Appendix A. User Manual for HR1.11

1, 11, 12, 20, 21, 23, 24, 25, 26, 27, 28, 29, 32, 42, 52, 62, 72, 82, ...
A057303. Integers where the number of distinct digits is a digit in base 10.

The latest Prolog version of HR is 1.11. As discussed in Chapter 14, we are presently writing version 2.1 in Java, but discussion of that implementation is beyond the scope of this book. HR 1.11 consists of a set of modules which are loaded into the Sicstus Prolog interpreter, and some auxiliary files such as Unix shell scripts. The user interacts with HR by giving commands which either instruct HR to do something or ask a question about the theory that has been formed.

To describe the commands that are available, we assume the following plan for using HR:

[1] Some settings are specified for the theory formation.

[2] The theory is initialised by providing concepts.

[3] A theory is constructed.

[4] The theory is investigated.

The commands for performing each of these activities are given in sections §A.2 to §A.5. The commands given in the example sessions in Appendix B also highlight how HR is used. There are many individual commands required to use HR. In certain cases we have set up an **interface** where a family of similar commands are called using the same stem for the command. As a theory is being formed, HR calls relevant third party programs to perform various activities and in §A.1 we describe how to get hold of these programs and how to install HR 1.11. In §A.6 we describe the online help available in HR and the demonstrations package which should provide assistance for a new user of HR.

A.1 Installing HR 1.11

We describe how to install HR 1.11 on Unix platforms. For details about Windows installations, please contact `simonco@dai.ed.ac.uk`. Firstly, Sicstus Prolog version 3.5 (`http://www.sics.se/isl/sicstus.html`) is required to run HR 1.11. Unfortunately, the version of Sicstus Prolog must be 3.5, because later versions have a compatibility problem with the Prolog-objects package (and earlier versions may not be compatible). To enable HR to work with later versions of Sicstus would require a major overhaul. Also, the version of Prolog must be that supplied by Sicstus, as we use the Prolog-objects package which is not supported by other Prolog implementations.

The distribution of HR 1.11 is available from here:

`www.dai.ed.ac.uk/~simonco/research/hr/download/hr1p11.tar.gz`

To unpack this, it is first necessary to choose a directory for HR, which we will call `HRPath`. HR must be unpacked into this directory using these Unix commands:

```
gunzip hr1p11.tar.gz
tar -xvf hr1p11.tar
```

This will set up the directory structure as in Figure A.1. The code directory is where the Prolog modules which make up HR are stored. The data directory is where the data files for domains are stored (see §A.3). The modes directory contains mode files (see §A.2) and the scripts directory contains batch files enabling HR to interact with Otter and MACE. We provide the runs directory as a space for running HR sessions.

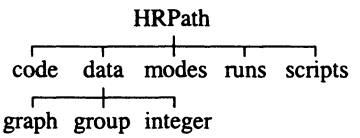


Figure A.1 Directory structure for HR 1.11

It is also necessary to set up a `.hr` file in the home directory of the user. This must contain one line containing the full path name for HR, e.g.

`'/home/user/hruser/hr1p11'.`

Note that the line must be in exactly the above format. Also, the files in the scripts subdirectory have to be made executable with the command:

`chmod 1777 /home/hruser/hr1p11/scripts/*`

(substituting the appropriate path name for HR).

To use the theorem proving and counterexample functionality, the Otter and MACE programs are needed, which can be obtained from here:

```
http://www-unix.mcs.anl.gov/AR/
```

These must be installed in such a way that the Unix command `otter` will call Otter and the command `mace` will call MACE from the runs subdirectory.

HR also uses the Dot program for drawing graphs which is available here:

```
http://www.research.att.com/sw/tools/graphviz
```

and the graph drawing tool GNUplot, which is usually available in Unix, but also available by FTP from here:

```
ftp.dartmouth.edu/pub/gnuplot/gnuplot3.5.tar.Z
```

Once in the `runs` directory, to run HR, the command is:

```
sicstus3p5 -l ../code/hr.pl
```

A.2 Specifying Settings

HR is designed to be highly customisable, with many parameters the user can set to change the way a theory is formed. HR has a `set` interface through which all settings for the theory formation should be specified. Most commands for this interface have the format: `set::parameter(value).` For example, this command:

```
set::complex_max(8).
```

specifies that there is to be a complexity depth limit on the search.

In Table A.1 we list the parameters available via the `set` interface along with details of what they do and the values which can be assigned. In the table, Y/N means that either a "yes" or "no" should be supplied, N means that an integer should be supplied, W means that a word should be supplied and L means that a list should be supplied.

The way in which concepts are sorted can also be stipulated using the `set` interface, by setting weights for the measures used in the overall evaluation of concepts and conjectures. If the user issues the command:

```
set::concept_weights.
```

then HR will list the measures for assessing concepts and ask the user for a weight for each one in turn. Similarly, the command:

```
set::conjecture_weights.
```

will enable setting of the weights for the assessment of conjectures.

Parameter	Explanation
arity_limit(N)	the limit on the arity of the concepts is set to N
axiom_scheme(L1,L2)	L1 is a list of algebra names in which to prove conjectures, with L2 being the weights as explained in §10.2.2
biggest_number(N)	the largest integer that can be introduced in number theory (as explained in §8.3.3)
complex_max(N)	the complexity depth limit is set to N
counterexamples(Y/N)	whether HR attempts to find counterexamples
gold_standard(L)	L is the categorisation of the entities as a list of lists against which the invariance and discrimination of concepts will be measured
integer_limit(N)	N is the largest integer HR can introduce as a counterexample in number theory
keep_conjectures(Y/N)	whether HR keeps the conjectures it makes
mace_time_limit(N)	N is the number of seconds MACE should spend looking for counterexamples at each example size
model_generator(W)	whether HR or MACE should generate examples. W should either be `hr` or `mace`
otter_time_limit(N)	N is the number of seconds Otter should spend attempting proofs
print_style(W)	What information is portrayed to screen during theory formation. W can be either `theory` for the theory or `debug` which describes what HR is doing
prodrules(L)	L is a list of production rule names which HR is to use. The options for the list are: `conjunct`, `common`, `compose`, `exists`, `forall`, `match`, `negate`, `size` and `split`
proof_attack(W)	W is either `subgoal` or `straight`. The former indicates that HR should break conjectures into subgoals before proving them, the latter does not break the conjectures
proofs(Y/N)	whether HR attempts to prove conjectures
report_when(N)	N is the number of steps after which HR presents a report on the theory
search(W)	W is the type of search to be performed. W should be `depth`, `breadth`, `random`, `novelty` or `productivity`
sort_conjectures(N)	whether the conjectures should be sorted
sort_increment(N)	HR sorts its concepts after every N new concepts have been introduced (or conjectures or steps, as specified by sort_marker)
sort_marker(W)	W is either `concept`, `conjecture` or `step` and specifies how HR decides when to sort the concepts
sort_when(N)	N is the number of initial concepts (or conjectures, etc.) before which HR should *not* sort its concepts
split_values(L)	a list of values to which variables can be instantiated by split production rule

Table A.1 The `set` interface

HR records all settings and they can be stored in a file to be read in at the start of later sessions. After specifying some settings and deciding on a name to identify this collection, the user can ask HR to store these as a **mode** in this way:

```
set::save_mode(mode_name).
```

This will save the settings in a file called `mode_name.mod` in the modes subdirectory from the HR path. The command:

```
set::mode(mode_name).
```

will retrieve the mode and HR will list the settings it has altered. In practice, to enable a more fine grained approach, we load more than one mode, each with different settings. To check the settings at any stage, the command is:

```
set::show.
```

To reset all the settings to the defaults, the command is:

```
set::reset_all.
```

The default settings should enable the user to start HR straight away. In the distribution of HR 1.11 we also supply three default modes which have settings specialised for each domain. These are called `number_default`, `graph_default` and `algebra_default` for number theory, graph theory and finite algebraic systems respectively.

A.3 Initialising Theories

As discussed in Chapter 5, there are two ways to give HR the background concepts it requires to start a theory. Firstly, the user can supply the data for concepts in a file with a `.dat` extension. The file must be stored in a subdirectory of the data subdirectory. The subdirectory should have the same name as the domain, e.g. a background file for group theory named `groupbg.dat` should be stored here:

```
HRPath/data/group/groupbg.dat
```

The data files contain only the data tables for the background concepts. All other information about the concept is stored in the `data.pl` file which is found in the code subdirectory. Adding a new concept is time consuming and we hope to provide a simpler interface for this in future. We suggest the user looks at the information in `data.pl` and the data files in the data subdirectories to determine the information required and how to provide it.

We have supplied many background information files in the distribution of HR 1.11. These contain various combinations of concepts, such as divisors, digits and multiplication and various sets of entities such as the numbers 1 to 10 or the numbers 1 to 30, the groups up to order 6 or 8, the complete graphs up to order 4 or 5 and so on. We suggest some experimentation with these background files.

To initialise the theory for domain D using file F, the command is:

```
data(D)::from_file(F).
```

For example, to use the `smalldiv` data file for number theory, which contains the concepts of integers, divisors and multiplication calculated for the numbers 1 to 10, the command is:

```
data(integer)::from_file(smalldiv).
```

Note here that the domain is called `integer` to avoid confusion with the 'number' type columns that the size production rule introduces.

The second way in which the theory can be initialised is by using MACE to generate the concepts from the axioms of a finite algebraic system, as discussed in §5.4. To do this for say, group theory, the command is:

```
data(group)::initialise(mace).
```

We have given HR 1.11 access to 20 different algebraic systems. The domain names (as supplied to HR) are the following:

`galois_field`	`group`
`ip_loop`	`ip_quasigroup`
`loop`	`medial_quasigroup`
`monoid`	`moufang_loop`
`nilpotent_quasigroup`	`qg3_quasigroup`
`qg4_quasigroup`	`qg5_quasigroup`
`qg6_quasigroup`	`qg7_quasigroup`
`quasigroup`	`ring`
`robbins_algebra`	`semigroup`
`trivial`	`ts_quasigroup`

It is not particularly difficult to add a new algebraic system and we suggest the user consult file `data.pl` in the code subdirectory to see how to do this. Note that the command: `restart` will start a new session.

A.4 Constructing Theories

The command to instruct HR to start theory formation has this format:

```
construct(Number, Objects).
```

The user can specify what the finished theory is to contain by stating a type of object, such as a concept or conjecture and the required number of them. For example, given the command:

```
construct(100,concepts).
```

HR will construct a theory until it contains 100 concepts. The list of objects that can be requested is:

categorisations	-	number of different categorisations achieved by the concepts
classifications	-	number of concepts achieving the gold standard categorisation
concepts	-	number of concepts
conjectures	-	number of conjectures (proved, disproved or open)
entities	-	number of entities introduced as counterexamples
open_conjectures	-	number of open conjectures
prime_implicates	-	number of prime implicates from theorems
theorems	-	number of proved conjectures

Alternatively, HR can be asked to construct a theory for a certain length of time with the commands:

```
construct(N,seconds). construct(N,minutes). construct(N,hours).
```

or it can be asked to perform a certain number of theory formation steps:

```
construct(N,steps).
```

A.5 Investigating Theories

We have provided many predicates to enable the user to investigate the theories HR produces. These include a `print` interface to collate results on screen, a `view` interface which presents graphical information, a query mechanism to find concepts or conjectures of a particular nature and a set of predicates to produce additional conjectures.

A.5.1 Printing Results to Screen

The commands in the `print` interface present individual or collated results on screen. There are two different formats for the commands:

```
print::X.   and   print::X(N).
```

where `X` is a property of the theory and `N` is a number. For example,

```
print::concepts.
```

prints definitions for all the concepts to screen. However,

```
print::concept(19).
```

only outputs the definition for concept 19. The `print` commands are summarised by the parameter specified and the information which is printed to screen in Table A.2.

Note that all forms of conjectures are output using their Otter-style definition. Cayley tables are output to enable the user to check calculations. For example, the command:

```
cayley_table('c3').
```

in group theory produces a Cayley table for the group c3:

```
c3
* | 0 | 1 | 2 |
--+---+---+---+
0 | 0 | 1 | 2 |
--+---+---+---+
1 | 1 | 2 | 0 |
--+---+---+---+
2 | 2 | 0 | 1 |
--+---+---+---+
```

A.5.2 Viewing Graphical Information

We have enabled HR to present information graphically using two graph drawing packages. These programs are invoked using the `view` interface.

The Dot program, [Koutsofios & North 98] is a useful tool for drawing graphs which we make extensive use of. To do this, HR writes a Dot-readable file and then invokes Dot to produce a postscript file. HR then displays the postscript file on screen. Our first use of this is to produce diagrams portraying the construction history of a concept, which can often be more informative than the definition alone. We have seen such diagrams throughout the book, in particular in §6.10. To generate a diagram for the construction history of concept number N, the command is:

```
view::construction_history(N).
```

Parameter	Information printed to screen
categorisations	the set of categorisations achieved by the concepts
cayley_tables	the set of Cayley tables (one for each entity in a finite algebraic system)
cayley_table(W)	the Cayley table for entity W
classifications	all concepts achieving the gold standard categorisation
concept(N)	the Otter-style definition for concept N
concepts	the Otter-style definition for every concept
concepts(N)	the Otter-style definition for every concept of arity N
conjecture(N)	the Nth conjecture that was made
conjectures	all the conjectures in the theory
counterexample(N)	the Nth entity that was introduced as a counterexample
facts	the set of fact concepts (as defined in §6.9.1)
missing_number_types(N)	the set of integer sequences in the theory which are missing from the Encyclopedia of Integer Sequences using the integers 1 to N
non_facts	the set of concepts which are not facts
non_theorem(N)	the Nth false conjecture that was made
non_theorems	the set of false conjectures that were made
number_of(W)	W is either `concepts`, `conjectures`, `theorems`, `prime_implicates`, `subgoalsor categorisations`. This counts how many there are
ordered_prime_implicates	the prime implicates ordered by proof length
ordered_theorems	the theorems ordered by proof length
open_conjectures	the set of open conjectures
predicate(N)	the Prolog definition for concept N
predicates	the Prolog definition for every concept
predicates(N)	the Prolog definition for every concept of arity N
prime_implicates	the prime implicates in the theory
report	a report including statistics about the number of concepts, conjectures etc.
subgoal(N)	the Nth subgoal that was introduced
subgoals	the set of subgoals of the conjectures
table(N)	the data table for the Nth concept
tables	the set of data tables for all concepts
theorems	the proved theorems in the theory
tree(N)	the construction history of concept N
weights(N)	the set of weights for measuring concepts and conjectures

Table A.2 The `print` interface

Command	Construction history portrayed to screen
construction_ history	construction of the entire theory (not recommended for large theories)
construction_ history_with_conjectures	construction of the entire theory with conjectures also shown
construction_ history_from(N)	the history of all concepts derived from concept N
construction_ history(complexity,N).	construction of all concepts up to complexity N
construction_ history(first_categorisations)	construction of all concepts which achieved their categorisation first
construction_ history(classifications)	construction of all concepts which achieve the gold standard categorisation

Table A.3 Construction history `view` commands

This can also be used to visualise the construction of conjectures. For example, suppose conjecture 10 is an equivalence conjecture, then the command:

```
view::construction_history(conjecture,10).
```

will produce a construction history for the conjecture, with a dotted line joining the equivalent concepts. There are more ways in which the construction history can be used, as presented in Table A.3.

Our second use of Dot is in graph theory, where we use it to visualise the concepts produced, which can be more revealing than their definitions. The command to produce a diagram for concept number N is:

```
view::gt_concept(N).
```

For example, HR re-invents the concept of graphs with no endpoints (closed graphs), and produces the diagram in Figure A.2 when asked to highlight which of the graphs in its theory have this property. We see that it draws boxes around those graphs with the property. Similarly, for concepts describing properties of nodes or edges, it highlights those nodes/edges which have the property, and for numerical invariants of graphs, it displays the number next to the graph.

If there are more example graphs available to HR than those in the theory (i.e. a set HR can access to find counterexamples to conjectures), then we can also ask HR to look in this set and identify all those graphs of a particular type. The command:

```
view::all_graphs_of_type(N).
```

will produce a diagram similar to the one in Figure A.2 but ranging over the larger set, identifying graphs with the property prescribed by concept `N`.

The final set of commands available with the `view` interface are those which invoke the GNUplot program to draw bar charts and line graphs like

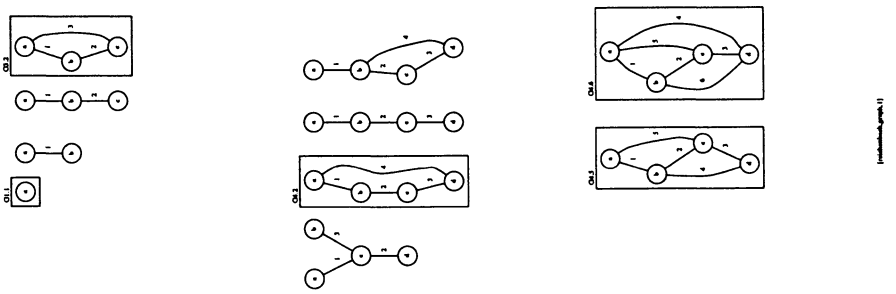


Figure A.2 Concept diagram for closed graphs

those given in Chapter 11. The graphs drawn range over either the concepts, conjectures or theory formation steps, and record a numerical measure of the concept/conjecture. Also, HR can calculate the average over the first n concepts/conjectures and plot this instead of the individual measures.

The commands available are:

```
view::concept_statistics(W).
view::concept_statistics(average,W).
view::conjecture_statistics(W).
view::conjecture_statistics(average,W).
view::step_statistics(W).
view::step_statistics(average,W).
```

In the case of concept statistics, `W` must be a measure of the concepts, namely one of:

```
      applicability          complexity       discrimination
        invariance           total_score   number_of_conjectures
discrimination+invariance
```

We include discrimination+invariance as a measure because this is often of interest when trying to find a concept which achieves a particular categorisation. In each the case of conjecture statistics, the `W` must be a measure of the conjectures, namely one of:

```
proof_length  complexity  applicability  surprisingness
```

For step statistics, any concept or conjecture measure above can be given. As an example, we note that the command:

```
view::concept_statistics(average,complexity).
```

will produce a graph showing the average complexity of the first n concepts.

A.5.3 Finding Concepts and Conjectures

The `concept` interface is also useful for investigating a theory. The commands for this are of the form:

```
concept(C)::property(P).
```

where C is the number of a concept and P is the value of a property of concepts. This can be used in two ways: if C is instantiated to a particular concept number, then HR will find the value P for that concept. For example, the command:

```
concept(10)::arity(A).
```

will return the arity for concept number 10.

However, if C is left uninstantiated, but P is given a value, then HR will attempt to find a concept with that value for property P. For example, the command:

```
concept(X)::arity(3).
```

will return a concept number which has arity 3. Using the ; key, the user can see all such concepts as HR will enumerate them one by one. This functionality is useful for finding concepts of a particular nature in the theory. We often use it in conjunction with the `print` interface. For example, if we wanted to identify a concept with exactly 17 conjectures, we would use the following command:

```
concept(X)::number_of_conjectures(17), print::concept(X).
```

The set of properties for the `concept` interface are given in Table A.4.

Similarly, the `conject` interface is used to find properties of conjectures and find conjectures with a particular property. The general format is:

```
conject(C)::property(P).
```

and the set of properties which are available is given in Table A.5. For example, the command:

```
conject(X)::proof_len(23).
```

will return all theorems proved with a proof length from Otter of 23.

BOOLEAN OUTPUT	
Property	Query returns
classification	whether this concept achieves the gold standard classification
fact	whether this concept is a fact (as defined in §6.9.1)
first_cat	whether this concept achieves its categorisation first
has_conjecture	whether this concept has any conjectures
NUMERICAL OUTPUT	
Property	Query returns
applicability(N)	the applicability of the concept
arity(N)	the arity of the concept
complexity(N)	the complexity score of this concept
conjecture_score(N)	what this concept scores when assessed using the conjectures it appears in
discrimination(N)	the discrimination score for the concept
invariance(N)	the invariance score for the concept
novelty(N)	the novelty score for the concept
number_of_ conjectures(N)	the number of conjectures this concept is involved in
number_of_user_ given_ancestors(N)	the number of user-given concepts from which the concept is derived
parsimony(N)	the parsimony of the concept
productivity(N)	the productivity of the concept
rank(N)	the rank in terms of interestingness of the concept
step_constructed(N)	the theory formation step number when the concept was built
total_score(N)	the overall score calculated for the concept
OTHER CONCEPTS OUTPUT	
Property	Query returns
ancestors(L)	the set of concepts from which this one is built
children(L)	the set of concepts built directly from this one
descendants(L)	the set of concepts with this one their construction path
user_given_ ancestors(L)	the set of user-given concepts from which this concept is built
MISCELLANEOUS OUTPUT	
Property	Query returns
entities(L)	the list of entities to which this concept applies
categorisation(L)	the categorisation produced by the concept
conjectures(L)	the set of conjecture numbers the concept is involved in
history(L)	the construction history of the concept
prodrule_used(W)	the name of the production rule used to construct the concept
table(L)	the data table of the concept
types(L)	the types in the columns of the data table of the concept

Table A.4 The `concept` interface

Property	Query returns
applicability(N)	the applicability of the conjecture
axioms_used(L)	the set of axioms used to prove the conjecture
complexity(N)	the complexity of the conjecture
concepts_involved(L)	the concepts involved in the conjecture
construction(L)	the construction which led to the conjecture
is_otter_compatible	whether the conjecture is in a format acceptable to Otter
measurements(L)	the full list of measurements for this conjecture
proof_length(N)	the length of the proof Otter found for the theorem
proof_status(W)	W is either `disproved`, `max_proofs`, `max_seconds` or `sos`. These are taken from Otter's output. `sos` means Otter has failed and so has MACE
rank(N)	the rank in terms of interestingness of this conjecture
subgoals(L)	the list of subgoals of this conjecture
surprisingness(N)	the surprisingness score for this conjecture
to_concept(N)	the number of the old concept that this equivalence conjecture re-defines
total_score(N)	the overall score for this conjecture
type(W)	W is either `iff` (equivalence), `non-exists`, `implies` or `applies`

Table A.5 The `conject` interface

A.5.4 Making More Conjectures

As discussed in §7.2 and §7.4, HR does not make implication or applicability conjectures during theory formation, but the user can ask for these after a theory has been formed.

