Adding Rule:  [25] (-(Z) + -((Z1 + Z))) ---> -(Z)  [8, 16]
```

```
Adding Rule:  [26] (Z + -((Z1 + -(Z)))) ---> Z  [8, 16]

Adding Rule:  [27] (-(Z) * -((Z1 + Z))) ---> -((Z1 + Z))  [25, 1]

Adding Rule:  [28] (Z * -((Z1 + -(Z)))) ---> -((Z1 + -(Z)))  [23, 27]

Adding Rule:  [29] (-(Z) + -(((Y + Z) * (Z1 + Z)))) ---> -(Z)  [3, 25]

Adding Rule:  [30] (-((Y * Z)) + -(((Y + Z1) * (Z + Z1)))) ---> -((Y * Z))  [3, 25]

Adding Rule:  [31] ((Y + -((Z + -((Y * Z1))))) * (Z1 + -((Z + -((Y * Z1)))))) --->
                   (Y * Z1)  [26, 3]

Adding Rule:  [32] (Z + -(((Y + -(Z)) * (Z1 + -(Z))))) ---> Z  [3, 26]

Adding Rule:  [33] ((X + -(Z)) * (X + -((Z1 + Z)))) ---> (X + -((Z1 + Z)))  [27, 3]

Adding Rule:  [34] (-(Z) * -(((Y + Z) * (Z1 + Z)))) ---> -(((Y + Z) * (Z1 + Z)))  [3, 27]

Adding Rule:  [35] (-((Y * Z)) * -(((Y + Z1) * (Z + Z1)))) --->
                   -(((Y + Z1) * (Z + Z1)))  [3, 27]

Adding Rule:  [36] ((X + Z) * (X + -((Z1 + -(Z))))) ---> (X + -((Z1 + -(Z))))  [28, 3]

Adding Rule:  [37] (Z * -(((Y + -(Z)) * (Z1 + -(Z))))) --->
                   -(((Y + -(Z)) * (Z1 + -(Z))))  [3, 28]

Adding Rule:  [38] (Z * -((Z1 + -((Z2 * Z))))) ---> -((Z1 + -((Z2 * Z))))  [28, 6]

Adding Rule:  [39] (-((Z + Z1)) * (Z2 + -(Z1))) ---> -((Z + Z1))  [25, 1]

Adding Rule:  [40] (-((Z + -(Z1))) * (Z1 + Z2)) ---> -((Z + -(Z1)))  [23, 39]

Adding Rule:  [41] ((Z + -((Z1 + -(Z2)))) * (Z3 + Z + Z2)) ---> (Z + -((Z1 + -(Z2))))  [26, 1]

Adding Rule:  [42] ((Y + Z + -((Z1 + -((Y * Z2))))) * (Z2 + Z + -((Z1 + -((Y * Z2)))))) --->
                   ((Y + Z) * (Z2 + Z))  [26, 3]

Adding Rule:  [43] (Z * -((Z * Z1)) * -((Z2 + -(Z1)))) ---> 0  [28, 10]

Adding Rule:  [44] (-((Z + Z1)) * -((Z2 + -(Z1)))) ---> 0  [25, 18]

Adding Rule:  [45] (-((Z + -(Z1))) * (Z + -((Z2 + Z1)))) ---> 0  [25, 18]

Adding Rule:  [46] (-((Z + Z1)) * (Z + -((Z2 + -(Z1))))) ---> 0  [26, 18]

Adding Rule:  [47] -((Z + -((Z1 * Z)))) ---> 0  [18, 28]

Adding Rule:  [48] -((Z + Z1 + -((Z2 * Z1)))) ---> 0  [18, 28]

Adding Rule:  [49] (Z + -((Z1 * Z))) ---> 1  [47, 23]

  Deleting rule: [15] ((Y + -((Y * Z))) * (Z + -((Y * Z)))) ---> 1  [8, 3]

  Deleting rule: [17] ((Y + Z + -((Y * Z1))) * (Z1 + Z + -((Y * Z1)))) ---> 1  [8, 3]

  Deleting rule: [47] -((Z + -((Z1 * Z)))) ---> 0  [18, 28]

  Deleting rule: [48] -((Z + Z1 + -((Z2 * Z1)))) ---> 0  [18, 28]

Adding Rule:  [50] ((X + Z) * (X + -((Z1 + -((Z2 * Z)))))) --->
                   (X + -((Z1 + -((Z2 * Z)))))  [38, 3]

Adding Rule:  [51] (Z * -(((Y + -((Z1 * Z))) * (Z2 + -((Z1 * Z)))))) --->
                   -(((Y + -((Z1 * Z))) * (Z2 + -((Z1 * Z)))))  [3, 38]
```

```
Adding Rule:  [52] ((X + -((Z + Z1))) * (X + Z2 + -(Z1))) ---> (X + -((Z + Z1)))  [39, 3]

Adding Rule:  [53] (-(((Y + Z) * (Z1 + Z))) * (Z2 + -(Z))) --->
                   -(((Y + Z) * (Z1 + Z)))  [3, 39]

Adding Rule:  [54] (-(((Y + Z) * (Z1 + Z))) * (Z2 + -((Y * Z1)))) --->
                   -(((Y + Z) * (Z1 + Z)))  [3, 39]

Adding Rule:  [55] (-(((Y + -(Z)) * (Z1 + -(Z)))) * (Z + Z2)) --->
                   -(((Y + -(Z)) * (Z1 + -(Z))))  [3, 40]

Adding Rule:  [56] (-((Z + -((Y * Z1)))) * (Y + Z2) * (Z1 + Z2)) --->
                   -((Z + -((Y * Z1))))  [3, 40]

Adding Rule:  [57] (-(Z) + -((Z1 * Z))) ---> -((Z1 * Z))  [49, 16]

Adding Rule:  [58] (Z + -((Z1 * -(Z)))) ---> -((Z1 * -(Z)))  [49, 16]

Adding Rule:  [59] (-(Z) * -((Z1 * Z))) ---> -(Z)  [57, 1]

Adding Rule:  [60] (Z * -((Z1 * -(Z)))) ---> Z  [23, 59]

Adding Rule:  [61] ((Y + -((Z * -((Y * Z1))))) * (Z1 + -((Z * -((Y * Z1)))))) --->
                   -((Z * -((Y * Z1))))  [58, 3]

Adding Rule:  [62] (Z + -((Z1 * -((Z2 + Z))))) ---> -((Z1 * -((Z2 + Z))))  [58, 5]

WARNING: More than 50 new rules generated without any change of the precedence.I undo it to last interaction.
----- Return to Step 5 -----

Trying to orient equation:
  (X + (Y * Z)) == ((X + Y) * (X + Z))  [USER, 7]

  To prove:  (X + (Y * Z))  >  ((X + Y) * (X + Z))
        or:  (X + (Y * Z))  <  ((X + Y) * (X + Z))
  Here are some precedence suggestions:
       1.  + > *      2.  * > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, * > +, is added.

----- Step 6 -----

Adding Rule:   [3] ((X + Y) * (X + Z)) ---> (X + (Y * Z))  [USER, 7]

Adding Rule:   [4] (Y * Z * Z) ---> (Y * Z)  [2, 1]

Adding Rule:   [5] (Z * Z) ---> Z  [2, 1]

  Deleting rule:  [4] (Y * Z * Z) ---> (Y * Z)  [2, 1]

Adding Rule:   [6] (Y + Z + Z) ---> (Y + Z)  [1, 2]

Adding Rule:   [7] (Z + Z) ---> Z  [1, 2]

  Deleting rule:  [6] (Y + Z + Z) ---> (Y + Z)  [1, 2]

Adding Rule:   [8] ((Z * Z1) + (Z * (Y + Z1))) ---> (Z * (Y + Z1))  [1, 2]

Adding Rule:   [9] ((Z * (Y + Z1)) + (Z2 * Z * Z1)) ---> (Z * (Y + Z1))  [1, 2]

Adding Rule:  [10] (Z + (Z1 * (Z2 + Z))) ---> (Z + (Z1 * Z2))  [7, 3]
```

```
Adding Rule:  [11] (Z * (Z1 + (Y * Z))) ---> ((Y * Z) + (Z1 * Z))  [2, 3]

Adding Rule:  [12] (Z * (Z1 + Z2)) ---> ((Z * Z1) + (Z * Z2))  [10, 1]

  Deleting rule:  [1] (X * (X + Y)) ---> X  [USER, 5]

  Deleting rule:  [3] ((X + Y) * (X + Z)) ---> (X + (Y * Z))  [USER, 7]

  Deleting rule:  [8] ((Z * Z1) + (Z * (Y + Z1))) ---> (Z * (Y + Z1))  [1, 2]

  Deleting rule:  [9] ((Z * (Y + Z1)) + (Z2 * Z * Z1)) ---> (Z * (Y + Z1))  [1, 2]

  Deleting rule: [10] (Z + (Z1 * (Z2 + Z))) ---> (Z + (Z1 * Z2))  [7, 3]

  Deleting rule: [11] (Z * (Z1 + (Y * Z))) ---> ((Y * Z) + (Z1 * Z))  [2, 3]

Your system is canonical.

