ELEVENTH WORKSHOP ON AUTOMATED REASONING:

BRIDGING THE GAP BETWEEN THEORY AND PRACTICE

(co-located with AISB'04)

E-Checker: A prototype tool for investigating some properties of multivalent logic systems.

David Anderson
Department of Information Systems and Computer Applications
University of Portsmouth
Portsmouth PO1 2EG
Tel / Fax: +44 1489 780 876
cdpa@btinternet.com

Peter W. Pearson
Department of Information Systems and Computer Applications
University of Portsmouth
Portsmouth PO1 2EG
Tel / Fax: +44 1489 780 876
cam10005@port.ac.uk

Abstract

Building on earlier work carried out in the area of automated multivalued paraconsistent reasoning systems [see Anderson: ARW 9 & ARW10], the authors present the prototype of a software tool for investigating some of the properties of MVL systems. The paper briefly outlines the domain in which the tool is being used and presents the main features offered in the current incarnation. Details of two interesting systems of logic tested successfully by E-Checker are given. At this stage in its development E-Checker, is incomplete, but already proving very valuable.

Let |- be a relation of logical consequence. It is common[1] to say that |- is *explosive* iff for arbitrary formulae q and r, (q & ¬*q*} |- r [2]. Classical logic, intuitionistic logic, and most other standard logics are explosive but ordinary human reasoning is not. The explosive character of standard logic has long been a cause of concern among mathematicians and logicians[3] and is something which should be explicitly taken into account when developing expert systems or automated reasoning systems which may have to deal with inconsistent information.

There are in the literature a number of interesting attempts to avoid or otherwise deal with the problem of logical explosiveness and prominent among these are the work Asenjo[4], da Costa [5], Dunn[6], Belnap[7] and Priest[8].

It is common practice for logicians to confine their discussions about logical operators to a small subset of those which are constructible in principle. Usually interest centres around negation, conjunction, disjunction, conditional and bi-conditional and it is common although by no means universal[9] to stipulate:

Definition 1: Disjunction P V Q $=^{df}$ ¬(¬P & ¬Q)
Definition 2: Conditional P => Q $=^{df}$ ¬(P & ¬Q)
Definition 3: Biconditional P <=> Q $=^{df}$ (P => Q) & (Q => P)

Perhaps the simplest way[10] to deal with logical explosiveness is to develop a multi-valued logic approach. However this strategy is not without its problems. One difficulty is the sharp increase in complexity which occurs as the number of possible valuations increases. The maximum number of distinct n-ary operators which are constructible within a given system permitting V possible valuations is given by:

$$\sum_{k=1}^{k=n} V^{(v^n)}$$

Valuations (V)	Unary Operators (n=1)	Binary Operators (n=2)	Distinct Operators (n=1) + (n=2)	Distinct Systems Based on 1 Unary & 1 Binary (n=1) x (n=2)
2	4	16	20	64
3	27	19,683	19,710	531,441
4	256	4,294,967,296	4,294,967,552	1,099,511,627,776

Issues which need to be addressed within a multi-valued logic approach are how, if at all, to interpret the values and which of those values to designate, anti-designate or leave non-designated[11].

The need was identified to develop a software tool to assist in the exploration of this logical space and in response, E-Checker is being developed at University of Portsmouth by members of the Artificial Intelligence group. The software is still a prototype but sufficient progress has been made to see that the final version will be of considerable assistance to future developments.

The particular interest of our group has been to explore paraconsistent systems using a variety of designation schemes, but E-Checker is not restricted to contradiction-tolerant systems alone but may be applied to any four valued system of the sort stipulated above.