The command:

```
applicability_conjectures(N).
```

will produce a set of applicability conjectures by finding concepts which are applicable to N or fewer entities. For example, if working in number theory with the numbers 1 to 30, setting N to 2 in the above command will result in HR producing a set of conjecture statements of the form:

```
C. Definition(C) is satisfied by only [X,Y]
```

where C is a concept which only applies to two integers, namely X and Y.

The command:

```
print::implication_conjectures(lh,C).
```

will produce implication conjectures of the form:

$$X \Rightarrow C$$

by finding concepts, X which have data tables contained in the data table of concept C. Similarly, the command:

```
print::implication_conjectures(rh,C).
```

will produce implication conjectures of the form:

$$C \Rightarrow X$$

However, these often produce too many conjectures, and it is necessary to reduce the number using the surprisingness measure. To do this, the command:

```
set::implication_surprisingness(N).
```

will set the threshold for surprisingness to N. We also supply some commands for using the Encyclopedia of Integer Sequences to generate conjectures about a chosen concept, C, as discussed in §7.5. Firstly, the concept must be of the correct type to be interpreted as a sequence. Then the command:

```
sequence(C)::extend_up_to(N).
```

will extrapolate the sequence from 1 to N. To relate this to sequences in the Encyclopedia, we first need the command:

```
eis::load_sequences.
```

Next the command:

```
eis::assert_new_sequence(C,N).
```

adds sequence C to the (local) Encyclopedia, with C extended up to N. Also the command:

```
eis::details(ANumber).
```

will give the details from the Encyclopedia about sequence `ANumber`. Finally, the command:

```
print::missing_number_types(N).
```

will identify all those sequences in HR's theory which describe types of numbers and are missing from the Encyclopedia (with the sequences calculated between the numbers 1 and `N`).

The commands to find subsequences, disjoint sequences and sequences 'less than' C, are respectively:

```
eis::subsequences_of(C). eis::sequences_disjoint_to(C).
                    eis::sequences_leq(C).
```

Finally, we can prune the output by setting measures using the command:

```
eis::set(Measure,Value).
```

This will set the threshold for `Measure` to be `Value`. At present, the measure can be one of:

`term_overlap_min`	`term_overlap_max`	`density_max`
`density_min`	`number_of_terms_max`	`number_of_terms_min`
`range_overlap_min`	`range_overlap_max`	`difference_min`
`difference_max`		

A.6 Help for a New User

The command: `help::me.` provides a set of four help sections. In each section, HR gives a list of commands for a particular interface and asks the user to choose one of them. It then provides a few sentences about the command, including the syntax and what it does. For example, the help information about the `set::mode` command is the following:

```
### set::mode/1 ###
set::mode(+ModeName).
Allows you to retrieve a list of settings previously
stored using the set::save_mode(+ModeName) command.
```

The help functionality is at present very basic and we hope to improve upon this in future versions of HR.

We have also set up a series of demonstration sessions and an interface for their use. For example, there is a demonstration session for graph theory called `graph_demo1` which can be run as soon as HR has been loaded, using this command:

```
demo(graph_demo1)::go.
```

This will guide the user through the demonstration, pausing after each command has been issued to ask whether the user is ready. Note that the reply should be either `y.` to continue or `n.` to end the demonstration (the full stop is required). Finally, we have a detailed set of web pages about HR, which can be found here:

`http://www.dai.ed.ac.uk/~simonco/research/hr`

Appendix B. Example Sessions

0, 0, 1, 0, 2, 0, 3, 1, 1, 2, 4, 0, 5, 3, 4, 0, 6, 1, 7, 2, 5, 6, 8, 0, ...
A047983. $f(n) = |\{a < n : \tau(a) = \tau(n)\}|$

We present four sessions using HR1.11 in graph theory, group theory, semigroup theory and number theory in §B.1 to §B.4 respectively. For each session, we provide an overview of the functionality we wish to highlight, followed by the session output and a commentary on some of the more interesting events which occurred in the session. Each session is annotated with letters in the left hand margin and the commentary discusses events occurring where the letters are placed. In the commentary, we provide page numbers relating events in the session to the relevant text in the book body and/or the manual.

Due to space considerations, we have edited the session output to condense the text. This has mostly involved removing entire sections of the session output. The edited sessions are still detailed enough to give a good sense of what HR is doing, and for each section removed, we provide a note in the commentary describing what was removed. The entire unedited sessions are available here:

`http://www.dai.ed.ac.uk/~simonco/research/thesis/appendixb`

Unfortunately, again due to space limitations, we cannot provide sessions showing all of HR's functionality. In the session described in §B.1, we use graph theory to show some general features of HR. In §B.2, we show the cycle of mathematical activity that HR performs. In §B.3, we highlight more features of theory formation, including HR's forward chaining mechanism to prove theorems and the use of counterexamples to disprove old conjectures. Finally, in §B.4, we provide a session in number theory, where the Encyclopedia of Integer Sequences is used to provide an interesting conjecture about an integer sequence that HR invents. To run these sessions, type

```
demo(demo_name)::go.
```

at HR's prompt (where `demo_name` is one of `graph_demo1`, `group_demo1`, `semigroup_demo1` or `number_demo1`).

B.1 Graph Theory Short Session

We use this session to illustrate some of the basic commands for starting a session and investigating the output. We also explain some of the terms in HR's output. Finally, it provides an opportunity to show how HR produces graphical representations to help illustrate the concepts. The commands we used for this session were the following:

```
1. set::mode(graph_default).
2. set::concept_weight(comprehensibility,0.8).
3. set::concept_weight(productivity,0.0).
4. set::concept_weight(novelty,0.2).
5. data(graph)::from_file(connected_graph).
6. construct(100,concepts).
7. print::entity_types.
8. view::gt_concept(95).
9. view::all_graphs_of_type(95).
10. view::construction_history(95).
```

B.1.1 Session Output

```
  SICStus 3 : #5: 1996 Oct 15
  HR1.11 is loaded. Please type help::me. for help.

  yes
A | ?- set::mode(graph_default).
  counterexamples=[no]
  keep_conjectures=[no]
  sort_conjectures=[no]
  prodrules=[[exists,match,forall,conjunct,size,split,negate,common]]
  complex_max=[8]
  concept_weight=[comprehensibility,0.2]
  concept_weight=[novelty,0.6]
  concept_weight=[productivity,0.2]
  sort_concepts=[yes]
  split_values=[[1,2]]

  yes
B | ?- set::concept_weight(comprehensibility,0.8).
  concept_weight=[comprehensibility,0.8]

  yes
  | ?- set::concept_weight(productivity,0.0).
  concept_weight=[productivity,0.0]

  yes
  | ?- set::concept_weight(novelty,0.2).
  concept_weight=[novelty,0.2]

  yes
C | ?- data(graph)::from_file(connected_graph).

  1. graph
  2. node
  3. edge
  4. edge_node

  yes
D | ?- construct(100,concepts).
E (5) [G,N] : N = |{n1 : node(n1)}|
F (6) [G] : (exists e1 (edge(e1)))
  (7) [G,N] : N = |{e1 : edge(e1)}|
  (8) [G,n1] : node(n1) & (exists e1 (edge(e1)))
  (9) [G,e1] : (all n2 (n2 is on e1))
  (10) [G,n1] : (all e2 (n1 is on e2))

G Top 20 concepts: 4 3 2 6 7 5 8 9 10
```

```
Top 20 live concepts: 4(1) 3 2 7 5 8 9 10
Sorted Production Rules: common conjunct exists forall match negate size split

(11) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1)
(12) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2)
(13) [G,e1,n1] : n1 is on e1 & (all n3 (n3 is on e1))
(14) [G,e1,n1] : n1 is on e1 & (all e3 (n1 is on e3))
(15) [G,N] : N = |{(e1 n1) : n1 is on e1}|
(16) [G,e1,N] : N = |{n1 : n1 is on e1}|
(17) [G,n1,N] : N = |{e1 : n1 is on e1}|
(18) [G] : 1 = |{e1 : edge(e1)}|
(19) [G] : 2 = |{e1 : edge(e1)}|
(20) [G,e1,n1] : n1 is on e1 & 2 = |{e2 : edge(e2)}|

Top 20 concepts: 2 3 4 6 5 8 7 11 12 15 16 17 10 19 9 18 14 13 20
Top 20 live concepts: 2(1) 3 4 6 5 8 7 11 12 15 16 17 10 19 9 18 14 13 20
Sorted Production Rules: common conjunct exists forall match negate size split

(21) [G,n1] : node(n1) & 2 = |{e1 : edge(e1)}|
(22) [G,n1] : node(n1) & 1 = |{e1 : edge(e1)}|
(23) [G,e1] : edge(e1) & 2 = |{e2 : edge(e2)}|
(24) [G,N] : N = |{n1 : node(n1)}| & N = |{e1 : edge(e1)}|
(25) [G,N] : N = |{n1 : node(n1)}| & N = |{(e1 n2) : n2 is on e1}|
(26) [G,N] : N = |{n1 : node(n1)}| & 2 = |{e1 : edge(e1)}|
(27) [G] : 1 = |{n1 : node(n1)}|
(28) [G,n1] : node(n1) & 1 = |{n2 : node(n2)}|
(29) [G,N] : N = |{n1 : node(n1)}| & 1 = |{n2 : node(n2)}|
(30) [G,n1,n2] : ((exists e1 (n1 is on e1)) & (exists e2 (n2 is on e2)))

Top 20 concepts: 2 3 4 5 6 7 8 11 12 15 16 17 10 27 30 9 18 19 14 28
H Top 20 live concepts: 4(3) 7 8 11 12 15 16 17 10 27 30 9 18 19 14 28
                        29 24 13 23
Sorted Production Rules: common conjunct exists forall match negate size split

(31) [G,N] : N = |{e1 : edge(e1)}| & 1 = |{e2 : edge(e2)}|
(32) [G,N] : N = |{e1 : edge(e1)}| & 2 = |{e2 : edge(e2)}|
(33) [G,N] : N = |{n1 : (exists e1 (n1 is on e1))}|
(34) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & (all n4 (n4 is on e1))
(35) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & 2 = |{e2 : edge(e2)}|
I (36) [G,n1,n2] : (exists e1 ((n1 is on e1 & n2 is on e1)))
(37) [G,n1,n2] : (all e2 ((n1 is on e2 & n2 is on e2)))
(38) [G,N] : N = |{(e1 n1 n2) : (n1 is on e1 & n2 is on e1)}|
(39) [G,e1,N] : N = |{(n1 n2) : (n1 is on e1 & n2 is on e1)}|
(40) [G,e1,n1,N] : N = |{n2 : (n1 is on e1 & n2 is on e1)}|

Top 20 concepts: 2 3 4 6 5 8 7 11 12 15 16 17 10 27 28 30 33 36 38 39
Top 20 live concepts: 4(3) 5 7 11 12 15 16 17 10 27 28 30 33 36 38 39
                      40 14 37 9
Sorted Production Rules: common conjunct exists forall match negate size split

(41) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1)
                    & (all e3 ((n1 is on e3 & n2 is on e3)))
(42) [G,n1,N] : N = |{(e1 n2) : (n1 is on e1 & n2 is on e1)}|
(43) [G,n1,n2,N] : N = |{e1 : (n1 is on e1 & n2 is on e1)}|
(44) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2) & (all e4 (n1 is on e4))
(45) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2) & 1 = |{e3 : edge(e3)}|
(46) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2) & 2 = |{e3 : edge(e3)}|
(47) [G,e1,e2] : (exists n1 ((n1 is on e1 & n1 is on e2)))
(48) [G,e1,e2] : (all n2 ((n2 is on e1 & n2 is on e2)))
(49) [G,N] : N = |{(e1 e2 n1) : (n1 is on e1 & n1 is on e2)}|
(50) [G,e1,N] : N = |{(e2 n1) : (n1 is on e1 & n1 is on e2)}|

Top 20 concepts: 2 3 4 6 5 8 7 15 16 11 12 17 47 49 50 27 28 10 30 33
Top 20 live concepts: 4(3) 5 7 15 16 11 12 17 47 49 50 27 28 10 30 33
                      38 39 19 36
Sorted Production Rules: common conjunct exists forall match negate size split

(51) [G,N] : N = |{(e1 n1) : n1 is on e1}| & 2 = |{e2 : edge(e2)}|
(52) [G,e1,e2,N] : (N = |{n1 : n1 is on e1}| & N = |{n2 : n2 is on e2}|)
(53) [G,e1,N] : N = |{n1 : n1 is on e1}| & N = |{n2 : node(n2)}|
(54) [G,e1,N] : N = |{n1 : n1 is on e1}| & N = |{e2 : edge(e2)}|
(55) [G,N] : (all e2 (N = |{n1 : n1 is on e2}|))
(56) [G,N,N] : N = |{e1 : N = |{n1 : n1 is on e1}|}|
(57) [G,e1,e2,N] : N = |{n1 : (n1 is on e1 & n1 is on e2)}|
(58) [G,e1,n1,N] : N = |{e2 : (n1 is on e1 & n1 is on e2)}|
(59) [G,n1,N] : N = |{(e1 e2) : (n1 is on e1 & n1 is on e2)}|
(60) [G,n1,N] : N = |{e1 : n1 is on e1}| & N = |{e2 : edge(e2)}|

Top 20 concepts: 2 3 4 5 6 8 7 15 16 11 12 17 47 49 50 57 27 28 30 33
Top 20 live concepts: 17(12) 47 49 50 57 27 28 30 33 10 38 39 52 56 9
                      18 19 36 40 42
Sorted Production Rules: common conjunct exists forall match negate size split

(61) [G,n1,N] : N = |{e1 : n1 is on e1}| & 2 = |{e2 : edge(e2)}|
(62) [G,n1,N] : N = |{e1 : n1 is on e1}|
                    & N = |{(e2 e3) : (n1 is on e2 & n1 is on e3)}|
(63) [G,n1,N] : N = |{e1 : n1 is on e1}| & (all e3 (N = |{n2 : n2 is on e3}|))
(64) [G,N] : (all n2 (N = |{e1 : n2 is on e1}|))
```

```
  (65) [G,n1,N] : N = |{M : M = |{e1 : n1 is on e1}|}|
  (66) [G,n1] : 1 = |{e1 : n1 is on e1}|
  (67) [G,n1] : 2 = |{e1 : n1 is on e1}|
  (68) [G,e1,n1] : n1 is on e1 & 1 = |{e2 : n1 is on e2}|
  (69) [G,e1,n1] : n1 is on e1 & 2 = |{e2 : n1 is on e2}|
  (70) [G,e1,N] : N = |{n1 : n1 is on e1}| & (all n3 (N = |{e2 : n3 is on e2}|))

  Top 20 concepts: 2 3 4 6 5 8 7 15 16 11 12 17 66 67 47 49 50 57 27 28
  Top 20 live concepts: 16(9) 11 12 17 66 67 47 49 50 57 27 28 10 30 33
                        65 38 39 52 56
  Sorted Production Rules: common conjunct exists forall match negate size split

  (71) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & 1 = |{e2 : n1 is on e2}|
  (72) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & 1 = |{e2 : n2 is on e2}|
  (73) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & 2 = |{e2 : n1 is on e2}|
  (74) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1) & 2 = |{e2 : n2 is on e2}|
  (75) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2) & 1 = |{e3 : n1 is on e3}|
  (76) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2) & 2 = |{e3 : n1 is on e3}|
  (77) [G,n1,N] : N = |{e1 : n1 is on e1}| & (all n3 (N = |{e2 : n3 is on e2}|))
  (78) [G,n1,n2] : (1 = |{e1 : n1 is on e1}| & 1 = |{e2 : n2 is on e2}|)
  (79) [G,n1] : 1 = |{e1 : n1 is on e1}| & 2 = |{e2 : edge(e2)}|
J (80) [G] : (exists n1 (1 = |{e1 : n1 is on e1}|))

  Top 20 concepts: 2 3 4 6 5 8 7 15 16 47 49 50 57 11 12 17 27 28 10 30
  Top 20 live concepts: 2(1) 3 4 6 5 8 7 15 16 47 49 50 57 11 12 17 27
                        28 10 30
  Sorted Production Rules: common conjunct exists forall match negate size split

  (81) [G,n1] : node(n1) & (exists n2 (1 = |{e1 : n2 is on e1}|))
  (82) [G,e1] : edge(e1) & (exists n1 (1 = |{e2 : n1 is on e2}|))
  (83) [G,e1,n1] : n1 is on e1 & (exists n2 (1 = |{e2 : n2 is on e2}|))
  (84) [G,N] : N = |{n1 : node(n1)}| & (exists n2 (1 = |{e1 : n2 is on e1}|))
  (85) [G,N] : N = |{e1 : edge(e1)}| & (exists n1 (1 = |{e2 : n1 is on e2}|))
  (86) [G,N] : N = |{(e1 n1) : n1 is on e1}|
               & (exists n2 (1 = |{e2 : n2 is on e2}|))
  (87) [G,e1,N] : N = |{n1 : n1 is on e1}|
                  & (exists n2 (1 = |{e2 : n2 is on e2}|))
  (88) [G,e1,e2,e3] : ((exists n1 ((n1 is on e1 & n1 is on e2)))
                       & (exists n2 ((n2 is on e1 & n2 is on e3))))
  (89) [G,e1,e2,e3] : ((exists n1 ((n1 is on e1 & n1 is on e3)))
                       & (exists n2 ((n2 is on e2 & n2 is on e3))))
  (90) [G,e1,e2] : (exists n1 ((n1 is on e1 & n1 is on e2)))
                   & (exists n2 (1 = |{e3 : n2 is on e3}|))

  Top 20 concepts: 2 3 4 6 5 8 7 15 16 11 12 17 27 28 10 30 33 47 49 50
  Top 20 live concepts: 11(10) 12 17 27 28 10 30 33 47 49 50 57 65 67 38
                        39 52 56 64 66
  Sorted Production Rules: common conjunct exists forall match negate size split

  (91) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1)
                      & (exists n3 (1 = |{e2 : n3 is on e2}|))
  (92) [G,e1,n1,n2] : (n1 is on e1 & n2 is on e1)
                      & (1 = |{e2 : n1 is on e2}| & 1 = |{e3 : n2 is on e3}|)
  (93) [G,e1,e2,n1] : (n1 is on e1 & n1 is on e2)
                      & (exists n2 (1 = |{e3 : n2 is on e3}|))
  (94) [G,n1] : (all e2 (n1 is on e2)) & 2 = |{e3 : n1 is on e3}|
K (95) [G] : (exists n1 ((all e2 (n1 is on e2))))
  (96) [G,N] : N = |{n1 : (all e2 (n1 is on e2))}|
  (97) [G,e1] : edge(e1) & (exists n1 ((all e3 (n1 is on e3))))
  (98) [G,e1,n1] : n1 is on e1 & (exists n2 ((all e3 (n2 is on e3))))
  (99) [G,N] : N = |{n1 : node(n1)}| & (exists n2 ((all e2 (n2 is on e2))))
  (100) [G,n1] : (exists e1 (n1 is on e1)) & (exists n2 ((all e3 (n2 is on e3))))

  Top 20 concepts: 2 3 4 6 5 8 7 15 16 11 12 17 27 28 30 33 47 49 50 57
  Top 20 live concepts: 8(6) 7 15 16 11 12 17 27 28 30 33 47 49 50 57 65
                        67 38 39 52
  Sorted Production Rules: common conjunct exists forall match negate size split

  yes
L | ?- print::entity_types.
  1 [G] : graph(G)
  6 [G] : (exists e1 (edge(e1)))
  18 [G] : 1 = |{e1 : edge(e1)}|
  19 [G] : 2 = |{e1 : edge(e1)}|
  27 [G] : 1 = |{n1 : node(n1)}|
  80 [G] : (exists n1 (1 = |{e1 : n1 is on e1}|))
  95 [G] : (exists n1 ((all e2 (n1 is on e2))))

  yes
M | ?- view::gt_concept(95).

  yes
N | ?- view::all_graphs_of_type(95).

  yes
O | ?- view::construction_history(95).

  yes
```