  [2] (X + (X * Y)) ---> X  [USER, 6]
  [5] (Z * Z) ---> Z  [2, 1]
  [7] (Z + Z) ---> Z  [1, 2]
 [12] (Z * (Z1 + Z2)) ---> ((Z * Z1) + (Z * Z2))  [10, 1]

Time used                          =  44.72 sec
Number of rules generated          = 12
Number of rules retained           = 4
Number of critical pairs           = 212
Number of unblocked critical pairs  = 122
Time spent in normalization         =  33.40 sec  (74.0 percent of time)
Time spent in unification           =   3.79 sec   (8.0 percent of time)
Time spent in ordering              =   0.30 sec   (0.0 percent of time)
Time spent in simplifying the rules  =   1.53 sec   (3.0 percent of time)
Time spent in blocking              =  43.71 sec   (97.0 percent of time)
Total time (including 'undo' action) = 41037.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

17 R. Milner's Theory of Nondeterm. Machines

17.1 Axioms

Definition 17.1 *The following equational theory was considered by R. Milner for axiomatizing the behavior of finite nondeterministic machines:*

$$\begin{aligned}
x + x &= x && (1)\\
x + 0 &= x && (2)\\
T(x) + x &= T(x) && (3)\\
T(x + T(y)) &= T(x + y) + T(y) && (4)\\
L(x + T(y)) &= L(x + y) + L(y) && (5)\\
(x + y) + z &= x + (y + z) && (6)\\
x + y &= y + x && (7)
\end{aligned}$$

17.2 Example

See the reference from [Hul80].

17.3 A canonical System

There is the following canonical system (+ ac):

$$\begin{aligned}
x + x &\longrightarrow x && (1)\\
x + 0 &\longrightarrow x && (2)\\
x + T(x) &\longrightarrow T(x) && (3)\\
T(x + T(y)) &\longrightarrow T(y) + T(x + y) && (4)\\
L(x + T(y)) &\longrightarrow L(y) + L(x + y) && (5)\\
T(T(y)) &\longrightarrow T(y) && (6)\\
L(T(y)) &\longrightarrow L(y) && (7)\\
z + T(y + z) &\longrightarrow T(y + z) && (8)
\end{aligned}$$

17.4 Trace

The operation L, T and + remain the same, also 0:

```
RRL-> read test/17.eq
New constant set is: { 0 }
Equations read in are:
 1. (X + X) == X  [USER, 1]
 2. (X + 0) == X  [USER, 2]
 3. (T(X) + X) == T(X)  [USER, 3]
 4. T((X + T(Y))) == (T((X + Y)) + T(Y))  [USER, 4]
 5. L((X + T(Y))) == (L((X + Y)) + L(Y))  [USER, 5]
 6. ((X + Y) + Z) == (X + (Y + Z))  [USER, 6]
```

```
 7. (X + Y) == (Y + X)  [USER, 7]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 1 -----

Adding Rule:   [1] (X + X) ---> X  [USER, 1]

Adding Rule:   [2] (X + 0) ---> X  [USER, 2]

Adding Rule:   [3] (T(X) + X) ---> T(X)  [USER, 3]

Trying to orient equation:
  T((X + T(Y))) == (T((X + Y)) + T(Y))  [USER, 4]

  To prove:  T((X + T(Y)))  >  (T((X + Y)) + T(Y))
        or:  T((X + T(Y)))  <  (T((X + Y)) + T(Y))
  Here are some precedence suggestions:
       1.  + > T      2.  T > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, + = T, is added.

----- Step 2 -----

Adding Rule:   [4] (T((X + Y)) + T(Y)) ---> T((X + T(Y)))  [USER, 4]

Trying to orient equation:
  L((X + T(Y))) == (L((X + Y)) + L(Y))  [USER, 5]

  To prove:  L((X + T(Y)))  >  (L((X + Y)) + L(Y))
        or:  L((X + T(Y)))  <  (L((X + Y)) + L(Y))
  Here are some precedence suggestions:
       1.  L > +      2.  + > L
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
Precedence relation, L = +, is added.

----- Step 3 -----

Adding Rule:   [5] (L((X + Y)) + L(Y)) ---> L((X + T(Y)))  [USER, 5]

Trying to orient equation:
  ((X + Y) + Z) == (X + (Y + Z))  [USER, 6]

  To prove:  ((X + Y) + Z)  >  (X + (Y + Z))
        or:  ((X + Y) + Z)  <  (X + (Y + Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> auto
The equation is postponed.

----- Step 4 -----

----- Step 5 -----
```

```
´+´ has the associative and commutative property now:
    (X + Y) == (Y + X)  [USER, 7]

  The LHS of Rule contains ´+´, which is AC now.
  [1] (X + X) ---> X  [USER, 1]

  The LHS of Rule contains ´+´, which is AC now.
  [2] (X + 0) ---> X  [USER, 2]

  The LHS of Rule contains ´+´, which is AC now.
  [3] (T(X) + X) ---> T(X)  [USER, 3]

  The LHS of Rule contains ´+´, which is AC now.
  [4] (T((X + Y)) + T(Y)) ---> T((X + T(Y)))  [USER, 4]

  The LHS of Rule contains ´+´, which is AC now.
  [5] (L((X + Y)) + L(Y)) ---> L((X + T(Y)))  [USER, 5]

Adding Rule:   [6] (X + X) ---> X  [DELETED, 1]

Adding Rule:   [7] (X + 0) ---> X  [DELETED, 2]

Adding Rule:   [8] (X + T(X)) ---> T(X)  [DELETED, 3]

Adding Rule:   [9] T((X + T(Y))) ---> (T(Y) + T((X + Y)))  [DELETED, 4]

Adding Rule:  [10] L((X + T(Y))) ---> (L(Y) + L((X + Y)))  [DELETED, 5]

Adding Rule:  [11] T(T(Y)) ---> T(Y)  [6, 9]

Adding Rule:  [12] L(T(Y)) ---> L(Y)  [6, 10]

Adding Rule:  [13] (Z + T((Z1 + Z))) ---> T((Z1 + Z))  [8, 6]

Processing all postponed equations ...

Your system is canonical.

  [6] (X + X) ---> X  [DELETED, 1]
  [7] (X + 0) ---> X  [DELETED, 2]
  [8] (X + T(X)) ---> T(X)  [DELETED, 3]
  [9] T((X + T(Y))) ---> (T(Y) + T((X + Y)))  [DELETED, 4]
 [10] L((X + T(Y))) ---> (L(Y) + L((X + Y)))  [DELETED, 5]
 [11] T(T(Y)) ---> T(Y)  [6, 9]
 [12] L(T(Y)) ---> L(Y)  [6, 10]
 [13] (Z + T((Z1 + Z))) ---> T((Z1 + Z))  [8, 6]

Time used                           =  14.78 sec
Number of rules generated           = 13
Number of rules retained            = 8
Number of critical pairs            = 127
Number of unblocked critical pairs  = 83
Time spent in normalization         =   9.03 sec  (61.0 percent of time)
Time spent in unification           =   1.90 sec   (12.0 percent of time)
Time spent in ordering              =   0.28 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   0.15 sec   (1.0 percent of time)
Time spent in blocking              =   1.78 sec   (12.0 percent of time)
Total time (including ´undo´ action) = 1484.0 sec


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

18 Arithmetic Theories

18.1 Axioms

Definition 18.1 *The following equational theory was considered by P. Degano and F. Sirovich for solving the equivalence problem for a subclass of primitive recursive functions:*

$$\begin{aligned}
x + y &= y + x && (1)\\
x + (y + z) &= (x + y) + z && (2)\\
x * y &= y * x && (3)\\
x * (y * z) &= (x * y) * z && (4)\\
0 + x &= x && (5)\\
0 * x &= 0 && (6)\\
S(x) + y &= S(x + y) && (7)\\
S(x) * y &= (x * y) + y && (8)\\
x * (y + z) &= (x * y) + (x * z) && (9)
\end{aligned}$$

Definition 18.2 *Stickel added the following axioms for exponentiation:*

$$\begin{aligned}
x^0 &= S(0) && (10)\\
S(0)^x &= S(0) && (11)\\
x^{S(y)} &= x * x^y && (12)\\
x^y * z^y &= (x * z)^y && (13)\\
x^{y+z} &= x^y * x^z && (14)
\end{aligned}$$

18.2 Example

Obviously this problem is about arithmetic. Thus the operations are just addition, multiplication and exponentiation.

18.3 A canonical System

For the canonical system the equations are simply oriented from left to right where $+$ and $*$ are ac (these rules are deleted). It was not possible to get the result without user interaction. The direction of the rules had to be added by hand, thus the system only showed local confluence.

18.4 Trace

The operation x^y is denoted by ^; $+$, $*$, S and 0 remain the same:

```
RRL-> read test/18a.eq
New constant set is: { 0 }
Equations read in are:
 1. (X + Y) == (Y + X)  [USER, 1]
 2. (X + (Y + Z)) == ((X + Y) + Z)  [USER, 2]
 3. (X * Y) == (Y * X)  [USER, 3]
 4. (X * (Y * Z)) == ((X * Y) * Z)  [USER, 4]
 5. (0 + X) == X  [USER, 5]
 6. (0 * X) == 0  [USER, 6]
 7. (S(X) + Y) == S((X + Y))  [USER, 7]
 8. (S(X) * Y) == ((X * Y) + Y)  [USER, 8]
 9. (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
----- Step 1 -----

´+´ has the commutative property now:
    (X + Y) == (Y + X)  [USER, 1]

----- Step 2 -----

´+´ has the associative and commutative property now:
    (X + (Y + Z)) == (Z + (X + Y))  [USER, 2]

----- Step 3 -----

´*´ has the commutative property now:
    (X * Y) == (Y * X)  [USER, 3]

----- Step 4 -----

´*´ has the associative and commutative property now:
    (X * (Y * Z)) == (Z * (X * Y))  [USER, 4]

Adding Rule:   [1] (X + 0) ---> X  [USER, 5]

Adding Rule:   [2] (X * 0) ---> 0  [USER, 6]

Trying to orient equation:
  (Y + S(X)) == S((X + Y))  [USER, 7]

  To prove:  (Y + S(X))  >  S((X + Y))
        or:  (Y + S(X))  <  S((X + Y))
  Here are some precedence suggestions:
       1.  + > S      2.  S > +
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 5 -----

Adding Rule:   [3] (Y + S(X)) ---> S((X + Y))  [USER, 7]

Trying to orient equation:
  (Y * S(X)) == (Y + (X * Y))  [USER, 8]

  To prove:  (Y * S(X))  >  (Y + (X * Y))
        or:  (Y * S(X))  <  (Y + (X * Y))
  Here are some precedence suggestions:
       1.  * > +      2.  + > *
       3.  S > *      4.  * > S
```

```
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> lr
Orientation of equations being done manually.