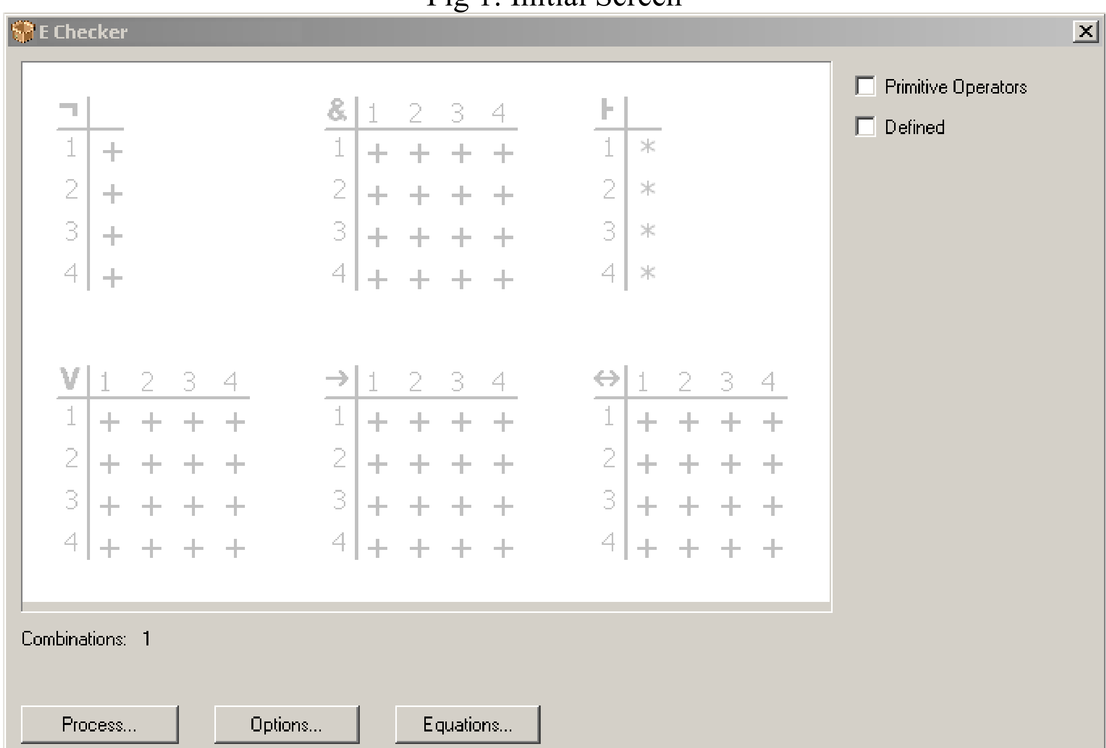
Following the approach set out in earlier papers[12] we have been concerned to test the degree to which classical logic may be weakened to permit paraconsistency but kept strong enough to allow as many forms of classically valid reasoning as possible to be preserved. The intended result is the creation of a system (or systems) of reasoning which most closely replicates the look and feel of classical logic while still being tolerant of inconsistency[13].

The result of our previous work has been the development of a number of interesting new systems such as LM4, Epsilon 442 and Epsilon 444 each of which appears to offer an optimal or nearly optimal weakening of classical logic. These systems have been tested against a suite of classically valid reasoning forms (given in appendix 1) and this same set of test formulae has been incorporated into E-Checker. This original set can be added to, or pruned as required to permit the best degree of freedom for users to investigate the consequences of choosing different matrices and designation schemes as the basis of their systems.

In the current version of E-Checker, Negation and Conjunction are treated as primitive operators in terms of which Disjunction, Implication and Bi-Implication may (or may not, depending on user choice) be defined.