B.1.2 Commentary

Event	Description	See pages
A	HR loads and the user selects the default settings for graph theory.	307
B	The user changes the weights for the weighted sum of interestingness measures. The new settings prefer more comprehensible concepts.	158, 306
C	The user chooses graph theory and loads the data from the `connected_graph` file. This file contains the connected graphs up to size 4 and 4 concepts, namely `graph`, `node`, `edge` and `edge_node`.	62, 307
D	The user asks HR to construct a theory containing 100 concepts.	309
E	HR invents the concept of the number of nodes of a graph using the size production rule.	81
F	HR distinguishes between the trivial graphs (with one node and no edges) and the other graphs.	
G	HR sorts the concepts for the first time. The earlier concepts are more comprehensible, hence they are more interesting with respect to the weights set by the user.	155
H	The term "live concepts" means those concepts which have theory formation steps remaining (i.e. those which can be built from). For some concepts, all the steps may be exhausted, hence they are not live. The notation 4(3) shows that 4 is the most interesting live concept, but this is the third most interesting overall.	
I	HR re-invents the concept of adjacency of nodes in a graph.	32
J	HR re-invents the concept of graphs with an endpoint.	32

Event	Description	See pages
K	HR re-invents the concept of star graphs, for which there exists a node which is on every edge.	32
L	The user asks HR to display all the concepts which are types of graphs. There are seven such concepts.	310
M	The user asks for a graphical representation of concept 95. The diagram produced, as shown in Figure B.2 identifies which of the 10 graphs have the property prescribed by concept 95.	310
N	The user asks for a similar diagram as before, but this time using all the graphs up to size 6 (which HR has stored and uses for finding counterexamples to false conjectures, a functionality not shown in this simple session). The diagram produced, as shown in Figure B.3 helps identify that the concept describes star graphs.	310
O	The user asks for a diagram of the construction history of concept 95. The diagram produced is given in Figure B.1.	98, 310

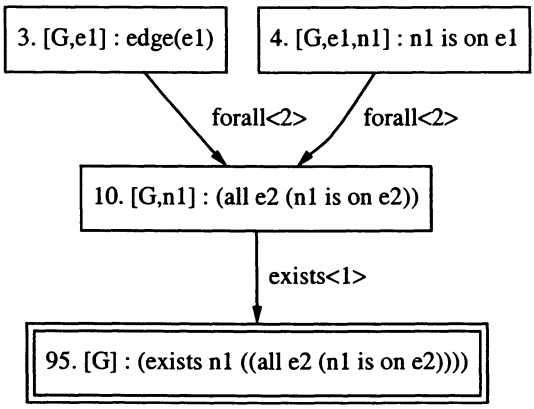


Figure B.1 Construction history of concept 95

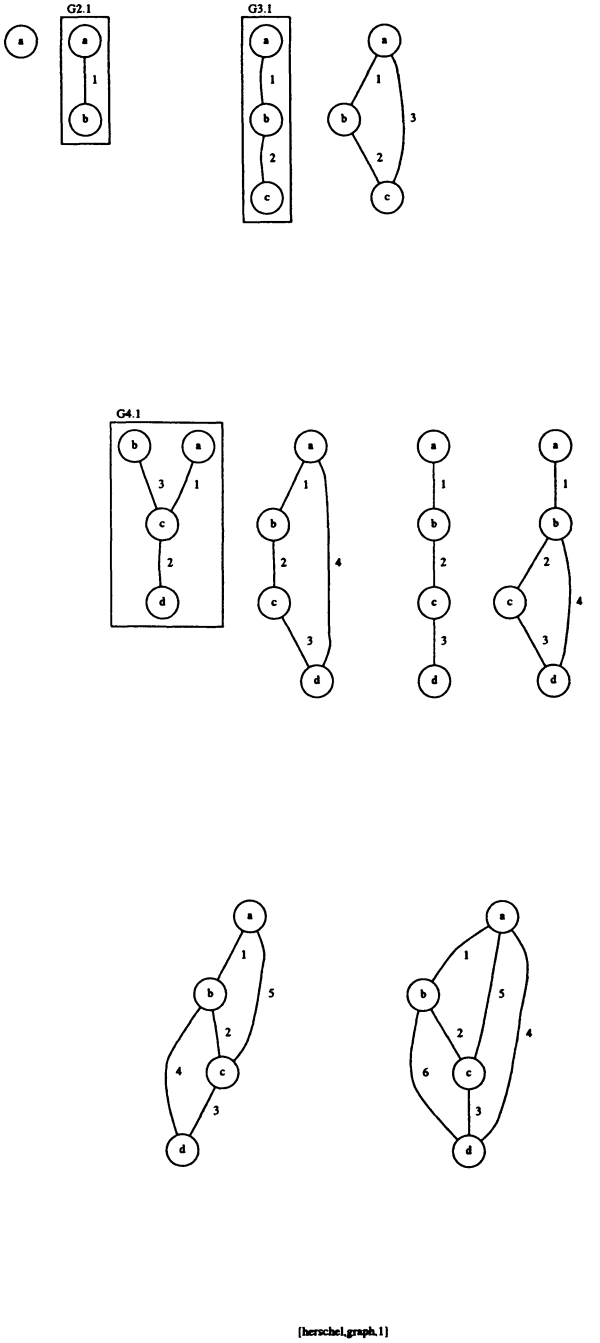


Figure B.2 Graphs up to size 5 described by concept 95 (the boxed graphs have the property defined by concept 95)

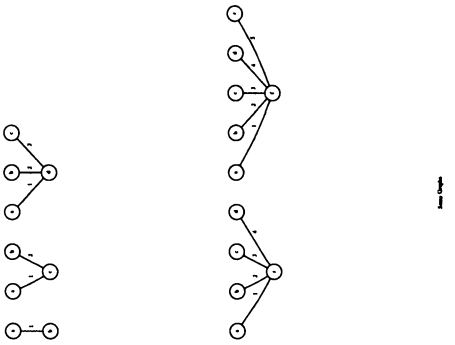


Figure B.3 Graphs up to size 6 described by concept 95

B.2 Theory Formation Session in Group Theory

In this session, we wished to highlight the cycle of mathematical activity that HR undertakes, whereby concepts are formed and conjectures made about the concepts. Then, information arising from attempts to settle the conjectures is used to re-assess the concepts, which drives the heuristic search. We chose group theory as this is the domain for which theory formation was originally developed. At the end of the session, the user investigates the concept which HR has assessed as the most interesting, and Otter is given more time to prove a conjecture which remains open.

The commands we used for this session were the following:

```
1. set::mode(algebra_default).
2. set::proof_attack(straight).
3. data(group)::initialise(mace).
4. construct(100,steps).
5. print::concept(28).
6. concept(28)::conjectures(A).
7. conject(38)::proof_len(A), conject(45)::proof_len(B).
8. conject(49)::proof_len(A).
9. conject(49)::surprisingness(A).
10. print::conjecture(49).
11. print::table(28).
12. print::cayley_table(1,'G04').
13. print::ordered_theorems.
14. print::open_conjectures.
15. set::otter_time_limit(600).
16. print::time, conject(46)::prove, print::time.
```

B.2.1 Session Output

```
   SICStus 3  #5: Tue Aug 26 10:14:51 BST 1997
   HR1.11 is loaded. Please type help::me. for help.

A  | ?- set::mode(algebra_default).
   sort_concepts=[yes]
   sort_conjectures=[yes]
   model_generator=[mace]
   prodrules=[[exists,forall,match,negate,conjunct]]
   complex_max=[6]
   concept_weight=[applicability,0.2]
   concept_weight=[novelty,0.3]
   concept_weight=[productivity,0.2]
   concept_weight=[theorem,0.3]
   theorem_weight=[proof_length,0.5]
   theorem_weight=[surprisingness,0.5]
   initial_concepts=[[element,pair,triple]]
   sort_marker=[step]

   yes
B  | ?- set::proof_attack(straight).
   yes
C  | ?- data(group)::initialise(mace).
   Mace Generating First Model of Order: 1 G01
   all ax1 ax2 ax3 associativity: (ax1 * (ax2 * ax3) = (ax1 * ax2) * ax3).
   all ax1 identity: (ax1*id=ax1 & id*ax1=ax1).
   all ax1 inverse: (inv(ax1)*ax1=id & ax1*inv(ax1)=id).
D  1. *
   2. id
   3. inv
   4. group
   5. element
   6. pair
   7. triple

E  * | 0 |
   --+---+
   0 | 0 |
   --+---+

   id: 0,

   inv|0
      |0

   element: 0,

   yes
F  | ?- construct(100,steps).
G  1. (exists a b c (a*b=c)). max_proofs 2

H  2. all a ((a=id) <-> ((exists b c (a*b=c)))). sos

   Mace generating counterexample of order: 1 2 G02
   Checking whether conjectures are disproved: 2:yes(New Conjecture = 3),
   * | 0 | 1 |
   --+---+---+
   0 | 0 | 1 |
   --+---+---+
   1 | 1 | 0 |
   --+---+---+

   id: 0,

   inv|0|1
      |0|1

   element: 0,1,

   3. all a ((exists b c (a*b=c))). max_proofs 2
   4. all a b ((exists c (a*b=c))). max_proofs 1
I  5. all a b ((exists c (a*c=b))). max_proofs 2
   6. all a ((exists b c (b*a=c))). max_proofs 1
   7. all a b ((exists c (c*a=b))). max_proofs 4
   8. all a ((exists b c (b*c=a))). max_proofs 0
   9. all a ((a=id) <-> (a*a=a)). max_proofs 8
   (8) [G,a,b] : a*a=b
   10. all a b ((a*a=b) <-> (a*b=a)). max_seconds

   Mace generating counterexample of order: 1 2 3 G03
   Checking whether conjectures are disproved: 10:yes(New Concept = 9),
   (9) [G,a,b] : a*b=a

   * | 0 | 1 | 2 |
   --+---+---+---+
   0 | 0 | 1 | 2 |
   --+---+---+---+
```

```
1 | 1 | 2 | 0 |
--+---+---+---+
2 | 2 | 0 | 1 |
--+---+---+---+

id: 0,

inv|0|1|2
   |0|2|1

element: 0,1,2,

Top 20 concepts: 2 5 1 3 8 9
Top 20 live concepts: 2(1) 5 1 3 8 9
Top 20 non theorems: 10 2
Top 20 theorems: 9 7 5 3 1 6 4 8
Sorted Production Rules: conjunct exists forall match negate

11. (exists a (a=id)). max_proofs 0

(10) [G,a] : -(a=id)
(11) [G,a,b,c] : a*b=c & a=id
(12) [G,a,b,c] : a*b=c & b=id
(13) [G,a,b,c] : a*b=c & c=id

Top 20 concepts: 2 1 3 8 9 11 12 13 5 10
Top 20 live concepts: 1(2) 3 8 9 11 12 13 10
Top 20 non theorems: 10 2
Top 20 theorems: 9 7 5 3 1 6 4 11 8
Sorted Production Rules: conjunct exists forall match negate

(14) [G,a,b,c] : a*b=c & a*c=b
J 12. all a b c ((a*b=c) <-> (a*b=c & b*a=c)). sos

Mace generating counterexample of order: 1 2 3 4 5 6 GO4
Checking whether conjectures are disproved: 12:yes(New Concept = 15),
(15) [G,a,b,c] : a*b=c & b*a=c

* | 0 | 1 | 2 | 3 | 4 | 5 |
--+---+---+---+---+---+---+
0 | 0 | 1 | 2 | 3 | 4 | 5 |
--+---+---+---+---+---+---+
1 | 1 | 0 | 3 | 2 | 5 | 4 |
--+---+---+---+---+---+---+
2 | 2 | 4 | 0 | 5 | 1 | 3 |
--+---+---+---+---+---+---+
3 | 3 | 5 | 1 | 4 | 0 | 2 |
--+---+---+---+---+---+---+
4 | 4 | 2 | 5 | 0 | 3 | 1 |
--+---+---+---+---+---+---+
5 | 5 | 3 | 4 | 1 | 2 | 0 |
--+---+---+---+---+---+---+

id: 0,

inv|0|1|2|3|4|5
   |0|1|2|4|3|5

element: 0,1,2,3,4,5,

(16) [G,a,b,c] : a*b=c & c*a=b
(17) [G,a,b,c] : a*b=c & b*c=a
(18) [G,a,b,c] : a*b=c & c*b=a
13. all a b c ((a*b=c & c=id) <-> (a*b=c & b=inv(a))). max_proofs 10
(19) [G,a,b,c] : a*b=c & c=inv(a)
14. all a b c ((a*b=c & c=id) <-> (a*b=c & a=inv(b))). max_proofs 14
15. all a b c ((a*b=c & c=inv(a)) <-> (a*b=c & a=inv(c))). max_proofs 9
(20) [G,a,b,c] : a*b=c & c=inv(b)

Top 20 concepts: 2 13 19 3 1 8 9 11 12 14 15 16 17 18 20 5 10
Top 20 live concepts: 13(2) 19 3 1 8 9 11 12 14 15 16 17 18 20 10
Top 20 non theorems: 12 10 2
Top 20 theorems: 14 13 15 9 7 5 3 1 6 4 11 8
Sorted Production Rules: conjunct exists forall match negate

(21) [G,a,b,c] : a*b=c & c=id & a=id
16. all a b c ((a*b=c & c=id & a=id) <-> (a*b=c & c=id & b=id)). max_proofs 9
17. all a b c ((a*b=c & c=id) <-> (a*b=c & c=id & b=inv(a))). max_proofs 14
18. all a b c ((a*b=c & c=id & a=id) <->
    (a*b=c & c=id & c=inv(a))). max_proofs 10
19. all a b c ((a*b=c & c=id) <-> (a*b=c & c=id & a=inv(b))). max_proofs 11
20. all a b c ((a*b=c & c=id & a=id) <->
    (a*b=c & c=id & a=inv(c))). max_proofs 9
21. all a b c ((a*b=c & c=id & a=id) <->
    (a*b=c & c=id & c=inv(b))). max_proofs 12
22. all a b c ((a*b=c & c=id & a=id) <->
    (a*b=c & c=id & b=inv(c))). max_proofs 13
(22) [G,a,b,c] : a*b=c & c=id & a*a=b
```

```
  (23) [G,a,b,c] : a*b=c & c=id & a*a=c
K .
  .
  59. -(exists a b c (a*b=c & c=id & a=id & -(b=id))). max_proofs 6
  60. (exists a b c (a*b=c & c=id & a=id)). max_proofs 0
  61. all a ((a=id) <-> ((exists b c (a*b=c & c=id & a=id)))). max_proofs 8
  (35) [G,a,b] : (exists c (a*b=c & c=id & a=id))
  62. all a b (((exists c (a*b=c & c=id & a=id))) <->
      ((exists d (a*d=b & b=id & a=id)))). max_proofs 14
  63. all a ((a=id) <-> ((exists b c (b*a=c & c=id & b=id)))). max_proofs 13
  64. all a b (((exists c (a*b=c & c=id & a=id))) <-> (
      (exists d (d*a=b & b=id & d=id)))). max_seconds

L Mace generating counterexample of order: 1 2 3 4 5 6 7 8
  65. all a ((a=id) <-> ((exists b c (b*c=a & a=id & b=id)))). max_proofs 8
  (36) [G] : (all a b c (a*b=c & c=id & a=id))
  (37) [G,a] : (all b c (a*b=c & c=id & a=id))

M Top 20 concepts: 28 26 35 22 2 23 30 24 25 32 29 33 36 37 21 19 1 13 11 12
  Top 20 live concepts: 35(3) 21 19 1 13 11 12 14 15 16 17 18 20 3 8 9 10
  Top 20 non theorems: 12 10 2
  Top 20 open conjectures: 64 48 47 46 45 38 34
  Top 20 theorems: 49 63 40 43 41 36 14 65 61 42 35 37 33 22 44 13 21 62 17 15
  Sorted Production Rules: conjunct exists forall match negate

  yes
  | ?- print::concept(28).
  [G,a] : (exists b c (a*b=c & c=id & a*a=c))
  yes
  | ?- concept(28)::conjectures(A).
  A = [38,45,49] ?
  yes
N | ?- conject(38)::proof_len(A), conject(45)::proof_len(B).
  A = 0 ? B = 0 ?
  yes
  | ?- conject(49)::proof_len(A).
  A = 32 ?
  yes
  | ?- conject(49)::surprisingness(A).
  A = 6 ?
  yes
  | ?- print::conjecture(49).
  all a (((exists b c (a*b=c & c=id & a*a=c))) <->
  ((exists d e (d*e=a & a=inv(d) & e=id)))).
  yes
O | ?- print::table(28).
  | G01 | 0 || G02 | 0 || G02 | 1 || G03 | 0 || G04 | 0 || G04 | 1 |
  | G04 | 2 || G04 | 5 |
  yes
  | ?- print::cayley_table(1,'G04').

  * | 0 | 1 | 2 | 3 | 4 | 5 |
  --+---+---+---+---+---+---+
  0 | 0 | 1 | 2 | 3 | 4 | 5 |
  --+---+---+---+---+---+---+
  1 | 1 | 0 | 3 | 2 | 5 | 4 |
  --+---+---+---+---+---+---+
  2 | 2 | 4 | 0 | 5 | 1 | 3 |
  --+---+---+---+---+---+---+
  3 | 3 | 5 | 1 | 4 | 0 | 2 |
  --+---+---+---+---+---+---+
  4 | 4 | 2 | 5 | 0 | 3 | 1 |
  --+---+---+---+---+---+---+
  5 | 5 | 3 | 4 | 1 | 2 | 0 |
  --+---+---+---+---+---+---+

  yes
P | ?- print::ordered_theorems.

  35. all a b (((exists c (a*c=b & b=id & a*a=c))) <->
      ((exists d (d*a=b & b=id & d*d=a)))). 37
  49. all a (((exists b c (a*b=c & c=id & a*a=c))) <->
      ((exists d e (d*e=a & a=inv(d) & e=id)))). 32
  43. all a b c ((a*b=c & c=inv(a) & b=id) <->
      (a*b=c & c=inv(a) & a*b=a)). 15
  62. all a b (((exists c (a*b=c & c=id & a=id))) <->
      ((exists d (a*d=b & b=id & a=id)))). 14
  17. all a b c ((a*b=c & c=id) <-> (a*b=c & c=id & b=inv(a))). 14
  14. all a b c ((a*b=c & c=id) <-> (a*b=c & a=inv(b))). 14
  .
  .
  37. (exists a b c (a*b=c & c=id & a*a=c)). 1
  33. (exists a b c (a*b=c & c=id & a*a=b)). 1
  6. all a ((exists b c (b*a=c))). 1
  4. all a b ((exists c (a*b=c))). 1
  60. (exists a b c (a*b=c & c=id & a=id)). 0
  44. (exists a b c (a*b=c & c=inv(a) & b=id)). 0
  11. (exists a (a=id)). 0
```

```
  8. all a ((exists b c (b*c=a))). 0

  yes
Q | ?- print::open_conjectures.

  34. all a (((exists b c (a*b=c & c=id & a*a=b))) <->
      ((exists d e (d*a=e & e=id & d*d=a)))).
  38. all a (((exists b c (a*b=c & c=id & a*a=c))) <->
      ((exists d e (d*a=e & e=id & d*d=e)))).
  45. all a (((exists b c (a*b=c & c=id & a*a=c))) <->
      ((exists d e (a*d=e & e=inv(a) & d=id)))).
  46. all a b (((exists c (a*c=b & b=id & a*a=b))) <->
      ((exists d (a*b=d & d=inv(a) & b=id)))).
  47. all a b (((exists c (a*b=c & c=id & a*a=c))) <->
      ((exists d (a*d=b & b=inv(a) & d=id)))).
  48. all a ((a=id) <-> ((exists b c (b*a=c & c=inv(b) & a=id)))).
  64. all a b (((exists c (a*b=c & c=id & a=id))) <->
      ((exists d (d*a=b & b=id & d=id)))).
  yes
R | ?- set::otter_time_limit(600).
  yes
  | ?- print::time, conject(46)::prove, print::time.
  Wed Jul 26 12:00:57 BST 2000
  46. all a b (((exists c (a*c=b & b=id & a*a=b))) <->
      ((exists d (a*b=d & d=inv(a) & b=id)))). max_proofs 63
  Wed Jul 26 12:01:59 BST 2000
  yes
```

B.2.2 Commentary

Event	Description	See pages
A	HR loads and the user selects the default algebra settings.	307
B	The user instructs HR not to break the conjectures into subgoals, just to use Otter to prove the theorem straight away.	122, 306
C	The user chooses group theory and asks HR to use MACE to construct a single group.	66, 307
D	HR extracts 4 concepts (`*`, `id`, `inv` and `group`) from MACE's output and adds three others: elements, pairs of elements and triples of elements.	66, 307
E	MACE finds the trivial group of size 1.	307
F	The user asks HR to construct a theory in 100 steps.	309
G	HR makes its first conjecture – that the multiplication tables for groups are non-empty.	102
H	HR makes a false conjecture which is disproved with a counterexample of size 2.	133