!!!!!! Warning: Rewriting may not terminate !!!!
!!!!!! Warning: Resulting system will be locally-confluent;
         But it may not be canonical  !!!!

Is it ok to continue ? (Y,N,YES,NO) yes
----- Step 6 -----

Adding Rule:   [4] (Y * S(X)) ---> (Y + (X * Y))  [USER, 8]

Trying to orient equation:
  (X * (Y + Z)) == ((X * Y) + (X * Z))  [USER, 9]

  To prove:  (X * (Y + Z))  >  ((X * Y) + (X * Z))
        or:  (X * (Y + Z))  <  ((X * Y) + (X * Z))
  Here are some precedence suggestions:
       1.  * > +      2.  + > *
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 7 -----

Adding Rule:   [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { *, + }
No equations in current system.

Rules:
  [1] (X + 0) ---> X  [USER, 5]
  [2] (X * 0) ---> 0  [USER, 6]
  [3] (Y + S(X)) ---> S((X + Y))  [USER, 7]
  [4] (Y * S(X)) ---> (Y + (X * Y))  [USER, 8]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 9]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 8 -----

Your system is locally confluent.

  [1] (X + 0) ---> X  [USER, 5]
  [2] (X * 0) ---> 0  [USER, 6]
  [3] (Y + S(X)) ---> S((X + Y))  [USER, 7]
  [4] (Y * S(X)) ---> (Y + (X * Y))  [USER, 8]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 9]

Time used                          =   9.91 sec
Number of rules generated          = 5
```

```
Number of rules retained             = 5
Number of critical pairs             = 90
Number of unblocked critical pairs   = 43
Time spent in normalization          =   7.17 sec  (72.0 percent of time)
Time spent in unification            =   1.23 sec   (12.0 percent of time)
Time spent in ordering               =   0.19 sec   (1.0 percent of time)
Time spent in simplifying the rules  =   0.07 sec   (0.0 percent of time)
Time spent in blocking               =   0.37 sec   (3.0 percent of time)
Total time (including 'undo' action) = 993.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/18b.eq
Equations read in are:
 1. (X ^ 0) == S(0)  [USER, 10]
 2. (S(0) ^ X) == S(0)  [USER, 11]
 3. (X ^ S(Y)) == (X * (X ^ Y))  [USER, 12]
 4. ((X ^ Z) * (Y ^ Z)) == ((X * Y) ^ Z)  [USER, 13]
 5. (X ^ (Y + Z)) == ((X ^ Y) * (X ^ Z))  [USER, 14]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> option order m

Orientation of equations being done manually.

!!!!! Warning: Rewriting may not terminate !!!!
!!!!! Warning: Resulting system will be locally-confluent;
         But it may not be canonical  !!!!



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
The equation can only be oriented from left-to-right (LR)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (X ^ 0) == S(0)  [USER, 10]
RRL>ORDER> lr
----- Step 9 -----

Adding Rule:   [6] (X ^ 0) ---> S(0)  [USER, 10]

The equation can only be oriented from left-to-right (LR)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (S(0) ^ X) == S(0)  [USER, 11]
RRL>ORDER> lr
----- Step 10 -----

Adding Rule:   [7] (S(0) ^ X) ---> S(0)  [USER, 11]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (X ^ S(Y)) == (X * (X ^ Y))  [USER, 12]
RRL>ORDER> lr
----- Step 11 -----

Adding Rule:   [8] (X ^ S(Y)) ---> (X * (X ^ Y))  [USER, 12]
```

```
The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: ((X ^ Z) * (Y ^ Z)) == ((X * Y) ^ Z)  [USER, 13]
RRL>ORDER> lr
----- Step 12 -----

Adding Rule:   [9] ((X ^ Z) * (Y ^ Z)) ---> ((X * Y) ^ Z)  [USER, 13]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (X ^ (Y + Z)) == ((X ^ Y) * (X ^ Z))  [USER, 14]
RRL>ORDER> lr
----- Step 13 -----

Adding Rule:  [10] (X ^ (Y + Z)) ---> ((X ^ Y) * (X ^ Z))  [USER, 14]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { *, + }
No equations in current system.

Rules:
  [1] (X + 0) ---> X  [USER, 5]
  [2] (X * 0) ---> 0  [USER, 6]
  [3] (Y + S(X)) ---> S((X + Y))  [USER, 7]
  [4] (Y * S(X)) ---> (Y + (X * Y))  [USER, 8]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 9]
  [6] (X ^ 0) ---> S(0)  [USER, 10]
  [7] (S(0) ^ X) ---> S(0)  [USER, 11]
  [8] (X ^ S(Y)) ---> (X * (X ^ Y))  [USER, 12]
  [9] ((X ^ Z) * (Y ^ Z)) ---> ((X * Y) ^ Z)  [USER, 13]
 [10] (X ^ (Y + Z)) ---> ((X ^ Y) * (X ^ Z))  [USER, 14]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 14 -----

Your system is locally confluent.

  [1] (X + 0) ---> X  [USER, 5]
  [2] (X * 0) ---> 0  [USER, 6]
  [3] (Y + S(X)) ---> S((X + Y))  [USER, 7]
  [4] (Y * S(X)) ---> (Y + (X * Y))  [USER, 8]
  [5] (X * (Y + Z)) ---> ((X * Y) + (X * Z))  [USER, 9]
  [6] (X ^ 0) ---> S(0)  [USER, 10]
  [7] (S(0) ^ X) ---> S(0)  [USER, 11]
  [8] (X ^ S(Y)) ---> (X * (X ^ Y))  [USER, 12]
  [9] ((X ^ Z) * (Y ^ Z)) ---> ((X * Y) ^ Z)  [USER, 13]
 [10] (X ^ (Y + Z)) ---> ((X ^ Y) * (X ^ Z))  [USER, 14]

Time used                           =  13.72 sec
Number of rules generated           = 10
Number of rules retained            = 10
Number of critical pairs            = 121
Number of unblocked critical pairs  = 52
Time spent in normalization         =   9.06 sec  (66.0 percent of time)
```

```
Time spent in unification           =   1.77 sec   (12.0 percent of time)
Time spent in ordering              =   0.31 sec   (2.0 percent of time)
Time spent in simplifying the rules =   0.16 sec   (1.0 percent of time)
Time spent in blocking              =   0.45 sec   (3.0 percent of time)
Total time (including 'undo' action) = 387.0 sec


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

19 R. J. Popplestone's Theory of Robot's Moves

19.1 Axioms

Definition 19.1 *The following equational theory was considered by R.J. Popplestone for an axiomatization of robot's movements:*

$$\begin{aligned}
Tr(x,y,z) * Tr(u,v,w) &= Tr(x+u, y+v, z+w) && (1)\\
T(\theta) * T(\theta') &= T(\theta + \theta') && (2)\\
M * M &= I && (3)\\
T(\theta) * Tr(x,0,0) &= Tr(x,0,0) * T(\theta) && (4)\\
T(\theta) * M &= M * T(-\theta) && (5)\\
Tr(x,y,z) * M &= M * Tr(-x,-y,z) && (6)\\
& && (7)
\end{aligned}$$

Furthermore axioms handling nonabelian groups are added:

$$\begin{aligned}
I * X &= X && (8)\\
X^{-1} * X &= I && (9)\\
(X * Y) * Z &= X * (Y * Z) && (10)\\
X^{-1} * (X * Y) &= Y && (11)\\
X * I &= X && (12)\\
I^{-1} &= I && (13)\\
(X^{-1})^{-1} &= X && (14)\\
X * X^{-1} &= I && (15)\\
X * (X^{-1} * Y) &= Y && (16)\\
(X * Y)^{-1} &= Y^{-1} * X^{-1} && (17)
\end{aligned}$$

Finally axioms handling the scalars as an Abelian group are added:

$$\begin{aligned}
x + y &= y + x && (18)\\
(x+y)+z &= x + (y+z) && (19)\\
x + 0 &= x && (20)\\
x + (-x) &= 0 && (21)
\end{aligned}$$

19.2 Example

The following interpretation is taken from [Hul80]:

The scalars are the real numbers with the corresponding operations. They form an Abelian group w.r.t. addition. The non-Abelian group is the set of 4-dimensional invertible real matrixes with the corresponding operations.