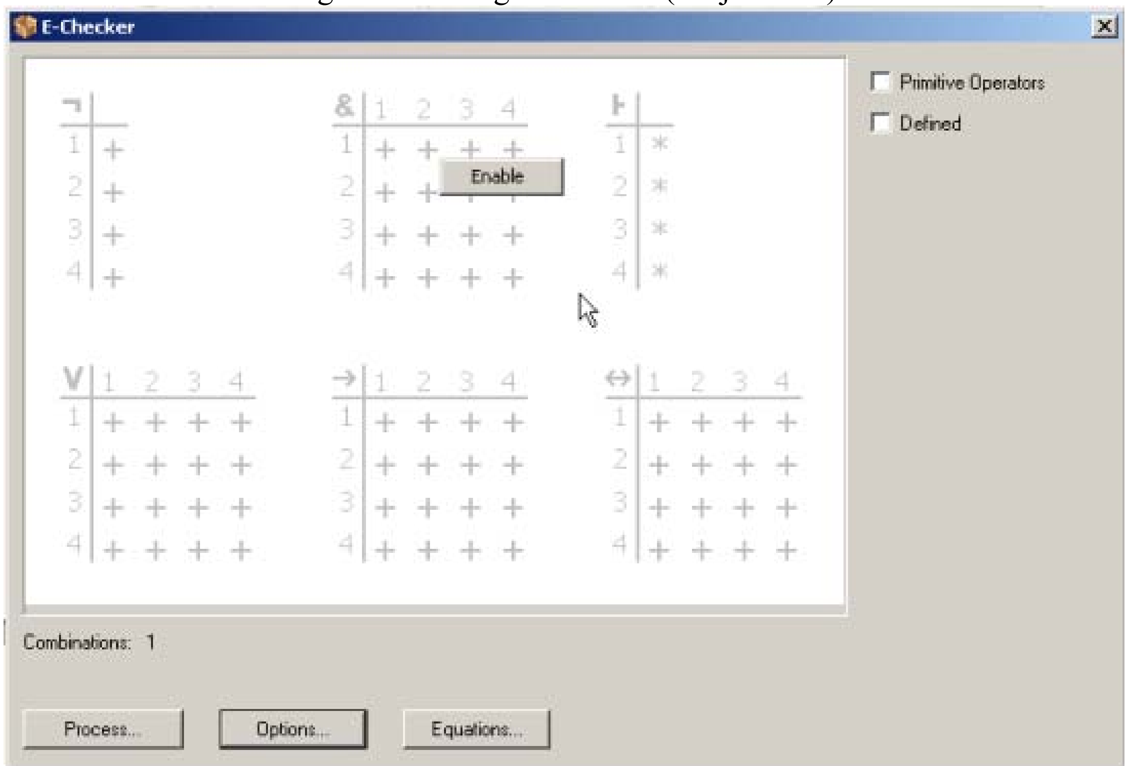
The initial appearance of E-Checker may be seen in Fig.1 below.

Fig 1: Initial Screen



Individual matrices are set up by moving the mouse over them and right clicking to enable them (Fig 2).

Fig 2: Enable a given matrix (conjunction)



From this point the individual cells of the matrix are available for users to either stipulate as taking one of the available values {1,2,3,4} or leave as wildcards {*}. In the latter case, the effect will be that when E-Checker is run it will try each of the available in turn and produce results for each.

Figs 3, 4 and 5, below show, respectively, an enabled matrix for conjunction for which no cells have yet had values stipulated , the value setting dialogue which is triggered by pressing the left mouse button over a given cell position, and the fully stipulated matrices for the Epsilon 442 system. Fig. 6 and 7 show the enabling and setting up of a designation scheme and Fig. 8 shows the completed Epsilon 442 configuration which is now ready to be processed. Fig 9 shows the process configuration screen. This offers the user the opportunity to set-up a number of paths to the test files and to specify where results will be saved. A number of options for result logging are offered in the current version including compact text output, verbose text output and HTML with linked tables.

Fig. 10 shows an example of output produced by E-Check in HTML linked form and Fig. 11 shows a sample linked table in HTML format (for Ex Contradictione Quodlibet).

Even in this relatively early stage of its development, E-Checker is providing a useful testing tool for 4 valued systems of logic. Its primary value is for checking matrices containing a number of wildcards. This has the potential to save a great deal of time when trying to uncover matrices which may be of interest for further development.