Event	Description	See pages
I	HR makes conjectures 5 and 7, which state that every group is a quasigroup (i.e. that every element appears in every row and column).	102
J	HR makes the conjecture that all groups are Abelian. This is disproved by a counterexample of size 6 – the smallest non-Abelian group.	102, 133
K	We have removed 47 steps from the output. In this period, 11 concepts and 36 conjectures were introduced.	
L	HR makes a conjecture which Otter cannot prove and MACE cannot disprove in the time available.	102, 133
M	The theory formation session ends and HR assesses that concept 28 is the most interesting using the weighted sum of measures.	159
N	After looking at the definition for concept 28, which are self-inversing elements, the user investigates why concept 28 is interesting by looking at the conjectures it is involved in. The third conjecture it is involved in (number 49) has a long proof length and is surprising.	175, 315, 316
O	The user looks at the data table for concept 28 and the multiplication table (concept 1) for group G04, confirming that the concept is self inversing elements.	310
P	The user prints the theorems in decreasing proof length order. We see that conjecture 35 has a proof of length 37.	310, 169
Q	The user looks at the open conjectures in the theory.	310
R	The user extends the time Otter is allowed to spend attempting a proof to 600 seconds, and tries to prove conjecture 46. Otter takes 62 seconds and produces a proof of length 63.	122, 305

B.3 Theory Formation Session in Semigroup Theory

In this session we wanted to highlight the theorem proving and disproving functionality of HR. In particular, we show how HR splits equivalence conjectures into implication conjectures and compiles a set of prime implicates which it uses to prove later theorems. Also, we show how a counterexample which has been introduced to disprove a conjecture also disproves a previously open conjecture. Such events are rare and often take a long time to happen. For this reason, this session is fairly long and we have had to remove much of the output for space considerations. The commands used were:

```
1. set::mode(algebra_default).
2. data(semigroup)::initialise(mace).
3. construct(14, entities).
```

B.3.1 Session Output

```
   SICStus 3  #5: Tue Aug 26 10:14:51 BST 1997
   HR1.11 is loaded. Please type help::me. for help

A |?- set::mode(algebra_default).

   sort_concepts=[yes]
   sort_conjectures=[yes]
   model_generator=[mace]
   prodrules=[[exists,forall,match,negate,conjunct]]
   complex_max=[6]
   concept_weight=[applicability,0.2]
   concept_weight=[novelty,0.3]
   concept_weight=[productivity,0.2]
   concept_weight=[theorem,0.3]
   theorem_weight=[proof_length,0.5]
   theorem_weight=[surprisingness,0.5]
   initial_concepts=[[element,pair,triple]]
   sort_marker=[step]

B |?- data(semigroup)::initialise(mace).

   Mace Generating First Model of Order: 1 S01
   all ax1 ax2 ax3 associativity: (ax1 * (ax2 * ax3) = (ax1 * ax2) * ax3).
C 1. *
   2. semigroup
   3. element
   4. pair
   5. triple

D * | 0 |
   --+---+
   0 | 0 |
   --+---+

   element: 0,

E |?- construct(14,entities).

   1. (exists a b c (a*b=c)). max_proofs semigroup 2
   2. all a ((exists b c (a*b=c))). max_proofs semigroup 2
   3. all a b ((exists c (a*b=c))). sos semigroup

F Mace generating counterexample of order: 1 2 3 4 5 6 7 8
   4. all a b ((exists c (a*c=b))). sos semigroup

   Mace generating counterexample of order: 1 2 S02
   Checking whether conjectures are disproved: 3:no,4:yes(New Concept = 6),
   (6) [S,a,b] : (exists c (a*c=b))

   * | 0 | 1 |
   --+---+---+
   0 | 0 | 0 |
   --+---+---+
   1 | 0 | 0 |
   --+---+---+
```

```
element: 0,1,

5. all a ((exists b c (b*a=c))). max_proofs semigroup 1
6. all a b (((exists c (a*c=b))) <-> ((exists d (d*a=b)))).
all a b ( (exists c (a*c=b)) -> (exists d (d*a=b)) ). sos
all a b ( (exists c (c*a=b)) -> (exists d (a*d=b)) ). sos

Mace generating counterexample of order: 1 2 S03
Checking whether conjectures are disproved: 3:no,6:yes(New Concept = 7),
(7) [S,a,b] : (exists c (c*a=b))

* | 0 | 1 |
--+---+---+
0 | 0 | 0 |
--+---+---+
1 | 1 | 1 |
--+---+---+

element: 0,1,

(8) [S,a] : (exists b c (b*c=a))
7. all a (((exists b c (b*c=a))) <-> (a*a=a)).
all a ( (exists b c (b*c=a)) -> a*a=a ). sos
all a ( a*a=a -> (exists b c (b*c=a)) ). max_proofs 0

Mace generating counterexample of order: 1 2 S04
Checking whether conjectures are disproved: 3:no,7:yes(New Concept = 9),
(9) [S,a] : a*a=a

* | 0 | 1 |
--+---+---+
0 | 0 | 1 |
--+---+---+
1 | 1 | 0 |
--+---+---+

element: 0,1,

(10) [S,a,b] : a*a=b
(11) [S,a,b] : a*b=a

Top 20 concepts: 9 10 3 7 8 1 6 11
Top 20 live concepts: 9(1) 10 3 7 8 1 6 11
Top 20 non theorems: 7 6 4
Top 20 open conjectures: 3
Top 20 theorems: 2 1 5
Sorted Production Rules: conjunct exists forall match negate

G 8. (exists a (a*a=a)). max_seconds semigroup

Mace generating counterexample of order: 1 2 3 4 5 6 7 8
(12) [S] : (all b (b*b=b))
(13) [S,a] : -(a*a=a)
(14) [S,a] : a*a=a & (all c (c*c=c))
(15) [S,a,b] : a*a=b & b*b=a
(16) [S,a,b] : a*a=b & (exists c d (c*d=a))
9. all a b ((a*a=b) <-> (a*a=b & (exists c d (c*d=b)))).
all a b ( a*a=b -> (exists c d (c*d=b)) ). max_proofs 0

H 10. all a b ((a*a=b & (exists c d (c*d=a))) <-> (a*a=b & a*b=a)).
all a b ( (exists c d (c*d=a)) & a*a=b -> a*b=a ). sos
all a b ( a*b=a -> (exists c d (c*d=a)) ). max_proofs 0
all a b ( a*a=b & a*b=a -> (exists c d (c*d=a)) ).
------------------------------------------------
all a b ( a*b=a -> (exists c d (c*d=a)) ).

Mace generating counterexample of order: 1 2 3 S05
Checking whether conjectures are disproved:
3:no,8:no,10:yes(New Concept = 17),
(17) [S,a,b] : a*a=b & a*b=a

* | 0 | 1 | 2 |
--+---+---+---+
0 | 0 | 0 | 0 |
--+---+---+---+
1 | 0 | 0 | 0 |
--+---+---+---+
2 | 0 | 0 | 1 |
--+---+---+---+

element: 0,1,2,

Top 20 concepts: 17 7 16 10 3 8 13 15 1 6 12 14 11 9
Top 20 live concepts: 17(1) 7 16 10 3 8 13 15 1 6 14 11
Top 20 non theorems: 10 7 6 4
Top 20 open conjectures: 8 3
Top 20 theorems: 2 1 5 9
```

```
Sorted Production Rules: conjunct exists forall match negate

11. all a b ((a*a=b & a*b=a) <-> (a*a=b & a*b=a & (exists c d (c*d=a)))).
12. all a b ((a*a=b & a*b=a) <-> (a*a=b & a*b=a & (exists c d (c*d=b)))).
all a b ( a*a=b & a*b=a -> (exists c d (c*d=b)) ).
------------------------------------------
all a b ( a*a=b -> (exists c d (c*d=b)) ).

13. all a b ((a*a=b & b*b=a) <-> (a*a=b & a*b=a & a*a=a)).
all a b ( a*a=a & a*a=b -> b*b=a ). max_proofs 4
all a b ( a*a=a & a*a=b & a*b=a -> b*b=a ).
------------------------------------------
all a b ( a*a=a & a*a=b -> b*b=a ).
all a b ( a*a=b & b*b=a -> a*a=a ). sos
all a b ( a*a=b & b*b=a -> a*b=a ). sos

Mace generating counterexample of order: 1 2 3 SO6
Checking whether conjectures are disproved:
3:no,8:no,13:yes(New Concept = 18),
(18) [S,a,b] : a*a=b & a*b=a & a*a=a

* | 0 | 1 | 2 |
--+---+---+---+
0 | 0 | 1 | 2 |
--+---+---+---+
1 | 1 | 2 | 0 |
--+---+---+---+
2 | 2 | 0 | 1 |
--+---+---+---+

element: 0,1,2,

14. all a b ((a*a=b & a*b=a) <-> (a*a=b & a*b=a & b*b=b)).
all a b ( a*a=b & a*b=a -> b*b=b ). max_proofs 3
15. (exists a b (a*a=b & a*b=a)). max_seconds semigroup

Mace generating counterexample of order: 1 2 3 4 5 6 7 8
(19) [S,a] : (exists b (a*a=b & a*b=a))
16. all a ((a*a=a) <-> ((exists b (b*b=a & b*a=b)))).
all a ( a*a=a -> (exists b (b*b=a & b*a=b)) ). max_proofs 0
all a ( (exists b (b*b=a & b*a=b)) -> a*a=a ). max_proofs 4

(20) [S] : (all a b (a*a=b & a*b=a))

Top 20 concepts: 19 9 7 15 16 18 17 10 20 13 3 12 14 6 8 11 1
Top 20 live concepts: 7(3) 15 16 17 10 13 3 14 6 8 11 1
Top 20 non theorems: 13 10 7 6 4
Top 20 open conjectures: 15 8 3
Top 20 theorems: 16 2 1 5 14 9 12 11
Sorted Production Rules: conjunct exists forall match negate

(21) [S,a,b] : (exists c (c*a=b)) & a*a=a
(22) [S,a,b] : (exists c (c*a=b)) & b*b=b
(23) [S,a,b] : (exists c (c*a=b)) & (exists d (d*b=a))
17. all a b ((a*a=b) <-> ((exists c (c*a=b)) & a*a=b)).
all a b ( a*a=b -> (exists c (c*a=b)) ). max_proofs 0

(24) [S,a,b] : (exists c (c*a=b)) & b*b=a
(25) [S,a,b] : (exists c (c*a=b)) & -(a*a=a)
(26) [S,a,b] : (exists c (c*a=b)) & -(b*b=b)
(27) [S,a,b] : (exists c (c*a=b)) & (all e (e*e=e))

Top 20 concepts: 19 22 23 9 25 26 15 16 18 21 24 7 17 10 13 20 6 12 14 27
Top 20 live concepts: 22(2) 23 25 26 15 16 21 24 7 17 10 13 6 14 3 8 11 1
Top 20 non theorems: 13 10 7 6 4
Top 20 open conjectures: 15 8 3
Top 20 theorems: 16 2 1 5 14 17 9 12 11
Sorted Production Rules: conjunct exists forall match negate

(28) [S,a,b] : (exists c (c*a=b)) & b*b=b & a*a=a
(29) [S,a,b] : (exists c (c*a=b)) & b*b=b & -(a*a=a)
(30) [S,a,b] : (exists c (c*a=b)) & b*b=b & (exists d e (d*e=a))
18. all a b (((exists c (c*a=b)) & b*b=b) <->
    ((exists d (d*a=b)) & b*b=b & (exists e f (e*f=b)))).
all a b ( a*a=a -> (exists c d (c*d=a)) ). max_proofs 0
all a b ( (exists c (c*a=b)) -> (exists d e (d*e=b)) ). max_proofs 0
all a b ( (exists c (c*a=b)) & b*b=b -> (exists d e (d*e=b)) ).
------------------------------------------
all a b ( b*b=b -> (exists c d (c*d=b)) ).

19. (exists a b ((exists c (c*a=b)) & b*b=b)). max_seconds semigroup

Mace generating counterexample of order: 1 2 3 4 5 6 7 8
20. all a ((exists b ((exists c (c*a=b)) & b*b=b))). max_seconds semigroup

Mace generating counterexample of order: 1 2 3 4 5 6 7 8
21. all a ((a*a=a) <-> ((exists b ((exists c (c*b=a)) & a*a=a)))).
all a ( a*a=a -> (exists b ((exists c (c*b=a)) & a*a=a)) ). max_proofs 2
```

```
all a ( (exists b ((exists c (c*b=a)) & a*a=a)) -> a*a=a ). max_proofs -1

22. ((all b (b*b=b))) <-> ((all c d ((exists e (e*c=d)) & d*d=d))).
(all b (b*b=b)) -> (all c d ((exists e (e*c=d)) & d*d=d)). max_seconds
(all a b ((exists c (c*a=b)) & b*b=b)) -> (all e (e*e=e)). max_proofs 0

Mace generating counterexample of order: 1 2 SO7
Checking whether conjectures are disproved:
3:no,8:no,15:no,19:no,20:no,22:yes(New Concept = 31),
(31) [S] : (all a b ((exists c (c*a=b)) & b*b=b))

* | 0 | 1 |
--+---+---+
0 | 0 | 0 |
--+---+---+
1 | 0 | 1 |
--+---+---+

element: 0,1,

Top 20 concepts: 15 16 19 21 23 24 28 30 7 22 9 25 31 10 17 12 14 18 26 27
Top 20 live concepts: 15(1) 16 21 23 24 28 7 22 25 10 17 14 26 8 3 13 6 11 1
Top 20 non theorems: 22 13 10 7 6 4
Top 20 open conjectures: 20 19 15 8 3
Top 20 theorems: 16 21 2 1 5 14 17 9 18 12 11
Sorted Production Rules: conjunct exists forall match negate

23. all a b ((a*a=b & a*b=a & a*a=a) <-> (a*a=b & b*b=a & a*a=a)).
all a b ( a*a=a & a*a=b -> a*b=a ). max_proofs 4
all a b ( a*a=a & a*a=b & b*b=a -> a*b=a ).
-------------------------------------------
all a b ( a*a=a & a*a=b -> a*b=a ).

I 24. all a b ((a*a=b & a*b=a & a*a=a) <-> (a*a=b & b*b=a & b*b=b)).
all a b ( a*a=a & a*a=b -> b*b=b ).
-------------------------------------------
all a b ( a*a=a & a*a=b -> a*b=a ).
all a b ( a*a=b & a*b=a -> b*b=b ).

all a b ( a*a=a & a*a=b & b*b=a -> b*b=b ).
-------------------------------------------
all a b ( a*a=a & a*a=b -> a*b=a ).
all a b ( a*a=b & a*b=a -> b*b=b ).

all a b ( a*a=a & a*a=b -> b*a=b ).
-------------------------------------------
all a b ( a*a=a & a*a=b -> b*b=a ).
all a b ( a*a=a & a*a=b -> a*b=a ).
all a b ( a*a=b & a*b=a -> b*b=b ).
all a b ( b*b=b & b*b=a -> b*a=b ).

all a b ( a*a=a & a*a=b & b*b=a -> b*a=b ).
-------------------------------------------
all a b ( a*a=a & a*a=b -> a*b=a ).
all a b ( a*a=b & a*b=a -> b*b=b ).
all a b ( b*b=b & b*b=a -> b*a=b ).

all a b ( a*a=a & a*a=b & a*b=a -> b*b=b ).
-------------------------------------------
all a b ( a*a=b & a*b=a -> b*b=b ).

25. all a b ((a*a=b & b*b=a) <-> (a*a=b & b*b=a & (exists c d (c*d=a)))).
all a b ( a*a=b & b*b=a -> (exists c d (c*d=a)) ).
-------------------------------------------
all a b ( b*b=a -> (exists c d (c*d=a)) ).

26. all a b ((a*a=b & b*b=a) <-> (a*a=b & b*b=a & (exists c d (c*d=b)))).
all a b ( a*a=b & b*b=a -> (exists c d (c*d=b)) ).
-------------------------------------------
all a b ( a*a=b -> (exists c d (c*d=b)) ).

(32) [S,a,b] : a*a=b & b*b=a & -(a*a=a)
27. all a b ((a*a=b & b*b=a & -(a*a=a)) <-> (a*a=b & b*b=a & -(b*b=b))).
all a b ( a*a=b & -(b*b=b) -> -(a*a=a) ). max_proofs 2
all a b ( a*a=b & b*b=a & -(b*b=b) -> -(a*a=a) ).
-------------------------------------------
all a b ( a*a=b & -(b*b=b) -> -(a*a=a) ).
all a b ( a*a=b & b*b=a & -(a*a=a) -> -(b*b=b) ).
-------------------------------------------
all a b ( b*b=a & -(a*a=a) -> -(b*b=b) ).

28. (exists a b (a*a=b & b*b=a)). max_seconds semigroup

Mace generating counterexample of order: 1 2 3 4 5 6 7 8
(33) [S,a] : (exists b (a*a=b & b*b=a))

Top 20 concepts: 18 22 16 19 21 23 24 28 30 32 33 7 17 15 10 25 31 9 12 14
Top 20 live concepts: 22(2) 16 21 23 24 28 33 7 17 15 10 25 9 14 26 13 8 3 6 11
```

```
Top 20 non theorems: 22 13 10 7 6 4
Top 20 open conjectures: 28 20 19 15 8 3
Top 20 theorems: 16 21 24 23 2 1 14 5 27 17 9 26 25 18 12 11
Sorted Production Rules: conjunct exists forall match negate

(34) [S,a] : (all c ((exists d (d*a=c)) & c*c=c))
(35) [S,a] : (all c ((exists d (d*c=a)) & a*a=a))
(36) [S,a,b] : -((exists c (c*a=b)) & b*b=b)
29. all a b ((a*a=b & a*b=a & a*a=a) <->
(a*a=b & (exists c d (c*d=a)) & a*a=a)).
all a b ( (exists c d (c*d=a)) & a*a=a & a*a=b -> a*b=a ).
------------------------------------------
all a b ( a*a=a & a*a=b -> a*b=a ).
all a b ( a*a=a & a*b=a -> (exists c d (c*d=a)) ).
------------------------------------------
all a b ( a*b=a -> (exists c d (c*d=a)) ).
all a b ( a*a=a & a*a=b -> (exists c d (c*d=a)) ).
------------------------------------------
all a b ( a*a=a -> (exists c d (c*d=a)) ).
all a b ( a*a=a & a*a=b & a*b=a -> (exists c d (c*d=a)) ).
------------------------------------------
all a b ( a*b=a -> (exists c d (c*d=a)) ).

(37) [S,a,b] : a*a=b & (exists c d (c*d=a)) & b*b=b
30. all a b ((a*a=b & (exists c d (c*d=a)))
<-> (a*a=b & (exists e f (e*f=a)) & (exists g h (g*h=b)))).
all a b ( (exists c d (c*d=a)) & a*a=b -> (exists e f (e*f=b)) ).
------------------------------------------
all a b ( a*a=b -> (exists c d (c*d=b)) ).

31. (exists a b (a*a=b & (exists c d (c*d=a)))). max_proofs semigroup 1

32. all a (((exists b c (b*c=a))) <->
((exists d (a*a=d & (exists e f (e*f=a)))))).
all a ( (exists b c (b*c=a)) ->
(exists d (a*a=d & (exists e f (e*f=a))))  ). max_proofs 5
all a ( (exists b (a*a=b & (exists c d (c*d=a)))) ->
(exists e f (e*f=a)) ). max_proofs 0

Top 20 concepts: 35 37 36 18 22 16 19 21 23 24 28 30 33 7 32 15 10 31 25 17
Top 20 live concepts: 36(3) 16 21 23 24 28 33 7 15 10 25 17 9 14 26 8 13 3 6 11
Top 20 non theorems: 22 13 10 7 6 4
Top 20 open conjectures: 28 20 19 15 8 3
Top 20 theorems: 32 16 31 21 29 24 23 2 1 14 5 27 30 17 9 26 25 18 12 11
Sorted Production Rules: conjunct exists forall match negate

(38) [S] : (exists a b (-((exists c (c*a=b)) & b*b=b)))
(39) [S,a] : (exists b (-((exists c (c*a=b)) & b*b=b)))
(40) [S,a] : (exists b (-((exists c (c*b=a)) & a*a=a)))
33. all a (((exists b (a*a=b & b*b=a))) <->
((exists c (c*c=a & (exists d e (d*e=c)))))).
all a ( (exists b (a*a=b & b*b=a)) ->
(exists c (c*c=a & (exists d e (d*e=c))))  ). max_proofs 0
all a ( (exists b (b*b=a & (exists c d (c*d=b)))) ->
(exists e (a*a=e & e*e=a)) ). max_seconds

J Mace generating counterexample of order: 1 2 3 4 S08
  Checking whether conjectures are disproved:
  3:no,8:no,15:no,19:no,20:no,28:no,33:yes(New Concept = 41),
  (41) [S,a] : (exists b (b*b=a & (exists c d (c*d=b))))

  * | 0 | 1 | 2 | 3 |
  --+---+---+---+---+
  0 | 0 | 0 | 0 | 0 |
  --+---+---+---+---+
  1 | 0 | 0 | 0 | 1 |
  --+---+---+---+---+
  2 | 0 | 0 | 1 | 2 |
  --+---+---+---+---+
  3 | 0 | 1 | 2 | 3 |
  --+---+---+---+---+

  element: 0,1,2,3,
K .
  .
  .

  (205) [S,a,b] : a*a=b & (all d ((exists e f (e*f=d))))
  (206) [S,a,b] : a*a=b & (exists c (a*c=a))
  (207) [S,a,b] : a*a=b & (exists c (b*c=b))
  (208) [S,a,b] : a*a=b & -(a*a=a)
  (209) [S,a,b] : a*a=b & -(b*b=b)
  (210) [S,a,b] : a*a=b & (exists c (c*a=a))
L 202. all a b ((a*a=b & (exists c (b*c=b))) <-> (a*a=b & (exists d (d*b=b)))).
  all a b ( a*a=b & (exists c (c*b=b)) -> (exists d (b*d=b)) ). sos
  all a b ( a*a=b & (exists c (b*c=b)) -> (exists d (d*b=b)) ). max_seconds

  Mace generating counterexample of order: 1 2 3 4 5 S14
```

```
Checking whether conjectures are disproved:
3:no,8:no,15:no,19:no,20:no,28:no,37:no,43:no,46:no,59:no,
70:no,77:no,79:no,83:no,88:no,89:no,90:no,91:no,93:no,95:no,
96:no,99:no,103:no,105:no,107:no,111:no,112:no,
M 121:yes(New Concept = 211),129:no,130:no,132:no,142:no,
143:no,145:no,158:no,167:no,173:no,184:no,195:no,201:no,
202:yes(New Concept = 212),
(211) [S,a,b] : (exists c (a*c=b)) & (exists d (d*a=b)) & (exists e (b*e=b))

(212) [S,a,b] : a*a=b & (exists c (c*b=b))

* | 0 | 1 | 2 | 3 | 4 |
--+---+---+---+---+---+
0 | 0 | 0 | 0 | 0 | 0 |
--+---+---+---+---+---+
1 | 0 | 0 | 0 | 0 | 0 |
--+---+---+---+---+---+
2 | 0 | 0 | 0 | 1 | 0 |
--+---+---+---+---+---+
3 | 0 | 0 | 0 | 1 | 0 |
--+---+---+---+---+---+
4 | 0 | 1 | 2 | 2 | 4 |
--+---+---+---+---+---+

element: 0,1,2,3,4,

yes
```