Finally we have to interpret T, Tr, M and I.

$$\begin{aligned}
T(\theta) &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) & 0 \\ 0 & -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
Tr(x,y,z) &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & z & 1 \end{pmatrix} \\
M &= \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
I &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

19.3 A canonical System

There is the following canonical system (+ ac):

$$\begin{aligned}
I * X &\longrightarrow X && (1) \\
X^{-1} * X &\longrightarrow I && (2) \\
(X * Y) * Z &\longrightarrow X * (Y * Z) && (3) \\
X^{-1} * (X * Y) &\longrightarrow Y && (4) \\
X * I &\longrightarrow X && (5) \\
I^{-1} &\longrightarrow I && (6) \\
(X^{-1})^{-1} &\longrightarrow X && (7) \\
X * X^{-1} &\longrightarrow I && (8) \\
X * (X^{-1} * Y) &\longrightarrow Y && (9) \\
(X * Y)^{-1} &\longrightarrow Y^{-1} * X^{-1} && (10) \\
x + 0 &\longrightarrow x && (11) \\
x + (-x) &\longrightarrow 0 && (12) \\
-0 &\longrightarrow 0 && (13) \\
-(-x) &\longrightarrow x && (14) \\
-(x + y) &\longrightarrow (-x) + (-y) && (15) \\
Tr(x,y,z) * Tr(u,v,w) &\longrightarrow Tr(x+u, y+v, z+w) && (16)
\end{aligned}$$

$$
\begin{aligned}
Tr(x,y,z)*(Tr(u,v,w)*X) &\longrightarrow Tr(x+u,y+v,z+w)*X && (17)\\
Tr(x,y,z)^{-1} &\longrightarrow Tr(-x,-y,-z) && (18)\\
Tr(x,0,0) &\longrightarrow Tr_1(x) && (19)\\
Tr_1(0) &\longrightarrow I && (20)\\
Tr_1(x)*Tr_1(u) &\longrightarrow Tr_1(x+u) && (21)\\
Tr_1(x)*(Tr_1(u)*X) &\longrightarrow Tr_1(x+u)*X && (22)\\
Tr_1(x)^{-1} &\longrightarrow Tr_1(-x) && (23)\\
Tr(0,y,z) &\longrightarrow Tr_2(y,z) && (24)\\
Tr_2(0,0) &\longrightarrow I && (25)\\
Tr_2(y,z)*Tr_2(v,w) &\longrightarrow Tr_2(y+v,z+w) && (26)\\
Tr_2(y,z)*(Tr_2(v,w)*X) &\longrightarrow Tr_2(y+v,z+w)*X && (27)\\
Tr_2(y,z)^{-1} &\longrightarrow Tr_2(-y,-z) && (28)\\
Tr_1(x)*Tr_2(y,z) &\longrightarrow Tr(x,y,z) && (29)\\
Tr_1(x)*(Tr_2(y,z)*X) &\longrightarrow Tr(x,y,z)*X && (30)\\
Tr_2(y,z)*Tr_1(x) &\longrightarrow Tr(x,y,z) && (31)\\
Tr_2(y,z)*(Tr_1(x)*X) &\longrightarrow Tr(x,y,z)*X && (32)\\
Tr_1(x)*Tr(u,v,w) &\longrightarrow Tr(x+u,v,w) && (33)\\
Tr_1(x)*(Tr(u,v,w)*X) &\longrightarrow Tr(x+u,v,w)*X && (34)\\
Tr(u,v,w)*Tr_1(x) &\longrightarrow Tr(x+u,v,w) && (35)\\
Tr(u,v,w)*(Tr_1(x)*X) &\longrightarrow Tr(x+u,v,w)*X && (36)\\
Tr_2(y,z)*Tr(u,v,w) &\longrightarrow Tr(u,y+v,z+w) && (37)\\
Tr_2(y,z)*(Tr(u,v,w)*X) &\longrightarrow Tr(u,y+v,z+w)*X && (38)\\
Tr(u,v,w)*Tr_2(y,z) &\longrightarrow Tr(u,y+v,z+w) && (39)\\
Tr(u,v,w)*(Tr_2(y,z)*X) &\longrightarrow Tr(u,y+v,z+w)*X && (40)\\
M*M &\longrightarrow I && (41)\\
M*(M*X) &\longrightarrow X && (42)\\
M^{-1} &\longrightarrow M && (43)\\
Tr(x,y,z)*M &\longrightarrow M*Tr(-x,-y,z) && (44)\\
Tr(x,y,z)*(M*X) &\longrightarrow M*(Tr(-x,-y,z)*X) && (45)\\
Tr_1(x)*M &\longrightarrow M*Tr_1(-x) && (46)\\
Tr_1(x)*(M*X) &\longrightarrow M*(Tr_1(-x)*X) && (47)\\
Tr_2(y,z)*M &\longrightarrow M*Tr_2(-y,z) && (48)\\
Tr_2(y,z)*(M*X) &\longrightarrow M*(Tr_2(-y,z)*X) && (49)\\
T(0) &\longrightarrow I && (50)\\
I(T(\theta)) &\longrightarrow T(-\theta) && (51)\\
T(\theta)*T(\theta') &\longrightarrow T(\theta+\theta') && (52)
\end{aligned}
$$

$$T(\theta) * (T(\theta') * X) \longrightarrow T(\theta + \theta') * X \tag{53}$$
$$T(\theta) * M \longrightarrow M * T(-\theta) \tag{54}$$
$$T(\theta) * (M * X) \longrightarrow M * (T(-\theta) * X) \tag{55}$$
$$T(\theta) * Tr_1(x) \longrightarrow Tr_1(x) * T(\theta) \tag{56}$$
$$T(\theta) * (Tr_1(x) * X) \longrightarrow Tr_1(x) * (T(\theta) * X) \tag{57}$$
$$T(\theta) * Tr(x, y, z) \longrightarrow Tr_1(x) * (T(\theta) * Tr_2(y, z)) \tag{58}$$
$$T(\theta) * (Tr(x, y, z) * X) \longrightarrow Tr_1(x) * (T(\theta) * (Tr_2(y, z) * X)) \tag{59}$$

It was not possible to get the result without user interaction. The direction of the rules had to be added by hand, thus the system only showed local confluence.