Fig 3: An enabled unset matrix (conjunction)

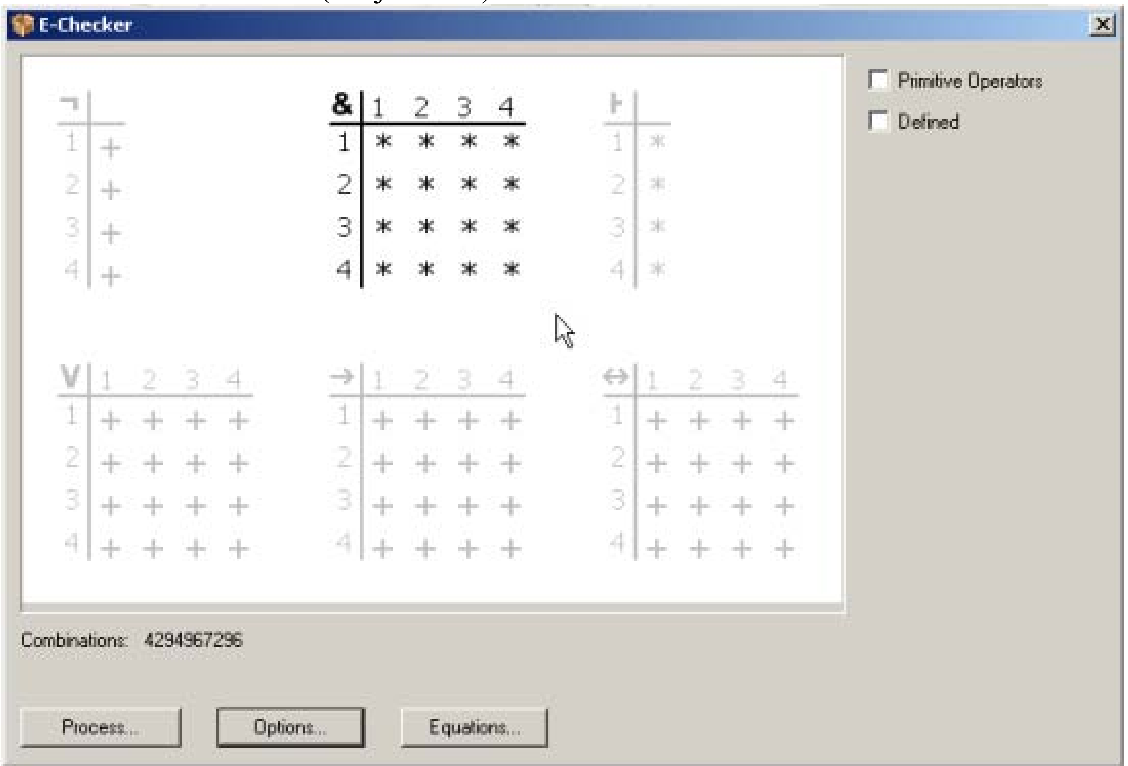


Fig 4: Setting a matrix cell (conjunction)