B.3.2 Commentary

Event	Description	See pages
A	HR loads and the user selects the default algebra settings.	307
B	The user chooses semigroup theory and asks HR to use MACE to construct a single semigroup.	66, 307
C	HR extracts 2 concepts (* and semigroup) from MACE's output and adds three others: elements, pairs of elements and triples of elements.	307
D	MACE finds the trivial semigroup of size 1.	307
E	The user asks HR to construct a theory until it has introduced 14 semigroups as counterexamples to false conjectures.	309
F	HR makes a false conjecture – that semigroups have the column-wise quasigroup property – which is disproved with a counterexample of size 2 where the body of the multiplication table contains only the element 0.	133

Event	Description	See pages
G	HR makes a conjecture which Otter cannot prove and MACE cannot disprove in the time available.	102, 133
H	HR makes a false conjecture which requires a semigroup of size 3 to disprove it.	133
I	HR uses its prime implicates to prove a theorem without using Otter. The conjecture is broken into 5 implication conjectures. These are stated above the lines and HR gives the proofs below the lines.	124, 127
J	HR makes a false conjecture which requires a semigroup of size 5 to disprove it.	133
K	We have removed 334 steps from the output. During this time, 165 concepts and 169 conjectures were introduced.	
L	HR makes conjecture number 202 which is disproved with a counterexample of size 5. This is the 14th semigroup in the theory.	133
M	The counterexample which disproved conjecture 202 also disproved conjecture 121. This leads to the introduction of concepts number 211 and 212.	137

B.4 Inventing and Investigating an Integer Sequence

In this session we recreate the session which led to the invention of the concept of integers for which the number of divisors is prime and the conjecture that, given an integer, if the sum of divisors is prime, then the number of divisors is prime. To do this, we asked HR to produce 50 concepts in number theory and then to identify those missing from the Encyclopedia. We then looked in detail at one of the sequences and asked HR to find subsequences of this in the Encyclopedia. The commands used for this session were:

```
1. set::mode(number_default).
2. data(integer)::from_file(smalldiv).
3. construct(50,concepts).
4. eis::load_sequences.
5. print::missing_number_types(100).
6. eis::assert_new_sequence(47,500).
7. eis::set(term_overlap_min,7), eis::set(term_overlap_max,10).
8. eis::subsequences_of(47).
9. eis::details('A023194').
10. concept(47)::learn_from_scratch(_).
11. eis::function('A023194',_,Num), \+ predicate(47,[Num]).
```

B.4.1 Session Output

```
  SICStus 3  #5: Tue Aug 26 10:14:51 BST 1997
  HR1.11 is loaded. Please type help::me. for help.

A | ?- set::mode(number_default).
  proofs=[no]
  counterexamples=[no]
  sort_conjectures=[no]
  prodrules=[[exists,match,forall,conjunct,size,split,negate]]
  complex_max=[8]
  concept_weight=[comprehensibility,0.2]
  concept_weight=[novelty,0.6]
  concept_weight=[productivity,0.2]
  sort_concepts=[yes]
  split_values=[[1,2]]

  yes
B | ?- data(integer)::from_file(smalldiv).

  1. integer
  2. divisor
  3. multiplication

  yes
C | ?- construct(50,concepts).
  (4) [I,N] : N = |{d1 : d1|I}|
  (5) [I] : 2|I
  (6) [I,d1] : d1|I & d1 = |{d2 : d2|I}|
  (7) [I,d1] : d1|I & I = |{d2 : d2|d1}|
  (8) [I,d1] : d1|I & 2|I
  (9) [I,d1] : d1|I & 2|d1
  (10) [I] : I=I*I

D Top 20 concepts: 4 5 10 6 7 2 3 8 9
  Top 20 live concepts: 4(1) 5 10 6 7 2 3 8 9
  Top 20 open conjectures: 1 2 3 4 5 6
  Sorted Production Rules: conjunct exists forall match negate size split

  (11) [I,N] : N = |{d1 : d1|I}| & 2|I
  (12) [I,N] : N = |{d1 : d1|I}| & 2|N
  (13) [I,N] : N = |{d1 : d1|I}| & I=I*I
  (14) [I] : I = |{d1 : d1|I}|
  (15) [I] : 2 = |{d1 : d1|I}|
  (16) [I,N] : N = |{d1 : d1|I}| & N = |{d2 : d2|N}|
  (17) [I,N] : N = |{d1 : d1|I}| & 2 = |{d2 : d2|I}|
  (18) [I,N] : N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}|
  (19) [I] : 2|I & I = |{d1 : d1|I}|
  (20) [I] : ~(2|I)

  Top 20 concepts: 6 7 4 11 12 16 18 14 19 5 8 9 20 2 17 15 3 13 10
  Top 20 live concepts: 6(1) 7 4 11 12 16 18 14 19 5 8 9 2 17 15 3 13 10
  Top 20 open conjectures: 1 2 3 4 5 6 7 8 9 10 11 12 13
  Sorted Production Rules: conjunct exists forall match negate size split

  (21) [I,d1] : d1|I & d1 = |{d2 : d2|I}| & 2|I
  (22) [I,d1] : d1|I & d1 = |{d2 : d2|I}| & ~(2|I)
  (23) [I,d1] : d1|I & d1 = |{d2 : d2|I}| & 2 = |{d3 : d3|I}|
  (24) [I,d1] : d1|I & d1 = |{d2 : d2|I}| & 2 = |{d3 : d3|d1}|
  (25) [I] : (exists d1 (d1|I & d1 = |{d2 : d2|I}|))
```

```
(26) [I,d1] : -(d1|I & d1 = |{d2 : d2|I}|)
(27) [I,N] : N = |{d1 : d1|I & d1 = |{d2 : d2|I}|}|
(28) [I,d1] : -(d1|I & I = |{d2 : d2|d1}|)
(29) [I,N] : N = |{d1 : d1|I & I = |{d2 : d2|d1}|}|
(30) [I,d1] : d1|I & d1 = |{d2 : d2|I}| & -(d1|I & I = |{d3 : d3|d1}|)

Top 20 concepts: 4 11 12 16 18 21 24 6 22 30 7 8 9 5 17 25 29 15 13 19
Top 20 live concepts: 4(1) 11 12 16 18 21 24 6 22 30 7 8 9 5 17 29 15 13 19 23
Top 20 open conjectures: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Sorted Production Rules: conjunct exists forall match negate size split

(31) [I,N] : N = |{d1 : d1|I}| & -(2|I)
(32) [I,N] : N = |{d1 : d1|I}| & -(2|N)
(33) [I,N] : N = |{d1 : d1|I}| & 2|I & 2|N
(34) [I,N] : N = |{d1 : d1|I}| & 2|I & 2 = |{d2 : d2|N}|
(35) [I,N] : N = |{d1 : d1|I}| & 2|I & -(2|N)
(36) [I,N] : N = |{d1 : d1|I}| & 2|I & -(N|I & I = |{d2 : d2|N}|)
(37) [I,N] : N = |{N : N = |{d1 : d1|I}| & 2|I}|
(38) [I,N] : N = |{d1 : d1|I}| & 2|N & -(2|I)
(39) [I] : (exists N (N = |{d1 : d1|I}| & 2|N))
(40) [I,N] : N = |{N : N = |{d1 : d1|I}| & 2|N}|

Top 20 concepts: 4 16 18 21 24 31 32 33 6 22 30 34 35 38 36 11 7 12 8 9
Top 20 live concepts: 4(1) 16 18 21 24 31 32 33 6 22 30 34 35 38 36 11 7 12 8 9
Top 20 open conjectures: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Sorted Production Rules: conjunct exists forall match negate size split

(41) [I,N] : N = |{d1 : d1|I}| & N = |{d2 : d2|N}| & -(2|I)
(42) [I] : (exists N (N = |{d1 : d1|I}| & N = |{d2 : d2|N}|))
(43) [I,N] : N = |{N : N = |{d1 : d1|I}| & N = |{d2 : d2|N}|}|
(44) [I,N] : N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}| & -(2|I)
(45) [I,N] : N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}| & -(2|N)
(46) [I,N] : N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}| & -(N|I & I = |{d3 : d3|N}|)
E (47) [I] : (exists N (N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}|))
(48) [I,N] : N = |{N : N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}|}|
(49) [I] : (exists d1 (d1|I & d1 = |{d2 : d2|I}| & 2|I))
(50) [I,d1] : -(d1|I & d1 = |{d2 : d2|I}| & 2|I)

Top 20 concepts: 4 6 24 31 32 33 22 30 34 35 38 49 50 11 36 41 44 45 46 7
Top 20 live concepts: 4(1) 6 24 31 32 33 22 30 34 35 38 50 11 36 41 44 45 46 7 12
Top 20 open conjectures: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Sorted Production Rules: conjunct exists forall match negate size split

yes
F | ?- eis::load_sequences.
This may take some time.

yes
G | ?- print::missing_number_types(100).
This will take a few minutes
H 5. [I] : 2|I
  2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42
  44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82
  84 86 88 90 92 94 96 98 100

I 47. [I] : (exists N (N = |{d1 : d1|I}| & 2 = |{d2 : d2|N}|))
  2 3 4 5 7 9 11 13 16 17 19 23 25 29 31 37 41 43 47 49 53 59 61
  64 67 71 73 79 81 83 89 97

J 49. [I] : (exists d1 (d1|I & d1 = |{d2 : d2|I}| & 2|I))
  2 8 12 18 24 36 40 56 60 72 80 84 88 96

yes
K | ?- eis::assert_new_sequence(47,500).
yes
L | ?- eis::set(term_overlap_min,7), eis::set(term_overlap_max,10).
yes
M | ?- eis::subsequences_of(47).

Looking for subsequences of:
47 [abraham,integer,5]

Subject to:
term_overlap_min=7
term_overlap_max=10

N A023194 Sum of divisors of n is prime.
  A020591 Smallest nonempty set S containing prime divisors of 4k+6
          for each k in S.
  A020613 Smallest nonempty set S containing prime divisors of 7k+9
          for each k in S.
  A007505 Primes of form 3.2^n -1.
  A007700 n, 2n+1, 4n+3 all prime.
  A031439 a(n) is the greatest prime factor of a(n-1)^2+1.
  .
  .
  A045703 Primes that can be written as sum of two squares of Fibonacci numbers.
  A037028 a(n) = prime closest to e^n.
```

```
A002230 Primes with record values of the least positive primitive root.
A037231 Length of Pratt certificate for prime increases.
A004062 (6^n - 1)/5 is prime.
A034900 a(n) is square mod a(i), i &lt; n; a(n) prime.
A030087 Primes such that digits of p do not appear in p^3.
A020599 Smallest nonempty set S containing prime divisors of 5k+7
        for each k in S.
A029973 Palindromic primes in base 5.
------

0 166 matches found in 800.86 seconds.
  yes
P | ?- eis::details('A023194').
  A023194 Sum of divisors of n is prime.
  ---
  2 4 9 16 25 64 289 729 1681 2401 3481 4096 5041 7921 10201 15625 17161
  27889 28561 29929 65536 83521 85849 146689 262144 279841 458329 491401
  531441 552049 579121 597529 683929 703921 707281 734449 829921 1190281
  ---
  nonn,easy,nice
  ---
  yes
  | ?- concept(47)::learn_from_scratch(_).
  yes
Q | ?- eis::function('A023194',_,Num), \+ predicate(47,[Num]).
  no
```

B.4.2 Commentary

Event	Description	See pages
A	HR loads and the user selects the default settings for number theory.	307
B	The user chooses the number theory data from the `smalldiv` file. This contains the numbers 1-10 and concepts: integers, divisor and multiplication.	63, 307
C	The user asks HR to construct 50 concepts.	309
D	Concept 4 is assessed as the most interesting. This concept is the well known τ function in number theory (number of divisors of an integer). This concept is always rated in the top 3 most interesting during this session.	159
E	HR invents the concept of integers with a prime number of divisors.	352
F	The user loads the Encyclopedia of Integer Sequences. This is not loaded by default as it is a very large file.	110, 317
G	The user asks HR to identify those sequences in the theory which are missing from the Encyclopedia. These are output in order of decreasing comprehensibility.	110

Event	Description	See pages
H	This is an anomaly – the sequence of even numbers is present in the Encyclopedia, but starts with a zero.	
I	This is the concept of integers with a prime number of divisors, one of HR's sequences which has since been added to the Encyclopedia of Integer Sequences (number A009087).	116, 352
J	This is the concept of even refactorable numbers, which has since been added to the Encyclopedia (sequence number 57265).	
K	The sequence from concept 47 is extended up to 500 and added to HR's internal copy of the Encyclopedia.	110
L	The user stipulates that subsequences must have at least 7 overlapping terms with concept 47, but at most 10. The latter restriction is to cut down on the number of sequences output which are specialisations of primes (which will be trivial subsequences of concept 47).	114, 110
M	The user asks for the subsequences of concept 47.	112, 110
N	The first answer produced is A023194: those integers, n, where the sum of divisors of n is prime. It is not obvious why this should be a subsequence.	
O	166 matches are produced (but we have omitted some of them, due to space considerations). It took around 14 minutes to run through the Encyclopedia completely.	
P	The user asks for more details about sequence A023194.	110
Q	The user checks that all the entries from sequence A023194 satisfy the conditions of concept 47. This adds greater evidence to the conjecture.	110

Appendix C. Number Theory Results

2, 3, 4, 5, 7, 9, 11, 13, 16, 17, 19, 23, 25, 29, 31, 37, 41, 43, ...
A009087. Integers with a prime number of divisors.

We present here proofs of some of the conjectures HR made in number theory, and the other results which arose from investigations of the concepts and conjectures HR made. We show that the refactorable numbers defined by HR have many interesting properties, some of which were noticed by HR. We also prove the conjecture made by HR that if the sum of divisors of an integer is prime, then the number of divisors will be prime. Finally we prove some ad hoc results found by HR. Throughout, where applicable, we briefly mention how the conjecture was made by HR.

C.1 Refactorable Numbers

As discussed in §12.3.2, HR produced a type of integer which we called refactorable numbers.[1]

Definition 1 (Refactorable Numbers)
An integer, n, is refactorable if the number of divisors of n is itself a divisor of n. i.e. $\tau(n)|n$.

The first members of this sequence are:

$$1, 2, 8, 9, 12, 18, 24, 36, 40, 56, 60, 72, 80, 84, 88, 96, \ldots$$

For example, 9 is in the sequence, as it has 3 divisors and 3 itself divides 9.

Refactorable numbers have many interesting properties. After giving some initial results, we describe the relationship between refactorables and some well known number types. We then look at pairs and triples of refactorables, and end by looking at the distribution of refactorables.

[1] Named by Toby Walsh and appearing as sequence A033950 in the Encyclopedia of Integer Sequences. A rival name proposed in [Kennedy & Cooper 90] is 'tau numbers', but the Encyclopedia gives preference to refactorable numbers, and we follow their example.

C.1.1 Initial Results

Lemma 1 (Theorem 273 from [Hardy & Wright 38])
If the prime factorisation of integer n is:

$$n = \prod_{i=1}^{l} p_i^{k_i},$$

then

$$\tau(n) = \prod_{i=1}^{l} (k_i + 1)$$

Lemma 2
For all odd integers a:

a is refactorable if and only if $2a$ is refactorable.

Proof of Lemma 2
Suppose a is refactorable and has prime factorisation: $p_1^{k_1} \dots p_l^{k_l}$. Then by Lemma 1, $(k_1 + 1) \dots (k_l + 1)$ divides a. Because a is odd, the prime factorisation of $2a$ will be $2p_1^{k_1} \dots p_l^{k_l}$, where $\forall i, p_i \neq 2$. Therefore $\tau(2a) = 2(k_1 + 1) \dots (k_l + 1)$ which divides $2a$, because $(k_1 + 1) \dots (k_l + 1)$ divides a. Hence $2a$ is also refactorable. Conversely, suppose that $2a$ is refactorable. Again, as a is odd, $\tau(2a) = 2(k_1 + 1) \dots (k_l + 1)$, which divides $2a$, meaning that $(k_1+1) \dots (k_l+1)$ divides a, and so $\tau(a)$ divides a, hence a is refactorable also. □

Theorem 1
There are an infinite number of odd refactorables and an infinite number of even refactorables.

Proof of Theorem 1
From Lemma 1, for any odd prime, p, integers of the form p^{p-1} will be odd and have p divisors, and so are refactorable. By Lemma 2, integers of the form $2p^{p-1}$ will be even and refactorable. As there are infinitely many primes, it follows that there are infinitely many odd and even refactorable numbers. □

A more interesting way to prove Theorem 1 is to map each integer onto a distinct refactorable number. This is achieving using the following map on the prime factorisation of the integer:

$$n = p_1^{k_1} \dots p_l^{k_l} \mapsto p_1^{p_1^{k_1}-1} \dots p_l^{p_l^{k_l}-1}$$

For example:

$$3 = 3^1 \mapsto 3^{3^1-1} = 3^2 = 9,$$

$$6 = 2^1 3^1 \mapsto 2^{2^1-1} 3^{3^1-1} = 2^1 3^2 = 18.$$

Integers produced by this map will have $p_1^{k_1} \ldots p_l^{k_l}$ $(= n)$ divisors, and hence will be refactorable. Furthermore, as the prime factorisation of n is unique, the refactorable number output from this map will also be unique, and hence there are an infinite number of refactorables. Note that this map is not 1:1, as it is easy to show that, for example, the number 12 is refactorable, but cannot be written in the form $p_1^{p_1^{k_1}-1} \ldots p_l^{p_l^{k_l}-1}$.

C.1.2 Relation to Other Number Types

Since primes have two divisors, 2 is the only prime refactorable number. It's also very easy to show that 2 is the only square-free refactorable number. We discuss here some relationships between refactorables and some other well known types of number.

Theorem 2
All odd refactorable numbers are squares.

Proof of Theorem 2
Suppose the prime factorisation of a is $p_1^{k_1} \ldots p_l^{k_l}$ and a is odd and refactorable. Therefore we know that $(k_1 + 1) \ldots (k_l + 1)$ divides a. Therefore, as a has no even divisors, each k_i must be even. Writing each k_i as $2j_i$ we see that:

$$a = p_1^{2j_1} \ldots p_l^{2j_l} = \left(p_1^{j_1} \ldots p_l^{j_l}\right)^2$$

and hence a is a square number. □

While the above theorem was conjectured by HR, we initially overlooked it, and discovered it independently. In fact, we were led to the discovery of the result when trying to prove a different conjecture made by HR:

Conjecture 1
Given a refactorable number, n, then define the following function:

$$f(n) = |\{(a, b) \in \mathbf{N} \times \mathbf{N} : ab = n \text{ and } a \neq b\}|$$

Then $f(n)$ divides n if and only if n is a non-square.

It turned out that this conjecture was false, but the smallest counterexamples to it are 36360900, 79388100 and 155600676, which are the first three square refactorable numbers that are divisible by $f(n)$.

