Manual for the tool Carpa

Hans Zantema\textsuperscript{1,2}

\textsuperscript{1} Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, email: H.Zantema@tue.nl
\textsuperscript{2} Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

\textbf{Abstract.} The tool \textbf{Carpa} (Counter examples in Abstract Rewriting Produced Automatically) automatically tries to find finite counter examples for any given set of rewriting properties.

The input for the tool \textbf{Carpa} (Counter examples for Abstract Rewriting Produced Automatically) is a list of properties of binary relations. On such an input the tool either builds a set of binary relations on the specified number of elements that satisfies these properties, or shows that this is impossible. The tool \textbf{Carpa} can be downloaded from \url{http://www.win.tue.nl/~hzantema/carpa.html} including the source code, a Linux executable, a file \texttt{Readme} with basic instructions, and encodings of several examples.

The input for \textbf{Carpa} always starts by two numbers \(n, m\), each on a separate line, possibly followed by comment. Here \(n = \#A\) is the cardinality of the set \(A\) on which we search for binary relations. The number \(m\) is the number of basic relations in the specification, internally referred to by the numbers 1, \ldots, \(m\). So if we look for a single relation \(R\) with a given set of properties we choose \(m = 1\), and if we look for two relations \(R\) and \(S\) with a given set of properties we choose \(m = 2\).

The rest of the input consists of a number of lines each being either a predicate or an assignment. In the following \(R, S\) refer to binary relations on \(A\), and \(x, y\) refer to elements of \(A\). The possible predicates are

- \texttt{subs}, where \texttt{subs}(\(R, S\)) means that \(R \subseteq S\),
- \texttt{nsubs}, where \texttt{nsubs}(\(R, S\)) means that \(\lnot(R \subseteq S)\),
- \texttt{disj}, where \texttt{disj}(\(R, S\)) means that \(R \cap S = \emptyset\),
- \texttt{trans}, where \texttt{trans}(\(R\)) means that \(R\) is transitive,
- \texttt{ntrans}, where \texttt{ntrans}(\(R\)) means that \(R\) is not transitive,
- \texttt{irr}, where \texttt{irr}(\(R\)) means that \(R\) is irreflexive,
- \texttt{nirr}, where \texttt{nirr}(\(R\)) means that \(R\) is not irreflexive,
- \texttt{symm}, where \texttt{symm}(\(R\)) means that \(R\) is symmetric,
- \texttt{sn}, where \texttt{sn}(\(R\)) means that \(R\) is terminating,
- \texttt{nsn}, where \texttt{nsn}(\(R\)) means that \(R\) is not terminating,
- wn, where wn(R) means that R is weakly normalizing (every element has at least one normal form),
- nwn, where nwn(R) means that R is not weakly normalizing,
- cr, where cr(R) means that R is confluent,
- ncr, where ncr(R) means that R is not confluent,
- wcr, where wcr(R) means that R is locally confluent,
- nwcr, where nwcr(R) means that R is not locally confluent,
- un, where un(R) means that R has the unique normal form property (every element has at least one normal form),
- nun, where nun(R) means that R does not have the unique normal form property,
- compl, where compl(R) means that R is complete,
- nf, where nf(x, R) means that x is a normal form with respect to R, and
- red, where red(x, y, R) means that (x, y) ∈ R.
- nrrules, where nrrules(R, j) means that R has at most j elements, and
- nriter, where nriter(j) means that the number k used to define transitive closures is replaced by j; its default value is ⌈log₂ n⌉. This default value is always safe, but in some cases smaller values may be appropriate.

Assignments always consist of a variable name followed by the symbol ‘=’, followed by an operation applied on a number of arguments. Here the variable names are always ‘x’ followed by a number. The possible operations are

- union, where union(R, S) represents the relation R ∪ S,
- inters, where inters(R, S) represents the relation R ∩ S,
- comp, where comp(R, S) represents the relation R · S,
- peak, where peak(R, S) represents the relation R⁻¹ · S,
- val, where val(R, S) represents the relation R · S⁻¹,
- inv, where inv(R) represents the inverse R⁻¹ of R,
- tc, where tc(R) represents the transitive closure R⁺ of R,
- rc, where rc(R) represents the reflexive closure R ∪ I of R, and
- trc, where trc(R) represents the transitive reflexive closure R* of R.

Here the relations R, S should be either one of the basic relations, numbered 1, . . . , m, or a variable name that has been defined in an earlier assignment.

Space symbols are not allowed; lines starting with a space symbol are considered as comment.

Examples

Finding a locally confluent irreflexive relation on four elements that is not confluent can be done by the following input:

4 (nr of elements)
1 (nr of basic relations)
wcr(1)
ncr(1)
irr(1)
Alternatively, avoiding \texttt{wcr} and \texttt{cr} in order to introduce \texttt{trc(1)} internally only once, for the same task the following input can be chosen:

\begin{verbatim}
4
1
x1=trc(1)
x2=peak(1,1)
x3=val(x1,x1)
subs(x2,x3)
x4=peak(x1,x1)
nsubs(x4,x3)
irr(1)
\end{verbatim}

Both versions give as output the desired relation:

\begin{verbatim}
Relation 1:
(1,2)
(1,3)
(2,1)
(2,4)
\end{verbatim}

which coincides with the well-known example of a locally confluent relation that is not confluent.

By the next example we look for two complete relations $R$ and $S$ satisfying $R^{-1} \cdot S \subseteq S \cdot R^* \cdot (R^{-1})^*$ for $R$ being 1 and $S$ being 2, on 8 elements, for which the element 1 has two distinct normal forms 2 and 3 with respect to the union of $R$ and $S$.

As the input we define

\begin{verbatim}
8
2
compl(1)
compl(2)
x1=union(1,2)
uf(2,x1)
uf(3,x1)
x2=tc(x1)
red(1,2,x2)
red(1,3,x2)
x1=trc(1)
x2=comp(2,x1)
x3=peak(1,2)
x4=val(x2,x1)
subs(x3,x4)
\end{verbatim}

On this input within a few seconds \texttt{Carpa} generates the output

3
Relation 1:
(1,4)
(5,4)
(7,6)
(8,6)
Relation 2:
(1,3)
(4,3)
(4,8)
(5,7)
(6,2)
(6,5)
(7,2)
(8,1)

that indeed can be checked to have the given properties.