19.4 Trace

The operations T, $*$ and O remain the same. Tr, Tr_1 and Tr_2 are represented by TR, TR1 and TR2, resp., $^{-1}$ is represented by INV, I by ID and + by ADD. The output was edited, missing parts are indicated by "...".

```
RRL-> read test/19a.eq
New constant set is: { ID }
Equations read in are:
 1. (ID * X) == X  [USER, 1]
 2. (INV(X) * X) == ID  [USER, 2]
 3. ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]
 4. (INV(X) * (X * Y)) == Y  [USER, 4]
 5. (X * ID) == X  [USER, 5]
 6. INV(ID) == ID  [USER, 6]
 7. INV(INV(X)) == X  [USER, 7]
 8. (X * INV(X)) == ID  [USER, 8]
 9. (X * (INV(X) * Y)) == Y  [USER, 9]
10. INV((X * Y)) == (INV(Y) * INV(X))  [USER, 10]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
Adding Rule:   [1] (ID * X) ---> X  [USER, 1]

Adding Rule:   [2] (INV(X) * X) ---> ID  [USER, 2]

Trying to orient equation:
  ((X * Y) * Z) == (X * (Y * Z))  [USER, 3]

  To prove:  ((X * Y) * Z)  >  (X * (Y * Z))
        or:  ((X * Y) * Z)  <  (X * (Y * Z))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> status * lr
Operator, *, given status: LR

----- Step 1 -----

Adding Rule:   [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
```

```
Adding Rule:   [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]

Adding Rule:   [5] (X * ID) ---> X  [USER, 5]

Adding Rule:   [6] INV(ID) ---> ID  [USER, 6]

Adding Rule:   [7] INV(INV(X)) ---> X  [USER, 7]

Adding Rule:   [8] (X * INV(X)) ---> ID  [USER, 8]

Adding Rule:   [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]

Trying to orient equation:
  INV((X * Y)) == (INV(Y) * INV(X))  [USER, 10]

  To prove:  INV((X * Y))  >  (INV(Y) * INV(X))
        or:  INV((X * Y))  <  (INV(Y) * INV(X))
  Here are some precedence suggestions:
       1.  INV > *      2.  * > INV
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 2 -----

Adding Rule:  [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:

No equations in current system.

Rules:
  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb

...

Your system is canonical.

  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
```

```
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]

Time used                           =   0.36 sec
Number of rules generated           = 10
Number of rules retained            = 10
Number of critical pairs            = 48
Time spent in normalization         =   0.13 sec  (36.0 percent of time)
Time spent in unification           =   0.03 sec   (8.0 percent of time)
Time spent in ordering              =   0.16 sec   (44.0 percent of time)
Time spent in simplifying the rules  =   0.10 sec   (27.0 percent of time)
Total time (including 'undo' action) = 40.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/19b.eq
New constant set is: { 0 }
Equations read in are:
 1. ADD(X, Y) == ADD(Y, X)  [USER, 11]
 2. ADD(ADD(X, Y), Z) == ADD(X, ADD(Y, Z))  [USER, 12]
 3. ADD(X, 0) == X  [USER, 13]
 4. ADD(X, -(X)) == 0  [USER, 14]
 5. -(0) == 0  [USER, 15]
 6. -(-(X)) == X  [USER, 16]
 7. -(ADD(X, Y)) == ADD(-(X), -(Y))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
----- Step 4 -----

'ADD' has the commutative property now:
    ADD(X, Y) == ADD(Y, X)  [USER, 11]

----- Step 5 -----

'ADD' has the associative and commutative property now:
    ADD(Z, ADD(X, Y)) == ADD(X, ADD(Y, Z))  [USER, 12]

Adding Rule:  [11] ADD(X, 0) ---> X  [USER, 13]

Adding Rule:  [12] ADD(X, -(X)) ---> 0  [USER, 14]

Adding Rule:  [13] -(0) ---> 0  [USER, 15]

Adding Rule:  [14] -(-(X)) ---> X  [USER, 16]

Trying to orient equation:
  -(ADD(X, Y)) == ADD(-(X), -(Y))  [USER, 17]

  To prove:  -(ADD(X, Y))  >  ADD(-(X), -(Y))
        or:  -(ADD(X, Y))  <  ADD(-(X), -(Y))
  Here are some precedence suggestions:
       1.  - > ADD      2.  ADD > -
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 6 -----
```

```
Adding Rule:  [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { ADD }
No equations in current system.

Rules:
  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
 [11] ADD(X, 0) ---> X  [USER, 13]
 [12] ADD(X, -(X)) ---> 0  [USER, 14]
 [13] -(0) ---> 0  [USER, 15]
 [14] -(-(X)) ---> X  [USER, 16]
 [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 7 -----

Your system is canonical.

  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
 [11] ADD(X, 0) ---> X  [USER, 13]
 [12] ADD(X, -(X)) ---> 0  [USER, 14]
 [13] -(0) ---> 0  [USER, 15]
 [14] -(-(X)) ---> X  [USER, 16]
 [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]

Time used                           =   1.86 sec
Number of rules generated           = 15
Number of rules retained            = 15
Number of critical pairs            = 84
Number of unblocked critical pairs  = 11
Time spent in normalization         =   0.85 sec  (45.0 percent of time)
Time spent in unification           =   0.40 sec   (21.0 percent of time)
Time spent in ordering              =   0.26 sec   (13.0 percent of time)
Time spent in simplifying the rules =   0.18 sec   (9.0 percent of time)
```

```
Time spent in blocking              =   0.09 sec   (4.0 percent of time)
Total time (including 'undo' action) = 156.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/19c.eq
New constant set is: { M }
Equations read in are:
 1. (T(X) * T(Y)) == T(ADD(X, Y))  [USER, 18]
 2. (TRO(U, V, W) * TRO(X, Y, Z)) == TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 19]
 3. (T(X) * M) == (M * T(-(X)))  [USER, 20]
 4. (TRO(X, Y, Z) * M) == (M * TRO(-(X), -(Y), Z))  [USER, 21]
 5. (T(X) * TRO(Y, 0, 0)) == (TRO(Y, 0, 0) * T(X))  [USER, 22]
 6. (M * M) == ID  [USER, 23]
 7. TRO(X, 0, 0) == TR1(X)  [USER, 24]
 8. TRO(0, Y, Z) == TR2(Y, Z)  [USER, 25]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
Trying to orient equation:
  (T(X) * T(Y)) == T(ADD(X, Y))  [USER, 18]

  To prove:  (T(X) * T(Y))  >  T(ADD(X, Y))
        or:  (T(X) * T(Y))  <  T(ADD(X, Y))
  Here are some precedence suggestions:
      1.  * > T      2.  T > *
      3.  * > ADD      4.  ADD > *
      5.  T > ADD      6.  ADD > T
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1 3 5
----- Step 8 -----

Adding Rule:  [16] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 18]

Trying to orient equation:
  (TRO(U, V, W) * TRO(X, Y, Z)) == TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 19]

  To prove:  (TRO(U, V, W) * TRO(X, Y, Z))  >  TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))
        or:  (TRO(U, V, W) * TRO(X, Y, Z))  <  TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))
  Here are some precedence suggestions:
      1.  * > TRO      2.  TRO > *
      3.  TRO > ADD      4.  ADD > TRO
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1 3
----- Step 9 -----

Adding Rule:  [17] (TRO(U, V, W) * TRO(X, Y, Z)) ---> TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 19]

Trying to orient equation:
  (T(X) * M) == (M * T(-(X)))  [USER, 20]

  To prove:  (T(X) * M)  >  (M * T(-(X)))
        or:  (T(X) * M)  <  (M * T(-(X)))
  Here are some precedence suggestions:
      1.  * > -      2.  - > *
```

```
      3.  T > -      4.  - > T
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1 3
----- Step 10 -----

Adding Rule:  [18] (T(X) * M) ---> (M * T(-(X)))  [USER, 20]

Adding Rule:  [19] (TRO(X, Y, Z) * M) ---> (M * TRO(-(X), -(Y), Z))  [USER, 21]

Trying to orient equation:
  (T(X) * TRO(Y, 0, 0)) == (TRO(Y, 0, 0) * T(X))  [USER, 22]

  To prove:  (T(X) * TRO(Y, 0, 0))  >  (TRO(Y, 0, 0) * T(X))
        or:  (T(X) * TRO(Y, 0, 0))  <  (TRO(Y, 0, 0) * T(X))
  Here are some precedence suggestions:
      1.  T > TRO      2.  TRO > T
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 11 -----

Sorry did not work, try again.

  To prove:  (T(X) * TRO(Y, 0, 0))  >  (TRO(Y, 0, 0) * T(X))
        or:  (T(X) * TRO(Y, 0, 0))  <  (TRO(Y, 0, 0) * T(X))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> lr
Orientation of equations being done manually.

!!!!!! Warning: Rewriting may not terminate !!!!
!!!!!! Warning: Resulting system will be locally-confluent;
         But it may not be canonical  !!!!

Is it ok to continue ? (Y,N,YES,NO) yes
Adding Rule:  [20] (T(X) * TRO(Y, 0, 0)) ---> (TRO(Y, 0, 0) * T(X))  [USER, 22]

Adding Rule:  [21] (M * M) ---> ID  [USER, 23]

Trying to orient equation:
  TRO(X, 0, 0) == TR1(X)  [USER, 24]

  To prove:  TRO(X, 0, 0)  >  TR1(X)
        or:  TRO(X, 0, 0)  <  TR1(X)
  Here are some precedence suggestions:
      1.  TRO > TR1      2.  TR1 > TRO
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 12 -----

Adding Rule:  [22] TRO(X, 0, 0) ---> TR1(X)  [USER, 24]

  Deleting rule: [20] (T(X) * TRO(Y, 0, 0)) ---> (TRO(Y, 0, 0) * T(X))  [USER, 22]

Trying to orient equation:
  (T(X) * TR1(Y)) == (TR1(Y) * T(X))  [DELETED, 20]
```

```
  To prove:  (T(X) * TR1(Y))  >  (TR1(Y) * T(X))
        or:  (T(X) * TR1(Y))  <  (TR1(Y) * T(X))
I have no precedence suggestions.
Try doing Equiv or Status.

Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> lr
----- Step 13 -----

Adding Rule:  [23] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [DELETED, 20]

Trying to orient equation:
  TR0(0, Y, Z) == TR2(Y, Z)  [USER, 25]

  To prove:  TR0(0, Y, Z)  >  TR2(Y, Z)
        or:  TR0(0, Y, Z)  <  TR2(Y, Z)
  Here are some precedence suggestions:
       1.  TR0 > TR2      2.  TR2 > TR0
Either type a list of numbers or
Type Abort, Display, Drop, Equiv, LR, MakeEq, Operator, Postpone, Quit, RL,
     Status, Superpose, Twoway, Undo or Help.
RRL>KB> 1
----- Step 14 -----

Adding Rule:  [24] TR0(0, Y, Z) ---> TR2(Y, Z)  [USER, 25]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { ADD }
No equations in current system.

Rules:
  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
 [11] ADD(X, 0) ---> X  [USER, 13]
 [12] ADD(X, -(X)) ---> 0  [USER, 14]
 [13] -(0) ---> 0  [USER, 15]
 [14] -(-(X)) ---> X  [USER, 16]
 [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]
 [16] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 18]
 [17] (TR0(U, V, W) * TR0(X, Y, Z)) ---> TR0(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 19]
 [18] (T(X) * M) ---> (M * T(-(X)))  [USER, 20]
 [19] (TR0(X, Y, Z) * M) ---> (M * TR0(-(X), -(Y), Z))  [USER, 21]
 [21] (M * M) ---> ID  [USER, 23]
 [22] TR0(X, 0, 0) ---> TR1(X)  [USER, 24]
 [23] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [DELETED, 20]
 [24] TR0(0, Y, Z) ---> TR2(Y, Z)  [USER, 25]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
```

```
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> delete rule 16 17 18 19 21 22 23 24
Deleted rule:  [16] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 18]

Deleted rule:  [17] (TRO(U, V, W) * TRO(X, Y, Z)) ---> TRO(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 19]