We used HR to find sequences described with the "nice" keyword in the Encyclopedia of Integer Sequences which were disjoint from refactorables. There were a hundred answers, many of which were specialisations of prime numbers. However, perfect numbers were also output. Perfect numbers are those positive integers for which the sum of the divisors equals twice the number itself. We use the notation $\sigma(n)$ for the sum of the divisors of n. For example, 28 is a perfect number because the divisors of 28 are $1, 2, 4, 7, 14$ and 28, so:

$$\sigma(28) = 1 + 2 + 4 + 7 + 14 + 28 = 56 = 2 \times 28$$

Perfect numbers have been studied since antiquity and are a very important concept in number theory. For more information on perfect numbers, see [Beiler 96]. HR had therefore noticed a relation between refactorables and perfect numbers, which we formulated as the following theorem:

Theorem 3
Perfect numbers are not refactorable.

Proof of Theorem 3
We need to refer to Theorems 18 and 277 from [Hardy & Wright 38]:

- Theorem 18: if $n > 1$ and $a^n - 1$ is prime then $a = 2$ and n is prime.

- Theorem 277: (paraphrased): Any even perfect number is of the form $2^{n-1}(2^n - 1)$ where $2^n - 1$ is prime.

(a) Even perfect numbers. Using Theorem 277 above we know that if a is an even perfect number, it has the form $2^{n-1}(2^n - 1)$ where $2^n - 1$ is an odd prime, say p. Using Theorem 18 above, we know that n must be prime also. Using Lemma 1, we see that:

$$\tau(a) = \tau(2^{n-1}(2^n - 1)) = \tau(2^{n-1}p) = 2n,$$

so a has $2n$ divisors. If a is refactorable then $2n$ divides a, which means that either $n = 2$ or $n = p$ (as n is a prime and $a = 2^{n-1}p$). If $n = 2$ then $a = 2^{2-1}(2^2 - 1) = 6$, which is not refactorable. If $n = p$ then $n = 2^n - 1$, which is impossible for a prime n, because for any $n > 1$, $2^n - 1 > n$. Hence a cannot be refactorable.

(b) Odd perfect numbers. No odd perfect numbers are known. If one were to exist, say b with divisors $d_1 < \ldots < d_k = b$, then each d_i must be odd, and by definition, $d_1 + \ldots + d_{k-1} = b$. The sum of an even number of odd integers is even, so, as b is odd, we know that $k - 1$ must be odd, so b has an even number of divisors. Therefore b cannot be refactorable as it is odd and cannot be divisible by an even number. □

Note that **multiply perfect numbers**, defined to be integers where the sum of divisors is a multiple of the number, *can* be refactorable. For example, the number 672 is such that $\sigma(672) = 3 \times 672$, and this is refactorable (it is actually the smallest refactorable multiply perfect number).

Furthermore, there is an appealing similarity between perfect numbers and refactorables. Using the methods discussed in §7.5, HR discovered the following theorem:

Theorem 4
For any even perfect number x, there is an integer, a, such that $lcm(a, \sigma(a)) = x$. (where $lcm(u, v)$ is the lowest common multiple of u and v).

Proof of Theorem 4
From Theorem 277 of [Hardy & Wright 38], we note that $x = 2^{n-1}(2^n - 1)$ for some n, where $2^n - 1$ is a prime. If we take $a = 2^{n-1}$ then $\sigma(a) = 1 + 2 + \ldots + 2^{n-1} = 2^n - 1$, and so $lcm(a, \sigma(a)) = 2^{n-1}(2^n - 1) = x$ as required. □

If we note that for all refactorables, $lcm(n, \tau(n)) = n$, (which HR also conjectured), we see the following similarity between perfect numbers and refactorables:

- Refactorable numbers are of the form $lcm(a, \tau(a))$ for some a.
- Perfect numbers are of the form $lcm(a, \sigma(a))$ for some a.

In fact:

- Refactorable numbers are those integers, n, for which $lcm(n, \tau(n)) = n$.
- Odd prime numbers are those integers, n, for which $lcm(n, \tau(n)) = 2n$.
- Perfect numbers are those integers, n, for which $lcm(n, \sigma(n)) = 2n$.

By asking HR to find subsequences and supersequences of refactorables from the Encyclopedia, it found the following three conjectures. We have proved the first (Theorem 5), but the final two remain open (Conjectures 2 and 3).

Theorem 5
Refactorable numbers are only congruent to 0, 1, 2 or 4 mod 8.

Proof of Theorem 5
By Theorem 2, odd refactorables are squares. If an odd number can be written as $8n + 1$ then $(8n + 1)^2 = 64n^2 + 16n + 1 \equiv 1 (\text{mod } 8)$. A similar analysis for odd numbers written as 8n+3, 8n+5 and 8n+7 shows that the square of an odd number is always congruent to 1 mod 8. Hence odd refactorables are congruent to 1 mod 8. Even numbers must be congruent to 0, 2, 4 or 6 mod 8. However, if an even refactorable was congruent to 6 mod 8, then it would be of the form $8n + 6 = 2(4n + 3)$ for some n. By Lemma 2 this means that $4n + 3$ is refactorable. But $4n + 3$ is congruent to 3 or 7 mod 8, which is a contradiction to our above result that odd refactorables are congruent to 1 mod 8. Hence even refactorables are congruent to $0, 2$ or 4 mod 8. □

Conjecture 2
Integers, n, for which $\phi(\sigma(n)) = n$ are refactorable.

We use the notation $\phi(n)$ to be the number of integers less than and relatively prime to n. (Where two numbers are relatively prime if the only divisor they share is 1).

When investigating this conjecture, we noticed the following pattern:

$$\phi(\sigma(2^{2^n-1})) = 2^{2^n-1},$$

which was true for $n = 1, 2, 3, 4$ and 5.

Using Theorem 275 from [Hardy & Wright 38], we see that:

$$\begin{aligned}\phi(\sigma(2^{2^n-1})) &= \phi(2^{2^n} - 1) \\ &= \phi(1 + 2 + \ldots + 2^{2^n-1}) \\ &= \phi((1+2)(1+4)\ldots(1+2^{2^n-2})(1+2^{2^n-1}))\end{aligned}$$

Now if each $(1 + 2^{2^i})$ is prime, by Theorem 62 of [Hardy & Wright 38]:

$$\begin{aligned}\phi(\sigma(2^{2^n-1})) &= \phi(1+2)\phi(1+4)\ldots\phi(1+2^{2^n-1}) \\ &= (2)(4)\ldots(2^{2^n-1}) \\ &= 2^{1+2+4+\ldots+2^{n-1}} \\ &= 2^{2^n-1}\end{aligned}$$

Unfortunately $(1 + 2^{2^6})$ is composite, so the pattern stops at $n = 6$.

Conjecture 3
For all integers $k \geq 3$, numbers of the form $k!/3$ are refactorable.

C.1.3 Pairs and Triples of Refactorables

As odd refactorables are square numbers, we cannot have four or more consecutive refactorables since positive squares always differ by more than 2. However, there are 13 adjacent pairs of refactorable numbers between 1 and 1,000,000. For example, the first four pairs of refactorable numbers are $(1, 2), (8, 9), (1520, 1521)$ and $(50624, 50625)$. We have not yet found any adjacent triples of refactorables.

Theorem 6
If refactorable numbers x and y are relatively prime, then xy will also be refactorable. In particular if a and $a + 1$ are refactorable then $a(a + 1)$ will be refactorable.

Proof of Theorem 6
If x and y are relatively prime, then $\tau(xy) = \tau(x)\tau(y)$, and if they are both refactorable, then $\tau(x)|x$ and $\tau(y)|y$, so $\tau(xy)|xy$ and we see that xy is also refactorable. Two consecutive integers are relatively prime, so the product of two consecutive refactorables will also be refactorable. □

Hence, if we multiply any adjacent pair of refactorables, we get a third. For example, $8 \times 9 = 72$ is refactorable, and so is $1520 \times 1521 = 2311920$.

Conjecture 4
There are infinitely many pairs of refactorable numbers.

The above conjecture is based purely on our intuition of refactorable numbers and was not made by HR. As yet, we have no insight about the truth of this statement.

We cannot yet rule out triples of refactorable numbers, but the following theorem imposes a very restrictive constraint on their value.

Theorem 7
If $(a-1, a, a+1)$ is a triple of refactorable numbers, then a must be of the form:

$$\left(\sum_{i=0}^{n} 2^i \left({}^{2n+1}C_{2i}\right)\right)^2$$

for some integer n. Note that ${}^xC_y = \frac{x!}{(x-y)!y!}$.

Proof of Theorem 7
By Theorem 5, three consecutive refactorables must be of the form $(8m, 8m+1, 8m+2)$ for some m. Hence a must be odd, and so by Theorem 2, a must be a square number, say b^2. Furthermore, $a+1$ is not divisible by 4, so must have prime factorisation $a+1 = 2p_1^{k_1} \ldots p_l^{k_l}$, where the p_is are distinct odd primes. Therefore $\tau(a+1) = 2(k_1+1)\ldots(k_l+1)$ and each k_i+1 must be odd as $a+1$ is refactorable. So each k_i must be even and hence $a+1$ is twice an odd square number, so we can write $a+1 = 2c^2$, in particular, $b^2+1 = 2c^2$. This means that (b, c) must be a solution of the Diophantine equation $x^2 - 2y^2 = -1$. Theorem 244 of [Hardy & Wright 38] states that positive integer solutions to this equation are given by:

$$x + y\sqrt{2} = (1+\sqrt{2})^{2n+1}$$

for integers n. Expanding the coefficient of x on the right hand side, we get:

$$x = \sum_{i=0}^{n} 2^i \left({}^{2n+1}C_{2i}\right) \tag{C.1}$$

and so a is the square of this, as required. □

Numbers of the form in equation C.1 become very large as n increases. For example, if we take $n = 10$, then $a = 2982076586042449$. By considering $n \leq 35$ we have calculated that there are no triples of refactorables between 1 and 10^{53}.

Conjecture 5
There are no triples of refactorable numbers.

Again, this conjecture was not made by HR and is based on the fact above that there are no triples of refactorables less than 10^{53}.

C.1.4 Distribution

We cannot yet give an accurate measure for the number of refactorables less than a given n, but we can say how many there are with a given number of divisors.

Theorem 8
The number of refactorables with n divisors is:

- 1 if $n = 1$ or $n = 4$.
- $k!$ if n is the product of k distinct primes (i.e. it is square free).
- Infinite otherwise.

Proof of Theorem 8
(i) Clearly 1 is the only refactorable number with one divisor. If an integer, s, has four divisors, then it must be of the form p^3 or pq for distinct primes p and q. Taking the first case, if s is refactorable, then p must be 2 and the refactorable number is 8. In the second case, there are no refactorables of the form pq because 4 cannot divide the product of two distinct primes. Hence there is a single refactorable number with 4 divisors.

(ii) If n is the product of k distinct primes, then $n = p_1 \dots p_k$ and any integer, b, with n divisors must be of the form:

$$b = a_1^{p_1-1} \dots a_k^{p_k-1}$$

for distinct primes $a_1, \dots, a_k$. If b is refactorable with n divisors, then n must divide b, so $\{a_1, \dots, a_k\} = \{p_1, \dots, p_k\}$ and there are $k!$ ways to choose the a_i's from the p_i's. Each choice will produce a different prime factorisation for b, which, because prime factorisations are unique, will produce a different value for b. Hence there are $k!$ possibilities for b.

(iii) Suppose that n is not square free and has prime factorisation $p_1^{m_1} \dots p_k^{m_k}$. Hence, $m_i > 1$ for some m_i. Then, for any prime, q, such that $q \notin \{p_1, \dots, p_k\}$, the integer:

$$s = q^{p_i-1} p_1^{m_1-1} \dots p_i^{m_i-1-1} \dots p_k^{m_k-1} \tag{C.2}$$

has n divisors. This is because, by the application of Lemma 1 above:

$$\tau(s) = p_i(p_1^{m_1} \dots p_{i-1}^{m_{i-1}} p_i^{m_i-1} p_{i+1}^{m_{i+1}} \dots p_k^{m_k}) = p_1^{m_1} \dots p_k^{m_k} = n$$

Comparing the prime factorisations of s and $\tau(s)$, we see that s is refactorable unless $m_i > p_i^{m_i-1} - 1$, which only occurs if $p_i = m_i = 2$. Hence, because there are infinitely many primes, for any square free integer n, there are infinitely many numbers of the form (C.2) above which have n divisors and

are refactorable, with two exceptions. Firstly, if $n = 2^2$, then $n = 4$, which has been dealt with above. Secondly, if $n = 2^2 p_2 \dots p_k$, then for any prime, q, such that $q \notin \{p_2, \dots, p_k\}$, the integer:

$$t = q 2^{p_2 - 1} p_2 p_3^{p_3 - 1} \dots p_k^{p_k - 1} \tag{C.3}$$

has $4 p_2 p_3 \dots p_k = n$ divisors, and is refactorable because $p_2 > 2$ implies $p_2 - 1 \geq 2$, so n divides t. Again, because there are infinitely many prime numbers, there are infinitely many numbers of the form (C.3). Therefore, given an integer, n, of the form $n = 2^2 p_2 \dots p_k$, there are infinitely many refactorable numbers with n divisors. □

This theorem shows that, for instance, there are precisely 2 refactorable numbers with six divisors, namely 12 and 18, and precisely 6 refactorables with 30 divisors, namely:

$$\begin{array}{llllll} 2^4 3^2 5^1 & = & 720 & 2^4 3^1 5^2 & = & 1200 \\ 2^2 3^4 5^1 & = & 1620 & 2^1 3^4 5^2 & = & 4050 \\ 2^2 3^1 5^4 & = & 7500 & 2^1 3^2 5^4 & = & 11250 \end{array}$$

Using the GAP program, [Gap 00], we have calculated the distribution of the refactorables, and present the results compared with the distribution of the prime numbers in Table C.1.

$n <$	primes	refacs.	odd refacs.	even refacs.	prime pairs	refacs. pairs
10	4	4	2	2	2	2
10^2	25	16	2	14	8	2
10^3	168	92	5	87	35	2
10^4	1229	665	15	650	205	3
10^5	9592	5257	34	5223	1224	5
10^6	78498	44705	87	44618	8169	13
10^7	664579	394240	237	394003	58980	27
10^8	5761455	**3558181**	650	**3557531**	440312	75
10^9	50847534	**32608999**	1813	**32607186**	3424506	187
10^{10}	455052511	**302172507**	5152	**302167355**	27412679	468
10^{11}	4118054813	?	14889	?	224376048	1219

Table C.1 The distribution of refactorables compared with primes

The values in bold face have been supplied by David Wilson, and we are very grateful for his contribution [Wilson & Sloane 99]. David has also pointed out that if $(a - 1, a, a + 1)$ is a triple of refactorables, then each prime factor of a must occur to at least the 6th power [Wilson 00]. Note that Theorem 2 has helped us to calculate the distribution of odd refactorables further than even refactorables. From this empirical evidence, we can make a prediction about the distribution of the refactorables:

Conjecture 6
The number of refactorables less than x is at least $\frac{x}{2log(x)}$.

We made this conjecture because the prime number theorem (see Theorem 6 from [Hardy & Wright 38]) states that the number of primes less than x tends to $\frac{x}{log(x)}$, and the number of refactorables in Table C.1 is always more than half the number of primes.

C.2 Integers with a Prime Number of Divisors

We are interested here in integers where the number of divisors is a prime number, i.e. those n for which $\tau(\tau(n)) = 2$. These are a type of integer invented by HR (sequence number A009087) with first terms as follows:

$$2, 3, 4, 5, 7, 9, 11, 13, 16, 17, 19, 23, 25, 29, 31, 37, 41, 43, 47, 49, 53, 59, 61, \ldots$$

As discussed in §7.5 we are interested in proving a pleasing result HR conjectured: if the sum of the divisors of n is prime, then the number of divisors of n will be prime. We prove a more general result (Theorem 9 below) from which this theorem follows as a corollary. We first require a lemma about the nature of integers with a prime number of divisors and some results from [Hardy & Wright 38].

Lemma 3
For all integers, n:

$$\tau(n) \text{ is prime} \iff n = p^{q-1} \text{ for primes } p \text{ and } q.$$

Proof of Lemma 3
If $n = p^{q-1}$ then $\tau(n) = q$, hence $\tau(n)$ is prime. Conversely, suppose that the prime factorisation of n is $p_1^{k_1} \ldots p_l^{k_l}$, and that $\tau(n)$ is prime. Now $\tau(n) = (k_1 + 1) \ldots (k_l + 1)$, hence $l = 1$, and n must be of the form p^a for some a. So, $\tau(p^a) = a + 1$, and a must be one less than a prime, q. □

Lemma 4 (Theorem 274 of [Hardy & Wright 38])
If the prime factorisation of integer n is:

$$n = \prod_{i=1}^{l} p_i^{k_i},$$

then:

$$\sigma_m(n) = \prod_{i=1}^{l} \left(\frac{p_i^{m(k_i+1)} - 1}{p_i - 1} \right)$$

(where $\sigma_m(n)$ is the sum of the mth powers of the divisors of n).

We also need to remind ourselves of the following well known identity:

$$\frac{a^b - 1}{a - 1} = 1 + a^2 + \ldots + a^{b-1} = \sum_{i=0}^{b-1} a^i.$$

Theorem 9
$\forall\, m, n \in \mathbf{N}, \quad \tau(\sigma_m(n)) = 2 \Rightarrow \tau(\tau(n)) = 2.$

Proof of Theorem 9
Let the prime factorisation of n be $p_1^{k_1} \ldots p_l^{k_l}$, and let m be an integer. Suppose also that $\tau(\sigma_m(n)) = 2$, i.e. that $\sigma_m(n)$ is prime. We see from Lemma 4 that $\sigma_m(n)$ has at least $l + 1$ factors (counting 1 as well). Therefore, as $\sigma_m(n)$ is prime, $l = 1$. Hence we can write $n = p^a$ for some prime p and some $a \in \mathbf{N}$. Assume that $\tau(n)$ is composite, then $\tau(n) = a + 1 = xy$ for some $x, y \in \mathbf{N}, x > 1, y > 1$. Hence $a = xy - 1$. So, using Lemma 4 again:

$$\begin{aligned}
\sigma_m(n) &= \frac{p^{m(a+1)} - 1}{p - 1} = \frac{p^{m(xy-1+1)} - 1}{p - 1} = \frac{p^{mxy} - 1}{p - 1} \\
&= \frac{(p^{mx} - 1)(p^{(y-1)mx} + p^{(y-2)mx} + \ldots + p^{mx} + 1)}{p - 1} \\
&= \frac{p^{mx} - 1}{p - 1} \sum_{i=1}^{y} p^{(y-i)mx} \\
&= \sum_{i=0}^{mx-1} p^i \sum_{j=1}^{y} p^{(y-j)mx}
\end{aligned}$$

As neither of the summations in this final product equal 1, this provides a contradiction, because $\sigma_m(n)$ is prime. Hence our assumption that $\tau(n)$ is composite must be false, and we see that $\tau(n)$ is a prime. □

Corollary 1
Taking $m = 1$ in Theorem 9, we see that:

$$\forall\, n \in \mathbf{N}, \quad \tau(\sigma(n)) = 2 \Rightarrow \tau(\tau(n)) = 2$$

That is, if the sum of divisors of n is prime, then the number of divisors of n will be prime.

Corollary 2
If the sum of divisors of integer n is prime then n will be of the form p^{q-1} for primes p and q.

This final corollary enables a quick calculation of the terms of the sequence where $\sigma(n)$ is prime (sequence number A023914 in the Encyclopedia).

C.3 Other Results

Using the Encyclopedia of Integer Sequences, HR noticed that perfect numbers form a subsequence of sequence A007517, the sequence where the *nth* term is $\phi(n)(\sigma(n) - n)$. We interpreted this result as the following theorem.

Theorem 10
For any perfect number x, there is an integer, a, such that $\phi(a)(\sigma(a)-a) = x$.

Proof of Theorem 10
Using Theorem 277 from [Hardy & Wright 38] again, we know that if x is an even perfect number, it has the form $2^{n-1}(2^n - 1)$. If we take $a = 2^n$, then $\sigma(a) = 2^{n+1} - 1$ and the odd integers less than a will be relatively prime to it. So $\phi(a) = a/2 = 2^{n-1}$. Therefore:

$$\begin{aligned}\phi(a)(\sigma(a) - a) &= 2^{n-1}(2^{n+1} - 1 - 2^n) \\ &= 2^{n-1}(2^n - 1) \\ &= x\end{aligned}$$

as required. □

For the final investigation of HR's number theory concepts, we look at this function defined by HR:

Definition 2
Let $f(n)$ be the function defined on **N** *given by:*

$$f(n) = |\{(a, b) \in \mathbf{N} \times \mathbf{N} : a \times b = n \text{ and } a|b\}|.$$

We see that f counts the number of ways an integer can be written as a product of two divisors, the first of which divides the second. We submitted this and it was accepted into the Encyclopedia, being given sequence number A046951. We took an interest in this function because it is similar to the τ function. The first terms are:

$$1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 3, 1, 2,$$
$$1, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 1, 1, 3, 1, 1, 1, 4, 1$$

To produce more sequences, we also implemented some code which took a function sequence such as the one above and produced the 'record' sequence from it, as described on page 234 above. The record sequence is produced by finding those integers which output a bigger number than any smaller number in the original function sequence. This transformation has become a production rule in the Java version of HR mentioned in Chapter 14, but discussion of it is beyond the scope of this book. For example, the first appearance of the number 1 in the sequence above is as the output for the number 1. The

first appearance of the number 2 is as the output for number 4 and the first appearance of the number 3 is as the output for number 16. Continuing in this fashion, the first few terms of the record sequence for function f above are: $1, 4, 16, 36, 144$ and 576. HR noticed that these are square numbers and we investigated this, which led to the following interesting result.