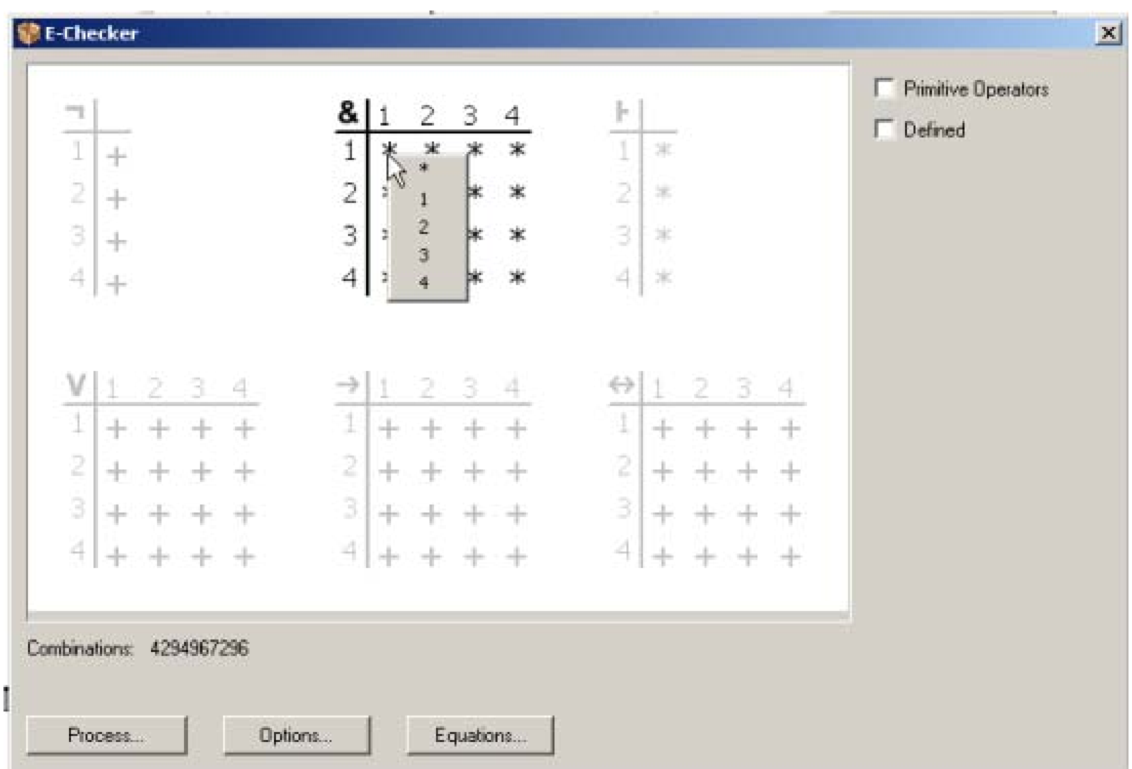


Fig 5: Conjunction & Negation set up for system Epsilon 442

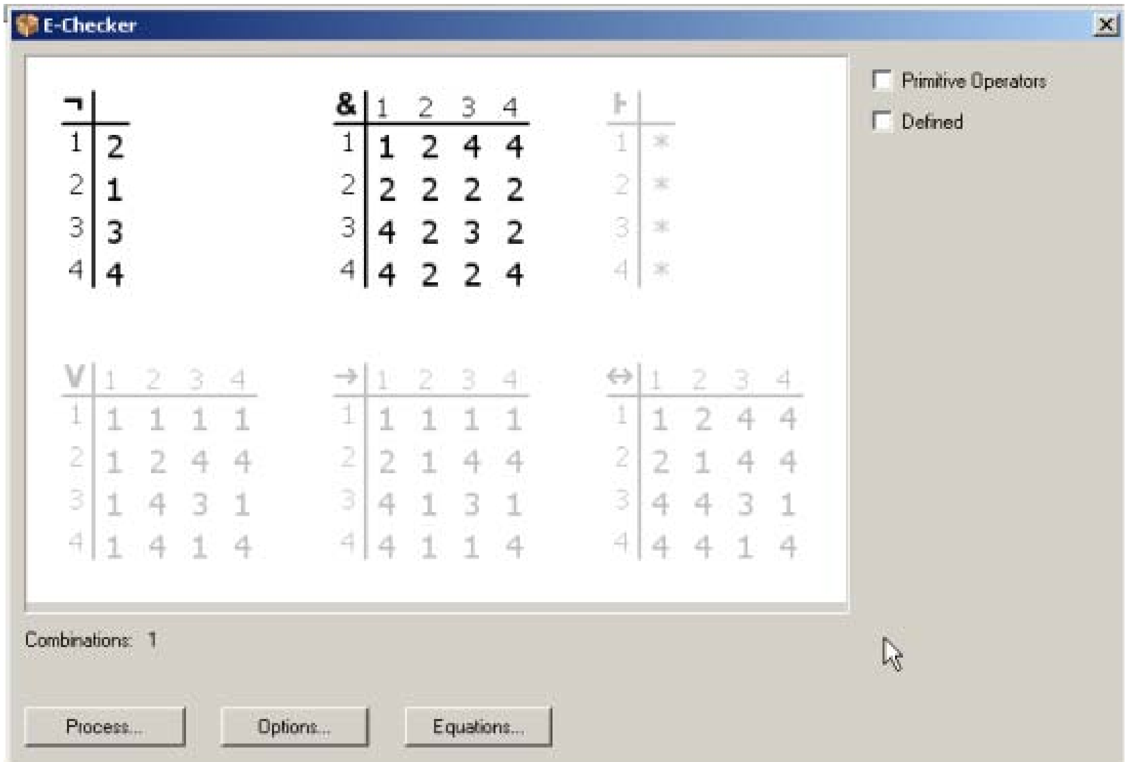