Deleted rule:  [18] (T(X) * M) ---> (M * T(-(X)))  [USER, 20]

Deleted rule:  [19] (TRO(X, Y, Z) * M) ---> (M * TRO(-(X), -(Y), Z))  [USER, 21]

Deleted rule:  [21] (M * M) ---> ID  [USER, 23]

Deleted rule:  [22] TRO(X, 0, 0) ---> TR1(X)  [USER, 24]

Deleted rule:  [23] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [DELETED, 20]

Deleted rule:  [24] TRO(0, Y, Z) ---> TR2(Y, Z)  [USER, 25]



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { ADD }
No equations in current system.

Rules:
  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
 [11] ADD(X, 0) ---> X  [USER, 13]
 [12] ADD(X, -(X)) ---> 0  [USER, 14]
 [13] -(0) ---> 0  [USER, 15]
 [14] -(-(X)) ---> X  [USER, 16]
 [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> read test/19d.eq
New constant set is: { M }
Equations read in are:
 1. (TR(X, Y, Z) * TR(U, V, W)) == TR(ADD(X, U), ADD(Y, V), ADD(Z, W))  [USER, 26]
 2. (TR(X, Y, Z) * (TR(U, V, W) * ZZ)) == (TR(ADD(X, U), ADD(Y, V), ADD(Z, W)) * ZZ)  [USER, 27]
 3. INV(TR(X, Y, Z)) == TR(-(X), -(Y), -(Z))  [USER, 28]
 4. TR(X, 0, 0) == TR1(X)  [USER, 29]
 5. TR1(0) == ID  [USER, 30]
 6. (TR1(X) * TR1(Y)) == TR1(ADD(X, Y))  [USER, 31]
 7. (TR1(X) * (TR1(Y) * Z)) == (TR1(ADD(X, Y)) * Z)  [USER, 32]
 8. INV(TR1(X)) == TR1(-(X))  [USER, 33]
 9. TR(0, Y, Z) == TR2(Y, Z)  [USER, 34]
10. TR2(0, 0) == ID  [USER, 35]
```

```
11. (TR2(Y, Z) * TR2(V, W)) == TR2(ADD(Y, V), ADD(Z, W))  [USER, 36]
12. (TR2(Y, Z) * (TR2(V, W) * X)) == (TR2(ADD(Y, V), ADD(Z, W)) * X)  [USER, 37]
13. INV(TR2(Y, Z)) == TR2(-(Y), -(Z))  [USER, 38]
14. (TR1(X) * TR2(Y, Z)) == TR(X, Y, Z)  [USER, 39]
15. (TR1(X) * (TR2(Y, Z) * U)) == (TR(X, Y, Z) * U)  [USER, 40]
16. (TR2(Y, Z) * TR1(X)) == TR(X, Y, Z)  [USER, 41]
17. (TR2(Y, Z) * (TR1(X) * U)) == (TR(X, Y, Z) * U)  [USER, 42]
18. (TR1(X) * TR(U, V, W)) == TR(ADD(X, U), V, W)  [USER, 43]
19. (TR1(X) * (TR(U, V, W) * Y)) == (TR(ADD(X, U), V, W) * Y)  [USER, 44]
20. (TR(U, V, W) * TR1(X)) == TR(ADD(X, U), V, W)  [USER, 45]
21. (TR(U, V, W) * (TR1(X) * Y)) == (TR(ADD(X, U), V, W) * Y)  [USER, 46]
22. (TR2(Y, Z) * TR(U, V, W)) == TR(U, ADD(Y, V), ADD(Z, W))  [USER, 47]
23. (TR2(Y, Z) * (TR(U, V, W) * X)) == (TR(U, ADD(Y, V), ADD(Z, W)) * X)  [USER, 48]
24. (TR(U, V, W) * TR2(Y, Z)) == TR(U, ADD(V, Y), ADD(W, Z))  [USER, 49]
25. (TR(U, V, W) * (TR2(Y, Z) * X)) == (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 50]
26. (M * M) == ID  [USER, 51]
27. (M * (M * X)) == X  [USER, 52]
28. INV(M) == M  [USER, 53]
29. (TR(X, Y, Z) * M) == (M * TR(-(X), -(Y), Z))  [USER, 54]
30. (TR(X, Y, Z) * (M * U)) == (M * (TR(-(X), -(Y), Z) * U))  [USER, 55]
31. (TR1(X) * M) == (M * TR1(-(X)))  [USER, 56]
32. (TR1(X) * (M * U)) == (M * (TR1(-(X)) * U))  [USER, 57]
33. (TR2(Y, Z) * M) == (M * TR2(-(Y), Z))  [USER, 58]
34. (TR2(Y, Z) * (M * X)) == (M * (TR2(-(Y), Z) * X))  [USER, 59]
35. T(0) == ID  [USER, 60]
36. INV(T(X)) == T(-(X))  [USER, 61]
37. (T(X) * T(Y)) == T(ADD(X, Y))  [USER, 62]
38. (T(X) * (T(Y) * Z)) == (T(ADD(X, Y)) * Z)  [USER, 63]
39. (T(X) * M) == (M * T(-(X)))  [USER, 64]
40. (T(X) * (M * Y)) == (M * (T(-(X)) * Y))  [USER, 65]
41. (T(X) * TR1(Y)) == (TR1(Y) * T(X))  [USER, 66]
42. (T(X) * (TR1(Y) * Z)) == (TR1(Y) * (T(X) * Z))  [USER, 67]
43. (T(U) * TR(X, Y, Z)) == (TR1(X) * (T(U) * TR2(Y, Z)))  [USER, 68]
44. (T(U) * (TR(X, Y, Z) * V)) == (TR1(X) * (T(U) * (TR2(Y, Z) * V)))  [USER, 69]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> option order m

Orientation of equations being done manually.

!!!!!! Warning: Rewriting may not terminate !!!!
!!!!!! Warning: Resulting system will be locally-confluent;
         But it may not be canonical  !!!!



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> makerule
The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(X, Y, Z) * TR(U, V, W)) == TR(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 26]
RRL>ORDER> lr
----- Step 15 -----

Adding Rule:  [25] (TR(X, Y, Z) * TR(U, V, W)) ---> TR(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 26]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(ADD(U, X), ADD(V, Y), ADD(W, Z)) * ZZ) == (TR(X, Y, Z) * (TR(U, V, W) * ZZ))  [USER, 27]
RRL>ORDER> rl
```

```
----- Step 16 -----

Adding Rule:  [26] (TR(X, Y, Z) * (TR(U, V, W) * ZZ)) --->
(TR(ADD(U, X), ADD(V, Y), ADD(W, Z)) * ZZ)  [USER, 27]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: INV(TR(X, Y, Z)) == TR(-(X), -(Y), -(Z))  [USER, 28]
RRL>ORDER> lr
----- Step 17 -----

Adding Rule:  [27] INV(TR(X, Y, Z)) ---> TR(-(X), -(Y), -(Z))  [USER, 28]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: TR(X, 0, 0) == TR1(X)  [USER, 29]
RRL>ORDER> lr
----- Step 18 -----

Adding Rule:  [28] TR(X, 0, 0) ---> TR1(X)  [USER, 29]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: TR1(0) == ID  [USER, 30]
RRL>ORDER> lr
----- Step 19 -----

Adding Rule:  [29] TR1(0) ---> ID  [USER, 30]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * TR1(Y)) == TR1(ADD(X, Y))  [USER, 31]
RRL>ORDER> lr
----- Step 20 -----

Adding Rule:  [30] (TR1(X) * TR1(Y)) ---> TR1(ADD(X, Y))  [USER, 31]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(ADD(X, Y)) * Z) == (TR1(X) * (TR1(Y) * Z))  [USER, 32]
RRL>ORDER> rl
----- Step 21 -----

Adding Rule:  [31] (TR1(X) * (TR1(Y) * Z)) ---> (TR1(ADD(X, Y)) * Z)  [USER, 32]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: INV(TR1(X)) == TR1(-(X))  [USER, 33]
RRL>ORDER> lr
----- Step 22 -----

Adding Rule:  [32] INV(TR1(X)) ---> TR1(-(X))  [USER, 33]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: TR(0, Y, Z) == TR2(Y, Z)  [USER, 34]
RRL>ORDER> lr
----- Step 23 -----

Adding Rule:  [33] TR(0, Y, Z) ---> TR2(Y, Z)  [USER, 34]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: TR2(0, 0) == ID  [USER, 35]
RRL>ORDER> lr
```

```
----- Step 24 -----

Adding Rule:  [34] TR2(0, 0) ---> ID  [USER, 35]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * TR2(V, W)) == TR2(ADD(V, Y), ADD(W, Z))  [USER, 36]
RRL>ORDER> lr
----- Step 25 -----

Adding Rule:  [35] (TR2(Y, Z) * TR2(V, W)) ---> TR2(ADD(V, Y), ADD(W, Z))  [USER, 36]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(ADD(V, Y), ADD(W, Z)) * X) == (TR2(Y, Z) * (TR2(V, W) * X))  [USER, 37]
RRL>ORDER> rl
----- Step 26 -----

Adding Rule:  [36] (TR2(Y, Z) * (TR2(V, W) * X)) ---> (TR2(ADD(V, Y), ADD(W, Z)) * X)  [USER, 37]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: INV(TR2(Y, Z)) == TR2(-(Y), -(Z))  [USER, 38]
RRL>ORDER> lr
----- Step 27 -----

Adding Rule:  [37] INV(TR2(Y, Z)) ---> TR2(-(Y), -(Z))  [USER, 38]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * TR2(Y, Z)) == TR(X, Y, Z)  [USER, 39]
RRL>ORDER> lr
----- Step 28 -----

Adding Rule:  [38] (TR1(X) * TR2(Y, Z)) ---> TR(X, Y, Z)  [USER, 39]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * (TR2(Y, Z) * U)) == (TR(X, Y, Z) * U)  [USER, 40]
RRL>ORDER> lr
----- Step 29 -----

Adding Rule:  [39] (TR1(X) * (TR2(Y, Z) * U)) ---> (TR(X, Y, Z) * U)  [USER, 40]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * TR1(X)) == TR(X, Y, Z)  [USER, 41]
RRL>ORDER> lr
----- Step 30 -----

Adding Rule:  [40] (TR2(Y, Z) * TR1(X)) ---> TR(X, Y, Z)  [USER, 41]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * (TR1(X) * U)) == (TR(X, Y, Z) * U)  [USER, 42]
RRL>ORDER> lr
----- Step 31 -----

Adding Rule:  [41] (TR2(Y, Z) * (TR1(X) * U)) ---> (TR(X, Y, Z) * U)  [USER, 42]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * TR(U, V, W)) == TR(ADD(U, X), V, W)  [USER, 43]
RRL>ORDER> lr
----- Step 32 -----
```

```
Adding Rule:  [42] (TR1(X) * TR(U, V, W)) ---> TR(ADD(U, X), V, W)  [USER, 43]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * (TR(U, V, W) * Y)) == (TR(ADD(U, X), V, W) * Y)  [USER, 44]
RRL>ORDER> lr
----- Step 33 -----

Adding Rule:  [43] (TR1(X) * (TR(U, V, W) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 44]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(U, V, W) * TR1(X)) == TR(ADD(U, X), V, W)  [USER, 45]
RRL>ORDER> lr
----- Step 34 -----

Adding Rule:  [44] (TR(U, V, W) * TR1(X)) ---> TR(ADD(U, X), V, W)  [USER, 45]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(U, V, W) * (TR1(X) * Y)) == (TR(ADD(U, X), V, W) * Y)  [USER, 46]
RRL>ORDER> lr
----- Step 35 -----

Adding Rule:  [45] (TR(U, V, W) * (TR1(X) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 46]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * TR(U, V, W)) == TR(U, ADD(V, Y), ADD(W, Z))  [USER, 47]
RRL>ORDER> lr
----- Step 36 -----

Adding Rule:  [46] (TR2(Y, Z) * TR(U, V, W)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 47]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * (TR(U, V, W) * X)) == (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 48]
RRL>ORDER> lr
----- Step 37 -----

Adding Rule:  [47] (TR2(Y, Z) * (TR(U, V, W) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 48]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(U, V, W) * TR2(Y, Z)) == TR(U, ADD(V, Y), ADD(W, Z))  [USER, 49]
RRL>ORDER> lr
----- Step 38 -----

Adding Rule:  [48] (TR(U, V, W) * TR2(Y, Z)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 49]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(U, V, W) * (TR2(Y, Z) * X)) == (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 50]
RRL>ORDER> lr
----- Step 39 -----

Adding Rule:  [49] (TR(U, V, W) * (TR2(Y, Z) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 50]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * M) == ID  [USER, 51]
RRL>ORDER> lr
----- Step 40 -----
```

```
Adding Rule:  [50] (M * M) ---> ID  [USER, 51]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * (M * X)) == X  [USER, 52]
RRL>ORDER> lr
----- Step 41 -----

Adding Rule:  [51] (M * (M * X)) ---> X  [USER, 52]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: INV(M) == M  [USER, 53]
RRL>ORDER> lr
----- Step 42 -----

Adding Rule:  [52] INV(M) ---> M  [USER, 53]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR(X, Y, Z) * M) == (M * TR(-(X), -(Y), Z))  [USER, 54]
RRL>ORDER> lr
----- Step 43 -----

Adding Rule:  [53] (TR(X, Y, Z) * M) ---> (M * TR(-(X), -(Y), Z))  [USER, 54]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * (TR(-(X), -(Y), Z) * U)) == (TR(X, Y, Z) * (M * U))  [USER, 55]
RRL>ORDER> rl
----- Step 44 -----

Adding Rule:  [54] (TR(X, Y, Z) * (M * U)) ---> (M * (TR(-(X), -(Y), Z) * U))  [USER, 55]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR1(X) * M) == (M * TR1(-(X)))  [USER, 56]
RRL>ORDER> lr
----- Step 45 -----

Adding Rule:  [55] (TR1(X) * M) ---> (M * TR1(-(X)))  [USER, 56]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * (TR1(-(X)) * U)) == (TR1(X) * (M * U))  [USER, 57]