Theorem 11
The nth integer setting the record for f as above is the square of the nth highly composite number. (Where a highly composite number has more divisors than any smaller integer – sequence A002182 in the Encyclopedia of Integer Sequences).

To prove this, we need the following lemma:

Lemma 5
For a given integer, n, let $s(n)$ be the largest square number which divides n. Then:

$$f(n) = \tau\left(\sqrt{s(n)}\right).$$

(With $f(n)$ as in Definition 2 above).

Proof of Lemma 5
Let the prime factorisation of n be $p_1^{k_1} \dots p_m^{k_m}$. Then the largest square dividing n is:

$$p_1^{2\left[\frac{k_1}{2}\right]} \dots p_m^{2\left[\frac{k_m}{2}\right]}$$

and the square root of this is:

$$p_1^{\left[\frac{k_1}{2}\right]} \dots p_m^{\left[\frac{k_m}{2}\right]}$$

where $[z]$ denotes the integer part of rational z. Now the pairs of integers whose product is n are of the form:

$$\langle a,\ b\rangle = \langle p_1^{x_1} \dots p_m^{x_m},\ p_1^{k_1 - x_1} \dots p_m^{k_m - x_m}\rangle,$$

and if $a|b$ (as dictated by $f(n)$), then $\forall\ i, x_i \leq k_i - x_i$, that is, $0 \leq x_i \leq k_i/2$. Therefore, there are $\left[\frac{k_i}{2}\right] + 1$ possibilities for x_i. So, if we count the number of possible pairs, we see that:

$$\begin{aligned} f(n) &= \left(\left[\frac{k_1}{2}\right] + 1\right) \dots \left(\left[\frac{k_m}{2}\right] + 1\right) \\ &= \tau\left(p_1^{\left[\frac{k_1}{2}\right]} \dots p_m^{\left[\frac{k_m}{2}\right]}\right) \\ &= \tau\left(\sqrt{s(n)}\right) \end{aligned}$$

as required. □

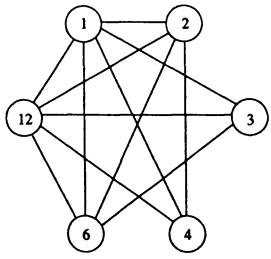
Proof of Theorem 11
Suppose that a sets a record for f. Therefore for $i = 1, 2, \ldots, a-1$, by definition, $f(a) > f(i)$, and by Lemma 5, this means that $\tau\left(\sqrt{s(a)}\right) > \tau\left(\sqrt{s(i)}\right)$ for $i = 1, 2, \ldots a-1$. Let c^2 be the largest square less than or equal to a. Then, for $j = 1, 4, 9, \ldots, (c-1)^2$, $\tau\left(\sqrt{s(a)}\right) > \tau\left(\sqrt{s(j)}\right)$. But, as each j is a square number, $\sqrt{s(j)} = \sqrt{j}$, and we see that:

$$\begin{aligned} &\tau\left(\sqrt{s(a)}\right) > \tau(1) \\ &\tau\left(\sqrt{s(a)}\right) > \tau(2) \\ &\qquad\vdots \\ &\tau\left(\sqrt{s(a)}\right) > \tau(c-1) \end{aligned} \tag{C.4}$$

If we suppose that $a > c^2$, then because c^2 is the largest square less than or equal to a, we see that $c^2 < a < (c+1)^2$. Hence the largest square dividing a cannot be larger than $(c+1)^2$ and it cannot be $(c+1)^2$ or c^2. Therefore, the largest square dividing a will be less than c^2 and $\sqrt{s(a)} < c$. But then $\sqrt{s(a)} = c - k$ for some k, and $\tau\left(\sqrt{s(a)}\right) = \tau(c-k)$ which contradicts equations (C.4). Hence it must be the case that $a = c^2$, which makes $\sqrt{s(a)} = c$. Furthermore, from equations (C.4) above, we note that $\tau(c) > \tau(c-i)$ for $i = 1, 2, \ldots, c-1$ and so c is a highly composite number and a is the square of a highly composite number. □

C.4 Divisor Graphs

For any integer we can draw a graph using the divisors as nodes and connecting any two divisors by an edge if they have a particular property. Such constructions were inspired by initial experimentation with HR working across domains, although the concept of divisor graphs cannot be attributed to HR. As an example, if we join any two divisors where one divides the other, we get the following graph for the number 12:



Asking questions about the nodes and edges of such graphs will be equivalent to asking questions about the divisors of integers. Topological questions about divisor graphs are much more interesting. In particular, we have tried to determine which integers produce divisor graphs which are planar, i.e. there is a way to draw them in the plane where no two edges cross. To solve this, we need Kuratowski's theorem [Kuratowski 30], that a graph is non-planar if and only if it has a subgraph which is homeomorphic to either K_5 or $K_{3,3}$. The details of this theorem are not needed here, as our study will be empirical, using the isplanar function of the Maple program [Abell & Braselton 94], to determine whether or not a particular graph is planar.

To proceed, we first note that if a divides b, the divisor graph for a will be a subgraph of b. This is clear because the divisors of a will be divisors of b, and they will still divide each other in the same way. Note also that if a graph G has a non-planar subgraph then G itself must be non-planar. Furthermore, we note that the actual values of the divisors is immaterial. For example, the integers 12 and 20 will have the same divisor graphs because they can both be written as p^2q for primes p and q and the way in which the divisors divide each other will be the same regardless of the values of p and q.

We say that the **prime signature** of an integer with prime factorisation $x = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$ is the set $\{n_1, n_2, \dots, n_k\}$. Hence, two integers will have isomorphic divisor graphs if they have the same prime signature. Furthermore, if integer a divides integer b, the prime signature of a will be a subset of the prime signature of b. These observations provide us with a scheme for determining which integers have planar divisor graphs. If we find the set of smallest prime signatures (in the sense that they are subsets of all other prime signatures), which have non-planar divisor graphs, then only the prime signatures which are not subsets of these will have planar divisor graphs.

We can enumerate over the number of prime factors and the powers of the primes. Using the `isplanar` function supplied with Maple, we find that integers of the form pqr have non-planar divisor graphs. In fact, the graphs produced are homeomorphic to K_3. Hence any integer divisible by three or more primes will have non-planar divisor graphs. Therefore, we only have to check integers of the form $p^n q^m$ for values of n and m. We found that integers of the form p^4 had non-planar divisor graphs, as did integers of the form p^2q^2 and p^3q. Hence we have enumerated all the cases where the divisor graph is planar: only the integer 1 and integers with one of the following prime signatures have planar divisor graphs:

$$p,\ p^2,\ p^3,\ pq,\ p^2q$$

Our initial choice for joining nodes where one divisor divides the other was arbitrary. Similarly, we could have joined nodes where one divisor was relatively prime to the other. We have used Maple to perform a similar analysis for different graph constructions, with the results summarised in the following theorem.

Theorem 12
Given an integer n, define the **divisor graph of** n to be the graph formed by writing down the divisors of n and joining any two by an edge if one divides the other. Then the only integers other than 1 which have a planar divisor graph are of the form:

$$p,\ p^2,\ p^3,\ pq,\ p^2q$$

for primes p and q.

Next, define the **co-prime graph of** n to be the graph formed by writing down the divisors of n and joining any two by an edge if they are relatively prime. Then the only integers other than 1 which have a planar co-prime graph are of the form:

$$p^i,\ pq,\ p^2q,\ p^2q^2, pqr$$

for primes p, q and r and any integer i.

Next, define the **prime signature graph of** n to be the graph formed by writing down the divisors of n and joining any two by an edge if they have the same prime signature. Then the only integers other than 1 which have a planar prime signature graph are of the form:

$$p^i,\ p^iq^j,\ p^iq^jr$$

for primes p, q and r and integers i and j.

We provide a Maple worksheet available here:

`http://www.dai.ed.ac.uk/~simonco/papers`

which verifies these and similar results about divisor graphs.

Glossary

$\tau(n)$ = number of divisors of integer n.

$\sigma(n)$ = sum of divisors of n.

$\sigma_m(n)$ = sum of the mth powers of the divisors of n.

$\phi(n)$ = number of positive integers less than n and co-prime to it.

$\pi(n)$ = number of primes less than or equal to n.

$a|b$ signifies that integer a divides integer b.

$n = |\{a : p(a)\}|$ states that n is the number of objects satisfying predicate p.

$\mathbf{N}$ is taken to be the set of natural (positive) numbers, i.e. $1, 2, 3, \ldots$

Abelian
A finite algebra A is Abelian if $\forall\ a, b \in A,\ a * b = b * a$.

Algebra
A set of elements along with a multiplication function assigning a third element to every pair of elements subject to various axioms. In this book, we mainly discuss algebras taken from finite algebraic systems, where the set of elements is finite.

AM program
The theory formation program written by Douglas Lenat [Lenat 82].

Applicability conjecture
See conjecture.

Arity
The arity of a concept is the number of columns in its data table, or alternatively, the number of variables in its predicate definition.

Axiom
A theorem which is held to be true about the objects of a domain. For instance, in group theory, one of the axioms is associativity: for all triples of elements in every group, the following is true: $a * (b * c) = (a * b) * c$.

BACON programs
The series of scientific conjecture making program written by Langley et al. [Langley *et al.* 87].

Bagai et al System
The theory formation program written by Bagai et al. [Bagai *et al.* 93].

Classically interesting
A concept or conjecture is classically interesting if it has appeared in the mathematical literature.

Compose production rule
A production rule able to form conjunctions of predicates and the composition of functions. See production rule.

Co-prime
Two integers are co-prime if they share no prime divisors. See prime numbers.

Concept
A concept in HR comprises a data table and optionally a definition. See data table, definition.

Conjecture
A conjecture in HR is a statement about one or two concepts. The conjectures can state the logical equivalence of two concepts (equivalence conjectures), the non-existence of models for a particular concept (non-existence conjectures), the implication of one concept by another (implication conjectures) or the restriction of the models of a concept to a set of finite examples (applicability conjectures). See concept, definition, models.

Construction history
The construction history of a concept is a triple of (i) old concept(s), (ii) production rule and (iii) parameterisation describing exactly how the concept was constructed from previous ones.

Data table
The data table of a concept is the set of tuples taken from the data available which satisfy the definition of the concept. In HR, concepts are represented by their data tables.

Decomposition
A way of breaking each entity into a finite set of sub-objects, e.g, decomposing a group into its elements, a graph into its nodes, or an integer into its divisors.

Definition

A written description of the predicate which is true of all the tuples in a concept's data table. There may be more than one definition for every concept. In HR, there are two formats for each definition, one in an Otter style and one in a Prolog style.

Encyclopedia of Integer Sequences

An online database of more than 60,000 integer sequences collected by Neil Sloane over 35 years, with contributions from many mathematicians, [Sloane 00]. See integer sequence.

Entity

An object of interest, such as a group, graph or integer which is present in the data HR has.

Evaluation function

A weighted sum of heuristic measures calculated to give an overall estimate of the worth of concepts. There is a similar evaluation function for conjectures. See heuristic measure.

Equivalence conjecture

See conjecture.

Exists production rule

A production rule which introduces existential quantification. See production rule.

Forall production rule

A production rule which introduces universal quantification. See production rule.

Graffiti

The conjecture making program written by Siemion Fajtlowicz [Fajtlowicz 88].

Graph

A collection of nodes and edges between nodes. In this book, we mainly discuss simple connected graphs, which have no edges between an element and itself, and a path connecting every pair of nodes.

Group

A finite algebra satisfying the associative, identity and inverse axioms. See algebra.

GT program

The theory formation program written by Susan Epstein [Epstein 88].

Heuristic measure
A calculation based on some aspect of a concept/conjecture which can be used to assess the relative worth of the concept/conjecture.

Idempotent
An element x in a finite algebra is idempotent if $x * x = x$. See algebra.

Implication conjecture
See conjecture.

Integer sequence
A list of integers (usually taken to be positive in this book). They are not necessarily increasing. See Encyclopedia of Integer Sequences.

IL program
The theory formation program written by Michael Sims [Sims 90].

MACE
The MACE model generator written by William McCune [McCune 94].

Match production rule
A production rule which produces new concepts by making inputs equal. See production rule.

Model
A tuple of objects which satisfy the definition of a concept. We also discuss our "model" of theory formation, which means the design and implementation of HR.

Negate production rule
A production rule which introduces negation by finding complements of data. See production rule.

Non-existence conjecture
See conjecture.

Otter
The Otter resolution theorem prover written by William McCune [McCune 90].

Perfect number
An integer n for which the sum of the proper divisors of n equals n.

Prime number
A positive integer with exactly two divisors.

Production rule
A construction technique which takes the data table of an old concept and turns it into a data table for a new concept. Production rules also take the definition of the old concept and produce a definition for the new concept. Each production rule has a set of pre-conditions which a concept must satisfy before the construction can take place.

Progol program
The Inductive Logic Programming machine learning program written by Stephen Muggleton [Muggleton 95].

Quasigroup
A quasigroup is a finite algebra with every element appearing in every row *and* column of the multiplication table. See algebra.

Refactorable
An integer n is called refactorable if the number of divisors of n is itself a divisor of n, e.g. 9 is refactorable because $\tau(9)$ divides 9.

Ring
An algebra with two operations (commonly called multiplication and addition). The addition operation forms an Abelian group and is distributive over the multiplication operation. See Abelian, algebra, group.

Size production rule
A production rule which counts set sizes. See production rule.

Split production rule
A production rule which instantiates variables. See production rule.

Square number
An integer of the form $m \times m$ for some $m \in \mathbf{N}$.

Sub-object
One of a finite set of objects which result from the decomposition of an entity. See decomposition.

Types
The "types" of a concept comprises the list of the types of objects and sub-objects in the columns of the data table of the concept.

Bibliography

[Abell & Braselton 94] M Abell and J Braselton. *Maple V by Example.* Associated Press Professional, 1994.

[Anderson 89] M Anderson. A critical evaluation of Lenat's AM program. Technical Report TR 89-09-19, Department of Computer Science and Engineering, University of Washington, 1989.

[Appel & Haken 77] K Appel and W Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[Backus 87] J Backus. Can programming be liberated from the von Neumann style? In *ACM Turing Award Lectures, the First Twenty Years*, pages 63–130. Addison-Wesley, 1987.

[Bagai *et al.* 93] R Bagai, V Shanbhogue, J Żytkow, and S Chou. Automatic theorem generation in plane geometry. In *LNAI 689*, pages 415–424. Springer-Verlag, 1993.

[Bailey 98] D Bailey. Finding new mathematical identities via numerical computations. *ACM SIGNUM*, 33(1):17–22, 1998.

[Bailey *et al.* 97] D Bailey, M Borwein, P Borwein, and S Plouffe. The quest for π. *Mathematical Intelligencer*, 19(1):50–57, 1997.

[Baker *et al.* 90] A Baker, B Bollobàs, and A Hajnal. *A Tribute to Paul Erdős.* Cambridge University Press, 1990.

[Barker-Plummer 92] D Barker-Plummer. Gazing: An approach to the problem of definition and lemma use. *Journal of Automated Reasoning*, 8(3):311–344, 1992.

[Beiler 96] A Beiler. *Recreations in the Theory of Numbers: The Queen of Mathematics Entertains. 2nd edition.* Dover, 1996.

[Bell 34] E Bell. Exponential numbers. *American Mathematics Monthly*, 41:411–419, 1934.

[Boden 92] A Boden, M. *The Creative Mind.* Abacus, 1992.

[Boden 94] M Boden, editor. *Dimensions of Creativity.* MIT Press, 1994.

[Boyer & Moore 79] S Boyer and J Moore. *Computational Logic.* Academic Press, 1979.

[Brachman & Levesque 85] R Brachman and H Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann, 1985.

[Bradshaw *et al.* 80] G Bradshaw, P Langley, and H Simon. BACON.4: The discovery of intrinsic properties. Technical Report 420, Department of Psychology, Carnegie-Mellon University, 1980.

[Buchanan 66] B Buchanan. *Logics of Scientific Discovery.* Unpublished PhD thesis, Department of Philosophy, Michigan State University, 1966.

[Buchanan 00] B Buchanan. Creativity at the meta-level. In *Keynote speech at AAAI-2000.* Available on audio tape from the American Association for Artificial Intelligence, 2000.

[Bundy 83] A Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, 1983.

[Bundy 85] A Bundy. Discovery and reasoning in mathematics. In A Joshi, editor, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1221–1230, 1985.

[Bundy 98] A Bundy. *Personal Email Communication.* 1998.

[Bundy *et al.* 98] A Bundy, S Colton, and T Walsh. HR - automatic concept formation in finite algebras. In *Proceedings of the ECAI Machine Discovery Workshop*, 1998.

[Bundy *et al.* 02] A Bundy, S Colton, R McCasland, and T Walsh. Semi-automated discovery in zariski spaces (a proposal). In *Proceedings of the Automated Reasoning Workshop*, 2002.

[Caporossi & Hansen 97] G Caporossi and P Hansen. Variable neighbourhood search for extremal graphs. 1. The AutoGraphiX system. Technical Report Les Cahiers du GERAD G-97-41, École des Hautes Études Commerciales, Montréal, 1997.

[Caporossi & Hansen 99] G Caporossi and P Hansen. Finding relations in polynomial time. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 780–785, 1999.

[Chaitin 98] G Chaitin. *The Limits of Mathematics.* Springer-Verlag, 1998.

[Chou 84] S Chou. *Mechanical Theorem Proving.* D. Reidel Publishing Company, Dordrecht, Netherlands, 1984.

[Chou 85] S Chou. Proving and discovering geometry theorems using Wu's method. Technical Report 49, Computing Science, University of Austin at Texas, 1985.

[Chou *et al.* 00] S Chou, X Gao, and J Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3):219–246, 2000.

[Chung 88] F Chung. The average distance and the independence number. *Journal of Graph Theory*, 12(2):229–235, 1988.

[Cohen 95] W Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the 12th International Conference*, pages 115–123, 1995.

[Colton & Bundy 99] S Colton and A Bundy. On the notion of interestingness in automated mathematical discovery. In *Proceedings of the AISB'99 Symposium on AI and Scientific Creativity*, 1999.

[Colton & Dennis 02] S Colton and L Dennis. The NumbersWithNames program. In *Proceedings of the 7th Symposium on Artificial Intelligence and Mathematics*, 2002.

[Colton & Miguel 01] S Colton and I Miguel. Constraint generation via automated theory formation. In *Proceedings of the 7th International Conference on the Principles and Practice of Constraint Programming*, pages 572–576, 2001.

[Colton & Sutcliffe 02] S Colton and G Sutcliffe. Automatic generation of benchmark problems for automated theorem proving systems. In *Proceedings of the 7th Symposium on Artificial Intelligence and Mathematics*, 2002.

[Colton 99] S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences,* `www.research.att.com/~njas/sequences/JIS`, 2, 1999.

[Colton 00a] S Colton. Assessing exploratory theory formation programs. In *Proceedings of the AAAI-2000 workshop on new research directions in machine learning*, 2000.

[Colton 00b] S Colton. Automated plugging and chugging. In M Kerber and M Kohlhase, editors, *Computation and Automated Reasoning*, pages 247–248. A. K. Peters, 2000.

[Colton 01a] S Colton. Automated theorem discovery: A future direction for theorem provers. In *Proceedings of the IJCAR-01 Workshop on Future Directions in Automated Reasoning*, 2001.

[Colton 01b] S Colton. Experiments in meta-theory formation. In *Proceedings of the AISB'01 Symposium on Creativity in Arts and Science*, 2001.

[Colton 01c] S Colton. Mathematics: A new domain for datamining? In *Proceedings of the IJCAI-01 Workshop on Knowledge Discovery from Distributed, Dynamic, Heterogenous, Autonomous Sources*, 2001.

[Colton 02a] S Colton. An application-based comparison of automated theory formation and inductive logic programming. *To appear in Linkoping Electronic Articles in Computer and Information Science, (special issue: Proceedings of Machine Intelligence 17)*, 2002.

[Colton 02b] S Colton. Automated puzzle generation. In *Proceedings of the AISB'02 Symposium on Creativity in the Arts and Science*, 2002.

[Colton 02c] S Colton. The HR program for theorem generation. In *Proceedings of CADE-18*, 2002.

[Colton 02d] S Colton. Making conjectures about maple functions. In *Proceedings of Calculemus 02, Systems for Integrated Computation and Deduction*, 2002.

[Colton *et al.* 97] S Colton, S Cresswell, and A Bundy. The use of classification in automated mathematical concept formation. In *Proceedings of SimCat 1997: An Interdisciplinary Workshop on Similarity and Categorisation*. University of Edinburgh, 1997.