RRL>ORDER> rl
----- Step 46 -----

Adding Rule:  [56] (TR1(X) * (M * U)) ---> (M * (TR1(-(X)) * U))  [USER, 57]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (TR2(Y, Z) * M) == (M * TR2(-(Y), Z))  [USER, 58]
RRL>ORDER> lr
----- Step 47 -----

Adding Rule:  [57] (TR2(Y, Z) * M) ---> (M * TR2(-(Y), Z))  [USER, 58]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * (TR2(-(Y), Z) * X)) == (TR2(Y, Z) * (M * X))  [USER, 59]
RRL>ORDER> rl
----- Step 48 -----

Adding Rule:  [58] (TR2(Y, Z) * (M * X)) ---> (M * (TR2(-(Y), Z) * X))  [USER, 59]
```

```
The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: T(0) == ID  [USER, 60]
RRL>ORDER> lr
----- Step 49 -----

Adding Rule:  [59] T(0) ---> ID  [USER, 60]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: INV(T(X)) == T(-(X))  [USER, 61]
RRL>ORDER> lr
----- Step 50 -----

Adding Rule:  [60] INV(T(X)) ---> T(-(X))  [USER, 61]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(X) * T(Y)) == T(ADD(X, Y))  [USER, 62]
RRL>ORDER> lr
----- Step 51 -----

Adding Rule:  [61] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 62]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(ADD(X, Y)) * Z) == (T(X) * (T(Y) * Z))  [USER, 63]
RRL>ORDER> rl
----- Step 52 -----

Adding Rule:  [62] (T(X) * (T(Y) * Z)) ---> (T(ADD(X, Y)) * Z)  [USER, 63]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(X) * M) == (M * T(-(X)))  [USER, 64]
RRL>ORDER> lr
----- Step 53 -----

Adding Rule:  [63] (T(X) * M) ---> (M * T(-(X)))  [USER, 64]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (M * (T(-(X)) * Y)) == (T(X) * (M * Y))  [USER, 65]
RRL>ORDER> rl
----- Step 54 -----

Adding Rule:  [64] (T(X) * (M * Y)) ---> (M * (T(-(X)) * Y))  [USER, 65]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(X) * TR1(Y)) == (TR1(Y) * T(X))  [USER, 66]
RRL>ORDER> lr
----- Step 55 -----

Adding Rule:  [65] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [USER, 66]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(X) * (TR1(Y) * Z)) == (TR1(Y) * (T(X) * Z))  [USER, 67]
RRL>ORDER> lr
----- Step 56 -----

Adding Rule:  [66] (T(X) * (TR1(Y) * Z)) ---> (TR1(Y) * (T(X) * Z))  [USER, 67]
```

```
The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(U) * TR(X, Y, Z)) == (TR1(X) * (T(U) * TR2(Y, Z)))  [USER, 68]
RRL>ORDER> lr
----- Step 57 -----

Adding Rule:  [67] (T(U) * TR(X, Y, Z)) ---> (TR1(X) * (T(U) * TR2(Y, Z)))  [USER, 68]

The equation can be oriented from left-to-right (LR) or right-to-left (RL)
Type Add-op, Give-Stat, LR, RL, Postpone, Undo, Quit, or Help.
To orient: (T(U) * (TR(X, Y, Z) * V)) == (TR1(X) * (T(U) * (TR2(Y, Z) * V)))  [USER, 69]
RRL>ORDER> lr
----- Step 58 -----

Adding Rule:  [68] (T(U) * (TR(X, Y, Z) * V)) ---> (TR1(X) * (T(U) * (TR2(Y, Z) * V)))  [USER, 69]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> list
The arities of the operators are:


Associative & commutative operator set = { ADD }
No equations in current system.

Rules:
  [1] (ID * X) ---> X  [USER, 1]
  [2] (INV(X) * X) ---> ID  [USER, 2]
  [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
  [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
  [5] (X * ID) ---> X  [USER, 5]
  [6] INV(ID) ---> ID  [USER, 6]
  [7] INV(INV(X)) ---> X  [USER, 7]
  [8] (X * INV(X)) ---> ID  [USER, 8]
  [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
 [10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
 [11] ADD(X, 0) ---> X  [USER, 13]
 [12] ADD(X, -(X)) ---> 0  [USER, 14]
 [13] -(0) ---> 0  [USER, 15]
 [14] -(-(X)) ---> X  [USER, 16]
 [15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]
 [25] (TR(X, Y, Z) * TR(U, V, W)) ---> TR(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 26]
 [26] (TR(X, Y, Z) * (TR(U, V, W) * ZZ)) ---> (TR(ADD(U, X), ADD(V, Y), ADD(W, Z)) * ZZ)  [USER, 27]
 [27] INV(TR(X, Y, Z)) ---> TR(-(X), -(Y), -(Z))  [USER, 28]
 [28] TR(X, 0, 0) ---> TR1(X)  [USER, 29]
 [29] TR1(0) ---> ID  [USER, 30]
 [30] (TR1(X) * TR1(Y)) ---> TR1(ADD(X, Y))  [USER, 31]
 [31] (TR1(X) * (TR1(Y) * Z)) ---> (TR1(ADD(X, Y)) * Z)  [USER, 32]
 [32] INV(TR1(X)) ---> TR1(-(X))  [USER, 33]
 [33] TR(0, Y, Z) ---> TR2(Y, Z)  [USER, 34]
 [34] TR2(0, 0) ---> ID  [USER, 35]
 [35] (TR2(Y, Z) * TR2(V, W)) ---> TR2(ADD(V, Y), ADD(W, Z))  [USER, 36]
 [36] (TR2(Y, Z) * (TR2(V, W) * X)) ---> (TR2(ADD(V, Y), ADD(W, Z)) * X)  [USER, 37]
 [37] INV(TR2(Y, Z)) ---> TR2(-(Y), -(Z))  [USER, 38]
 [38] (TR1(X) * TR2(Y, Z)) ---> TR(X, Y, Z)  [USER, 39]
 [39] (TR1(X) * (TR2(Y, Z) * U)) ---> (TR(X, Y, Z) * U)  [USER, 40]
 [40] (TR2(Y, Z) * TR1(X)) ---> TR(X, Y, Z)  [USER, 41]
 [41] (TR2(Y, Z) * (TR1(X) * U)) ---> (TR(X, Y, Z) * U)  [USER, 42]
 [42] (TR1(X) * TR(U, V, W)) ---> TR(ADD(U, X), V, W)  [USER, 43]
 [43] (TR1(X) * (TR(U, V, W) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 44]
 [44] (TR(U, V, W) * TR1(X)) ---> TR(ADD(U, X), V, W)  [USER, 45]
 [45] (TR(U, V, W) * (TR1(X) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 46]
 [46] (TR2(Y, Z) * TR(U, V, W)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 47]
```

```
[47] (TR2(Y, Z) * (TR(U, V, W) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 48]
[48] (TR(U, V, W) * TR2(Y, Z)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 49]
[49] (TR(U, V, W) * (TR2(Y, Z) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 50]
[50] (M * M) ---> ID  [USER, 51]
[51] (M * (M * X)) ---> X  [USER, 52]
[52] INV(M) ---> M  [USER, 53]
[53] (TR(X, Y, Z) * M) ---> (M * TR(-(X), -(Y), Z))  [USER, 54]
[54] (TR(X, Y, Z) * (M * U)) ---> (M * (TR(-(X), -(Y), Z) * U))  [USER, 55]
[55] (TR1(X) * M) ---> (M * TR1(-(X)))  [USER, 56]
[56] (TR1(X) * (M * U)) ---> (M * (TR1(-(X)) * U))  [USER, 57]
[57] (TR2(Y, Z) * M) ---> (M * TR2(-(Y), Z))  [USER, 58]
[58] (TR2(Y, Z) * (M * X)) ---> (M * (TR2(-(Y), Z) * X))  [USER, 59]
[59] T(0) ---> ID  [USER, 60]
[60] INV(T(X)) ---> T(-(X))  [USER, 61]
[61] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 62]
[62] (T(X) * (T(Y) * Z)) ---> (T(ADD(X, Y)) * Z)  [USER, 63]
[63] (T(X) * M) ---> (M * T(-(X)))  [USER, 64]
[64] (T(X) * (M * Y)) ---> (M * (T(-(X)) * Y))  [USER, 65]
[65] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [USER, 66]
[66] (T(X) * (TR1(Y) * Z)) ---> (TR1(Y) * (T(X) * Z))  [USER, 67]
[67] (T(U) * TR(X, Y, Z)) ---> (TR1(X) * (T(U) * TR2(Y, Z)))  [USER, 68]
[68] (T(U) * (TR(X, Y, Z) * V)) ---> (TR1(X) * (T(U) * (TR2(Y, Z) * V)))  [USER, 69]


Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> akb
----- Step 59 -----

Your system is locally confluent.

 [1] (ID * X) ---> X  [USER, 1]
 [2] (INV(X) * X) ---> ID  [USER, 2]
 [3] ((X * Y) * Z) ---> (X * (Y * Z))  [USER, 3]
 [4] (INV(X) * (X * Y)) ---> Y  [USER, 4]
 [5] (X * ID) ---> X  [USER, 5]
 [6] INV(ID) ---> ID  [USER, 6]
 [7] INV(INV(X)) ---> X  [USER, 7]
 [8] (X * INV(X)) ---> ID  [USER, 8]
 [9] (X * (INV(X) * Y)) ---> Y  [USER, 9]
[10] INV((X * Y)) ---> (INV(Y) * INV(X))  [USER, 10]
[11] ADD(X, 0) ---> X  [USER, 13]
[12] ADD(X, -(X)) ---> 0  [USER, 14]
[13] -(0) ---> 0  [USER, 15]
[14] -(-(X)) ---> X  [USER, 16]
[15] -(ADD(X, Y)) ---> ADD(-(X), -(Y))  [USER, 17]
[25] (TR(X, Y, Z) * TR(U, V, W)) ---> TR(ADD(U, X), ADD(V, Y), ADD(W, Z))  [USER, 26]
[26] (TR(X, Y, Z) * (TR(U, V, W) * ZZ)) ---> (TR(ADD(U, X), ADD(V, Y), ADD(W, Z)) * ZZ)  [USER, 27]
[27] INV(TR(X, Y, Z)) ---> TR(-(X), -(Y), -(Z))  [USER, 28]
[28] TR(X, 0, 0) ---> TR1(X)  [USER, 29]
[29] TR1(0) ---> ID  [USER, 30]
[30] (TR1(X) * TR1(Y)) ---> TR1(ADD(X, Y))  [USER, 31]
[31] (TR1(X) * (TR1(Y) * Z)) ---> (TR1(ADD(X, Y)) * Z)  [USER, 32]
[32] INV(TR1(X)) ---> TR1(-(X))  [USER, 33]
[33] TR(0, Y, Z) ---> TR2(Y, Z)  [USER, 34]
[34] TR2(0, 0) ---> ID  [USER, 35]
[35] (TR2(Y, Z) * TR2(V, W)) ---> TR2(ADD(V, Y), ADD(W, Z))  [USER, 36]
[36] (TR2(Y, Z) * (TR2(V, W) * X)) ---> (TR2(ADD(V, Y), ADD(W, Z)) * X)  [USER, 37]
[37] INV(TR2(Y, Z)) ---> TR2(-(Y), -(Z))  [USER, 38]
[38] (TR1(X) * TR2(Y, Z)) ---> TR(X, Y, Z)  [USER, 39]
[39] (TR1(X) * (TR2(Y, Z) * U)) ---> (TR(X, Y, Z) * U)  [USER, 40]
[40] (TR2(Y, Z) * TR1(X)) ---> TR(X, Y, Z)  [USER, 41]
[41] (TR2(Y, Z) * (TR1(X) * U)) ---> (TR(X, Y, Z) * U)  [USER, 42]
[42] (TR1(X) * TR(U, V, W)) ---> TR(ADD(U, X), V, W)  [USER, 43]
```

```
 [43] (TR1(X) * (TR(U, V, W) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 44]
 [44] (TR(U, V, W) * TR1(X)) ---> TR(ADD(U, X), V, W)  [USER, 45]
 [45] (TR(U, V, W) * (TR1(X) * Y)) ---> (TR(ADD(U, X), V, W) * Y)  [USER, 46]
 [46] (TR2(Y, Z) * TR(U, V, W)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 47]
 [47] (TR2(Y, Z) * (TR(U, V, W) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 48]
 [48] (TR(U, V, W) * TR2(Y, Z)) ---> TR(U, ADD(V, Y), ADD(W, Z))  [USER, 49]
 [49] (TR(U, V, W) * (TR2(Y, Z) * X)) ---> (TR(U, ADD(V, Y), ADD(W, Z)) * X)  [USER, 50]
 [50] (M * M) ---> ID  [USER, 51]
 [51] (M * (M * X)) ---> X  [USER, 52]
 [52] INV(M) ---> M  [USER, 53]
 [53] (TR(X, Y, Z) * M) ---> (M * TR(-(X), -(Y), Z))  [USER, 54]
 [54] (TR(X, Y, Z) * (M * U)) ---> (M * (TR(-(X), -(Y), Z) * U))  [USER, 55]
 [55] (TR1(X) * M) ---> (M * TR1(-(X)))  [USER, 56]
 [56] (TR1(X) * (M * U)) ---> (M * (TR1(-(X)) * U))  [USER, 57]
 [57] (TR2(Y, Z) * M) ---> (M * TR2(-(Y), Z))  [USER, 58]
 [58] (TR2(Y, Z) * (M * X)) ---> (M * (TR2(-(Y), Z) * X))  [USER, 59]
 [59] T(0) ---> ID  [USER, 60]
 [60] INV(T(X)) ---> T(-(X))  [USER, 61]
 [61] (T(X) * T(Y)) ---> T(ADD(X, Y))  [USER, 62]
 [62] (T(X) * (T(Y) * Z)) ---> (T(ADD(X, Y)) * Z)  [USER, 63]
 [63] (T(X) * M) ---> (M * T(-(X)))  [USER, 64]
 [64] (T(X) * (M * Y)) ---> (M * (T(-(X)) * Y))  [USER, 65]
 [65] (T(X) * TR1(Y)) ---> (TR1(Y) * T(X))  [USER, 66]
 [66] (T(X) * (TR1(Y) * Z)) ---> (TR1(Y) * (T(X) * Z))  [USER, 67]
 [67] (T(U) * TR(X, Y, Z)) ---> (TR1(X) * (T(U) * TR2(Y, Z)))  [USER, 68]
 [68] (T(U) * (TR(X, Y, Z) * V)) ---> (TR1(X) * (T(U) * (TR2(Y, Z) * V)))  [USER, 69]

Time used                           =  32.03 sec
Number of rules generated           = 68
Number of rules retained            = 59
Number of critical pairs            = 456
Number of unblocked critical pairs  = 45
Time spent in normalization         =  24.17 sec  (75.0 percent of time)
Time spent in unification           =   1.37 sec   (4.0 percent of time)
Time spent in ordering              =   3.77 sec   (11.0 percent of time)
Time spent in simplifying the rules =   3.72 sec   (11.0 percent of time)
Time spent in blocking              =   0.93 sec   (2.0 percent of time)
Total time (including 'undo' action) = 3074.0 sec



Type Add, Akb, Auto, Break, Clean, Delete, Grammar, History, Init, Kb, List,
     Load, Log, Makerule, Narrow, Norm, Option, Operator, Prove, Quit, Read,
     Refute, Save, Stats, Suffice, Undo, Unlog, Write or Help.
RRL-> quit
```

20 Timings

Unfortunately the timing facility of the system did not work properly. Nevertheless the reported times in the trace give a hint. Several examples were done in some seconds, most of the others were done in a few minutes. The only exception was example 12, which took almost 3 hours to be completed.

Acknowledgements

My thanks to H. Hong for introducing me to the problems in the area of term rewriting during his course about this subject and for pointing me to the "Rewrite Rule Laboratory". I would also like to thank J. Pfalzgraf for helpful discussions concerning the examples taken from algebra.

References

[Buc83] B. Buchberger. Algebraic Simplification. In B. Buchberger and G. E. Collins and R. Loos, editor, *Computer Algebra*, pages 11–43. Springer Verlag, 1983.

[Che86] Ph. Le Chenadec. *Canonical Forms in Finitely Presented Algebras*. Research Notes in Theoretical Computer Science. Pitman and John Wiley & Sons, 1986.

[Hue80] G. Huet. Equations and Rewrite Rules, A Survey. In R. V. Book, editor, *Formal Language Theory and Open Problems*, pages 349–405. Academic Press, 1980.

[Hul80] J.-M. Hullot. A Catalogue of Canonical Term Rewriting Systems. Technical Report CSL-113, SRI International, April 1980.

[KZ89] D. Kapur and H. Zhang. *RRL: Rewrite Rule Laboratory User's Manual*, 1987 revised 1989.