[Colton *et al.* 99a] S Colton, A Bundy, and T Walsh. Automated conjecture making in mathematics. *EPSRC Grant Proposal, GR/M98012*, 1999.

[Colton *et al.* 99b] S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 786–791, 1999.

[Colton *et al.* 00a] S Colton, A Bundy, and T Walsh. Agent based cooperative theory formation in pure mathematics. In *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.

[Colton *et al.* 00b] S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, pages 183–190, 2000.

[Colton *et al.* 00c] S Colton, A Bundy, and T Walsh. Automatic invention of integer sequences. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 558–563, 2000.

[Colton *et al.* 00d] S Colton, A Bundy, and T Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.

[Colton *et al.* 01a] S Colton, L Drake, A Frisch, I Miguel, and T Walsh. Automated generation of implied constraints: Initial progress. In *Proceedings of the Automated Reasoning Workshop*, 2001.

[Colton *et al.* 01b] S Colton, A Pease, and G Ritchie. The effect of input knowledge on creativity. In *Proceedings of the ICCBR'01 Workshop on Creative Systems*, 2001.

[Colton *et al.* 02] S Colton, R McCasland, A Bundy, and T Walsh. Automated theory formation for tutoring tasks in pure mathematics. In *Proceedings of the CADE workshop on the Role of Automated Deduction in Mathematics*, 2002.

[Conway & Norton 79] J Conway and S Norton. Monstrous moonshine. *Bulletin of the London Mathematical Society*, 11:308–339, 1979.

[Conway 76] J Conway. *On numbers and games*. Academic Press, 1976.

[Davis & Putnam 60] M Davis and H Putnam. A computing procedure for quantification theory. *Associated Computing Machinery*, 7:201–215, 1960.

[Denzinger & Gramlich 88] J Denzinger and B Gramlich. Efficient AC-matching using constraint propagation. Technical Report SR-88-15, University of Kaiserslautern, SEKI, 1988.

[Ekstrom-Meredith 87] M Ekstrom-Meredith. *Seek-Whence: A Model of Pattern Perception.* Unpublished PhD thesis, Department of Computer Science, Indiana University, 1987.

[Epstein 83] S Epstein. *Knowledge Representation in Mathematics: A Case Study in Graph Theory.* Unpublished PhD thesis, Department of Computer Science, Rutgers University, 1983.

[Epstein 87] S Epstein. On the discovery of mathematical theorems. In *Proceedings of the 10th International Joint Conference on Artificial Intellignce*, pages 194–197, 1987.

[Epstein 88] S Epstein. Learning and discovery: One system's search for mathematical knowledge. *Computational Intelligence*, 4(1):42–53, 1988.

[Epstein 91] S Epstein. Knowledge representation for mathematical discovery, three experiments in graph theory. *Applied Intelligence*, 1(1):7–33, 1991.

[Epstein 99] S Epstein. *Personal Email Communication.* 1999.

[Erdős & Selfridge 75] P Erdős and J Selfridge. The product of consecutive integers is never a power. *Illinois Journal of Mathematics*, 19:292–301, 1975.

[Erdős *et al.* 91] P Erdős, S Fajtlowicz, and W Staton. Degree sequences in triangle-free graphs. *Discrete Mathematics*, 92:85–88, 1991.

[Ernest 98] P Ernest. *Social Constructivism as a Philosophy of Mathematics.* SUNY Press, 1998.

[Ernest 99] P Ernest. Is mathematics discovered or invented? *Philosophy of Mathematics Education*, 12, 1999.

[Euler 36] L Euler. Solution problematis geometrian situs perinentis. *Opera Omnia*, 7(1):128–140, 1736.

[Fajtlowicz 88] S Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics 72*, 23:113–118, 1988.

[Fajtlowicz 99] S Fajtlowicz. The writing on the wall. Unpublished preprint, available from http://math.uh.edu/~clarson/, 1999.

[Franke *et al.* 99] A Franke, S Hess, C Jung, M Kohlhase, and V Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5(3):156–187, 1999. Special Issue on Integration of Deduction Systems.

[Furse 90] E Furse. Why did AM run out of steam? Technical Report CS-90-4, Department of Computer Studies, University of Glamorgan, 1990.

[Furse 99] E Furse. *Personal communication.* 1999.

[Gap 00] Gap. *GAP Reference Manual.* The GAP Group, School of Mathematical and Computational Sciences, University of St. Andrews, 2000.

[Ghiselin 96] B Ghiselin, editor. *The Creative Process.* University of California Press, 1996.

[Gorenstein 82] D Gorenstein. *Finite Simple Groups: An Introduction to Their Classification.* Plenum Press, New York, 1982.

[Haase 86a] K Haase. Discovery systems. In *Proceedings of the European Conference on Artificial Intelligence*, pages 546–555, 1986.

[Haase 86b] K Haase. Discovery systems. Technical Report 898, Department of Computer Science, MIT, 1986.

[Hardy & Wright 38] G Hardy and E Wright. *The Theory of Numbers.* Oxford University Press, 1938.

[Hardy 27] G Hardy, editor. *S Ramanujan - Collected Papers.* Cambridge University Press, 1927.

[Hardy 92] G Hardy. *A Mathematician's Apology.* Cambridge University Press, 1992.

[Hilbert & Cohn-Vossen 52] D Hilbert and S Cohn-Vossen. *Geometry and the Imagination.* Chelsea, 1952.

[Hirschhorn 95] M Hirschhorn. A proof in the spirit of Zeilberger of an amazing identity of Ramanujan. *Mathematics Magazine*, 68:3, 1995.

[Hoffman 99] P Hoffman. *The man who loved only numbers.* Fourth Estate, 1999.

[Hofstadter 95] D Hofstadter. *Fluid Concepts and Creative Analogies.* Basic Books, 1995.

[Humphreys 96] J Humphreys. *A Course in Group Theory.* Oxford University Press, 1996.

[ILF 99] ILF. The ILF server. *http://www-irm.mathematik.hu-berlin.de/ilf-serv/*, 1999.

[Jackson 92] P Jackson. Computing prime implicates incrementally. In *Proceedings of CADE-11, LNCS 607*, pages 253–267. Springer-Verlag, 1992.

[Jones 86] V Jones. A polynomial invariant for links via von Neumann algebras. *Bulleting of the American Mathematical Society*, 129:103–112, 1986.

[Kennedy & Cooper 90] R Kennedy and C Cooper. Tau numbers, natural density and Hardy and Wright's theorem 437. *International Journal of Mathematics and Mathematical Sciences*, 13:383–386, 1990.

[Kerber 91] M Kerber. Useful properties of a frame-based representation of mathematical knowledge. Technical Report SR-91-06, Universitat Des Saarlandes, Fachbereich Informatik, 1991.

[Kerber 92] M Kerber. *On the Representation of Mathematical Concepts and their Translation into First Order Logic.* Unpublished PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, 1992.

[Kitcher 83] P Kitcher. *The Nature of Mathematical Knowledge.* Oxford University Press, 1983.

[Kohlhase & Franke 00] M Kohlhase and A Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation, Special Issue on the Integration of Computer Algebra and Deduction Systems*, 11:1–37, 2000.

[Konrad & Wolfram 99] K Konrad and D Wolfram. System description: Kimba, a model generator for many-valued first order logics. In *Proceedings of CADE-16, LNAI 1632*, pages 282–286. Springer-Verlag, 1999.

[Koutsofios & North 98] E Koutsofios and C North. Dot user's guide. Technical report, AT+T Bell Labs, Murray Hill, NJ, 1998.

[Krattenthaler 91] C Krattenthaler. Advanced determinant calculus. Technical report, Institute of Mathematics, University of Vienna, 1991.

[Kronecker 70] L Kronecker. Auseinandersetzung einiger eigenschaften der klassenanzahl idealer komplexer zahlen. *Berlin Monatsber*, pages 881–889, 1870.

[Kuhn 70] T Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, 1970.

[Kuratowski 30] G Kuratowski. Sur la problème des courbes gauches en topologie. *Fund. Math*, 15-16, 1930.

[Laburthe 00] F Laburthe. Choco: implementing a CP kernel. In *CP00 Workshop on Techniques for Implementing Constraint programming Systems (TRICS)*, 2000.

[Lakatos 76] I Lakatos. *Proofs and Refutations: The logic of mathematical discovery.* Cambridge University Press, 1976.

[Langley 79] P Langley. Rediscovering physics with BACON.3. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pages 505–507. Morgan Kaufmann, 1979.

[Langley *et al.* 80a] P Langley, G Bradshaw, and H Simon. BACON.5: The discovery of conservation laws. Technical Report 430, Department of Psychology, Carnegie-Mellon University, 1980.

[Langley *et al.* 80b] P Langley, G Bradshaw, and H Simon. Rediscovering chemistry with BACON.4. Technical Report 423, Department of Psychology, Carnegie-Mellon University, 1980.

[Langley *et al.* 87] P Langley, H Simon, G Bradshaw, and J Żytkow. *Scientific Discovery - Computational Explorations of the Creative Processes.* MIT Press, 1987.

[Larson 99] C Larson. Intelligent machinery and discovery in mathematics. Unpublished preprint, available from http://math.uh.edu/~clarson/, 1999.

[Lenat & Brown 84] D Lenat and J Brown. Why AM and Eurisko to work. *Artificial Intelligence*, 23:269–294, 1984.

[Lenat 76] D Lenat. *AM: An Artificial Intelligence approach to discovery in mathematics.* Unpublished PhD thesis, Stanford University, 1976.

[Lenat 82] D Lenat. AM: Discovery in mathematics as heuristic search. In D Lenat and R Davis, editors, *Knowledge-Based Systems in Artificial Intelligence.* McGraw-Hill Advanced Computer Science Series, 1982.

[Lenat 83] D Lenat. Eurisko: A program which learns new heuristics and domain concepts. *Artificial Intelligence*, 21:61–98, 1983.

[Livingstone *et al.* 99] G Livingstone, B Buchanan, and J Rosenberg. Autonomous discovery in empirical domains. Technical Report CS-99-12, Department of Computer Science, University of Pittsburgh, 1999.

[Lovàsz 93] L Lovàsz. Paul Erdos is eighty. In D Miklòs, V Sós, and T Szony, editors, *Paul Erdős is Eighty*, volume 1. Bolyai Society Mathematical Studies, 1993.

[McCasland *et al.* 98] R McCasland, M Moore, and P Smith. An introduction to Zariski spaces over Zariski topologies. *Rocky Mountain Journal of Mathematics*, 28:1357–1369, 1998.

[McCasland *et al.* 02] R McCasland, S Colton, A Bundy, and T Walsh. Applying HR to the study of Zariski spaces. *EPSRC Grant Proposal, GR/R84559/01*, 2002.

[McCune & Padmanabhan 96] W McCune and R Padmanabhan. *Automated Deduction in Equational Logic and Cubic Curves, LNAI 1095.* Springer-Verlag, 1996.

[McCune 90] W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.

[McCune 92] W McCune. Automated discovery of new axiomatizations of the left group and right group calculi. *Journal of Automated Reasoning*, 9(1):1–24, 1992.

[McCune 93] W McCune. Single axioms for groups and abelian groups with various operations. *Journal of Automated Reasoning*, 10(1):1–13, 1993.

[McCune 94] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.

[McCune 97] W McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[McCune 00] W McCune. *Personal Communication.* 2000.

[McLeod & Adams 89] D McLeod and V Adams. *Affect and Mathematical Problem Solving: A New Perspective.* Springer-Verlag, 1989.

[Meier *et al.* 02] A Meier, V Sorge, and S Colton. Employing theory formation to guide proof planning. In *Proceedings of Calculemus 02, Systems for Integrated Computation and Deduction*, 2002.

[Meschowski 64] H Meschowski. *Ways of Thought of Great Mathematicians.* Holden-Day, 1964.

[Michalski & Larson 77] R Michalski and J Larson. Inductive inference of VL decision rules. In *Proceedings of the Workshop in Pattern-Directed Inference Systems (Published in SIGART Newsletter ACM, No. 63)*, pages 38–44, 1977.

[Miller 76] G L Miller. On the $n^{log_2 n}$ isomorphism technique. Technical Report TR17, The University of Rochester, 1976.

[Mitchell 96] M Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1996.

[Morales 85] E Morales. DC: a system for the discovery of mathematical conjectures. Unpublished M.Sc. thesis, University of Edinburgh, 1985.

[Muggleton & De Raedt 94] S Muggleton and L De Raedt. Inductive Logic Programming: Theory and methods. *Logic Programming*, 19-20(2):629–679, 1994.

[Muggleton & Page 94] S Muggleton and D Page. A learnability model for universal representations. Technical Report PRG-TR-3-94, Computing Laboratory, University of Oxford, 1994.

[Muggleton 91] S Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[Muggleton 95] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[Newell & Simon 72] A Newell and H Simon. *Human Problem Solving.* Prentice Hall, 1972.

[Padmanabhan & McCune 95] R Padmanabhan and W McCune. Single identities for ternary boolean algebras. *Computers and Mathematics with Applications*, 29(2):13–16, 1995.

[Parshall 98] K Parshall. The art of algebra from Al-Khwarizmi to Viète: A study in the natural selection of ideas. *History of Science*, 26(72):129–164, 1998.

[Pease *et al.* 00] A Pease, S Colton, A Smaill, and J Lee. Lakatos and machine creativity. In *Proceedings of the ECAI workshop on creative systems*, 200.

[Pease *et al.* 01] A Pease, D Winterstein, and S Colton. Evaluating machine creativity. In *Workshop on Creative Systems, 4th International Conference on Case Based Reasoning*, 2001.

[Penrose 89] R Penrose. *The Emperor's New Mind.* Oxford University Press, 1989.

[Pistori & Wainer 99] H Pistori and J Wainer. Automatic theory formation in graph theory. In *Argentine Symposium on Artificial Intelligence*, pages 131–140, 1999.

[Pólya 54] G Pólya. *Mathematics and Plausible Reasoning; Vol. 1. Induction and Analogy in Mathematics; Vol. 2. Patterns of Plausible Inference.* Princeton University Press, 1954.

[Pólya 81] G Pólya. *Mathematical Discovery.* Wiley, 1981.

[Pólya 88] G Pólya. *How to Solve it.* Princeton University Press, 1988.

[Popper 72a] K Popper. *Conjectures and Refutations - The Growth of Scientific Knowledge.* Routledge and Kegan Paul, 1972.

[Popper 72b] K Popper. *The Logic of Scientific Discovery.* Hutchinson, 1972.

[Quinlan 93] R Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Ramesh *et al.* 97] A Ramesh, G Becker, and N Murray. CNF and DNF considered harmful for computing prime implicants/implicates. *Journal of Automated Reasoning*, 18(3):337–356, 1997.

[Regin 96] J Regin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 362–367. American Association for Artificial Intelligence, 1996.

[Richardson *et al.* 98] J Richardson, A Smaill, and I Green. System description: proof planning in higher-order logic with λ-clam. In *Proceedings of CADE-15, LNAI 1421*, pages 129–133. Springer-Verlag, 1998.

[Ritchie & Hanna 84] G Ritchie and F Hanna. AM: A case study in methodology. *Artificial Intelligence*, 23:249–268, 1984.

[Ritchie 94] G Ritchie. Learning from AM. In J Johnson, S McKee, and A Vella, editors, *Artificial Intelligence in Mathematics*, pages 55–66. Oxford University Press, 1994.

[Ritchie 01] G Ritchie. Assessing creativity. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 2001.

[Robinson 65] J Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Saaty & Kainen 86] T Saaty and P Kainen. *The Four-Color Problem: Assaults and Conquest.* Dover, 1986.

[Schaffer 90] C Schaffer. Domain-independent scientific function finding. Technical Report LCSR-TR-149, Department of Computer Science, Rutgers University, 1990.

[Selden & Selden 96] A Selden and J Selden. Of what does mathematical knowledge consist? (Research sampler). *MAA Online*, 1996.

[Shen 87] W Shen. Functional transformations in AI discovery systems. Technical Report CMU-CS-87-117, Computer Science Department, CMU, 1987.

[Simon & Newell 58] H Simon and A Newell. Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(1):1–10, 1958.

[Simon 00] H Simon. *Personal Communication.* 2000.

[Sims & Bresina 89] M Sims and J Bresina. Discovering mathematical operator definitions. In *Machine Learning: Proceedings of the 6th International Conference*, pages 308–313. Morgan Kaufmann, 1989.

[Sims 90] M Sims. *IL: An Artificial Intelligence approach to theory formation in mathematics.* Unpublished PhD thesis, Rutgers University, 1990.

[Sims 98] M Sims. Explanation based learning and discovery in IL. In *Proceedings of the second annual AI research forum, NASA Ames, Moffett Field, CA.*, 1998.

[Singh 97] S Singh. *Fermat's Last Theorem.* Fourth Estate, 1997.

[Slaney 92] J Slaney. FINDER (finite domain enumerator): Notes and guide. Technical Report TR-ARP-1/92, Australian National University Automated Reasoning Project, 1992.

[Slaney 94] J Slaney. Finder: Finite domain enumerator – system description. In A Bundy, editor, *Proceedings of CADE-12, LNAI 814*, pages 798–801. Springer-Verlag, 1994.

[Sloane & Bernstein 95] N Sloane and M Bernstein. Some canonical sequences of integers. *Linear Algebra and its Applications*, 226-228:57–72, 1995.

[Sloane & Plouffe 95] N Sloane and S Plouffe. *The Encyclopedia of Integer Sequences.* Academic Press, 1995.

[Sloane 98] N Sloane. My favorite integer sequences. In *Proceedings of the International Conference on Sequences and Applications*, 1998.

[Sloane 00] N Sloane. The Online Encyclopedia of Integer Sequences. *http://www.research.att.com/~njas/sequences*, 2000.

[Steel 99] G Steel. Cross domain concept formation using HR. Unpublished M.Sc. thesis, Division of Informatics, University of Edinburgh, 1999.

[Steel *et al.* 00] G Steel, S Colton, A Bundy, and T Walsh. Cross domain mathematical concept formation. In *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.

[Stewart 89] I Stewart. *Galois Theory.* Chapman and Hall Mathematics, 1989.

[Stuyvaert 97] M Stuyvaert. Extraction de la racine carré d'un nombre entier. *Mathesis*, (2) 7:161–162, 1897.

[Sylow 72] L Sylow. Théorèmes sur les groupes de substitutions. *Mathematische Annalen*, 5:584–594, 1872.

[Trudeau 76] R Trudeau. *Introduction to Graph Theory.* Dover, 1976.

[Trybulec 89] A Trybulec. Tarski grothendieck set theory. *Journal of Formalised Mathematics*, 0, 1989.

[Tsang 93] E Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[Valdés-Pérez 99] R Valdés-Pérez. Principles of human computer collaboration for knowledge discovery in science. *Artificial Intelligence*, 107(2):335–346, 1999.

[Voronkov 95] A Voronkov. The anatomy of Vampire. *Journal of Automated Reasoning*, 15(2):237–265, 1995.

[Walsh 98] T Walsh. *Personal Email Communication.* 1998.

[Weidenbach 99] C Weidenbach. Spass: Combining superposition, sorts and splitting. In Robinson A and Voronkov A, editors, *Handbook of Automated Reasoning.* Elsevier Science, 1999.

[Wilder 68] R Wilder. *Evolution of Mathematical Concepts, An Elementary Study.* John Wiley and Sons, 1968.

[Wiles 95] A Wiles. Modular elliptic curves and Fernat's last theorem. *Annals of Mathematical Logic*, 141(3):443–551, 1995.

[Wilson & Sloane 99] D Wilson and N Sloane. *Personal Email Communication.* 1999.

[Wilson 00] D Wilson. *Personal Email Communication.* 2000.

[Wiseman 81] Wiseman. Results of the 1981 trillion credit squadron competition. *Travellers Aid Soceity*, 1981.

[Wolfram 99] S Wolfram. *The Mathematica Book, Fourth Edition.* Wolfram Media/Cambridge University Press, 1999.

[Wu 84] W Wu. Basic principles of mechanical theorem proving in geometries. *Journal of System Sciences and Mathematical Sciences*, 4(3):207–235, 1984.

[Yugami 95] N Yugami. Theoretical analysis of the Davis-Putnam procedure and propositional satisfiability. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 282–288, 1995.

[Zeilberger 98] D Zeilberger. Enumeration schemes, and more importantly, their automatic generation. *Annals of Combinatorics*, 2:185–195, 1998.

[Zeilberger 99] D Zeilberger. Automated counting of LEGO towers. *Difference Equations Applications*, 5:323–333, 1999.

[Zeitz 99] P Zeitz. *The Art and Craft of Problem Solving.* John Wiley and Sons, 1999.

[Zhang 99] J Zhang. MCS: Model-based conjecture searching. In *Proceedings of CADE-16, LNAI 1632*, pages 393–397. Springer-Verlag, 1999.

[Zimmer *et al.* 02] J Zimmer, A Franke, S Colton, and G Sutcliffe. Integrating HR and tptp2X into MathWeb to compare automated theorem provers. In *Proceedings of the CADE workshop on Problems and Problem sets*, 2002.

Index