Fig 6: Setting up a designation scheme

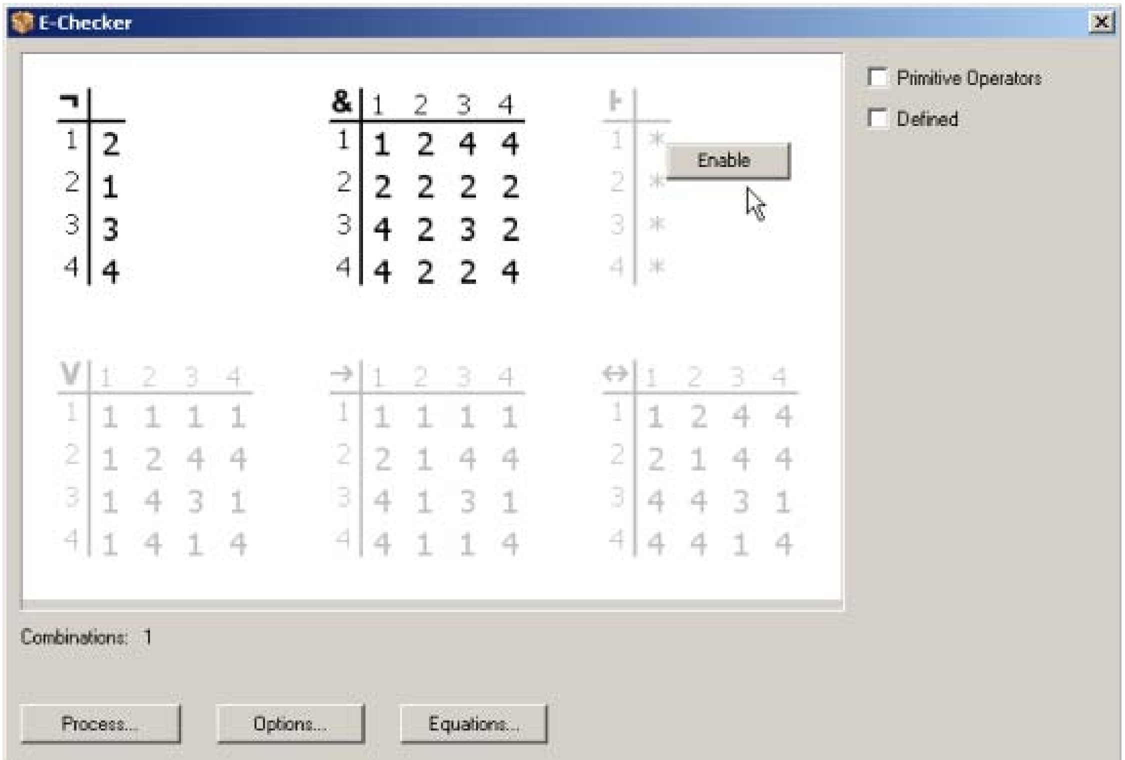


Fig 7: Setting up a designation scheme

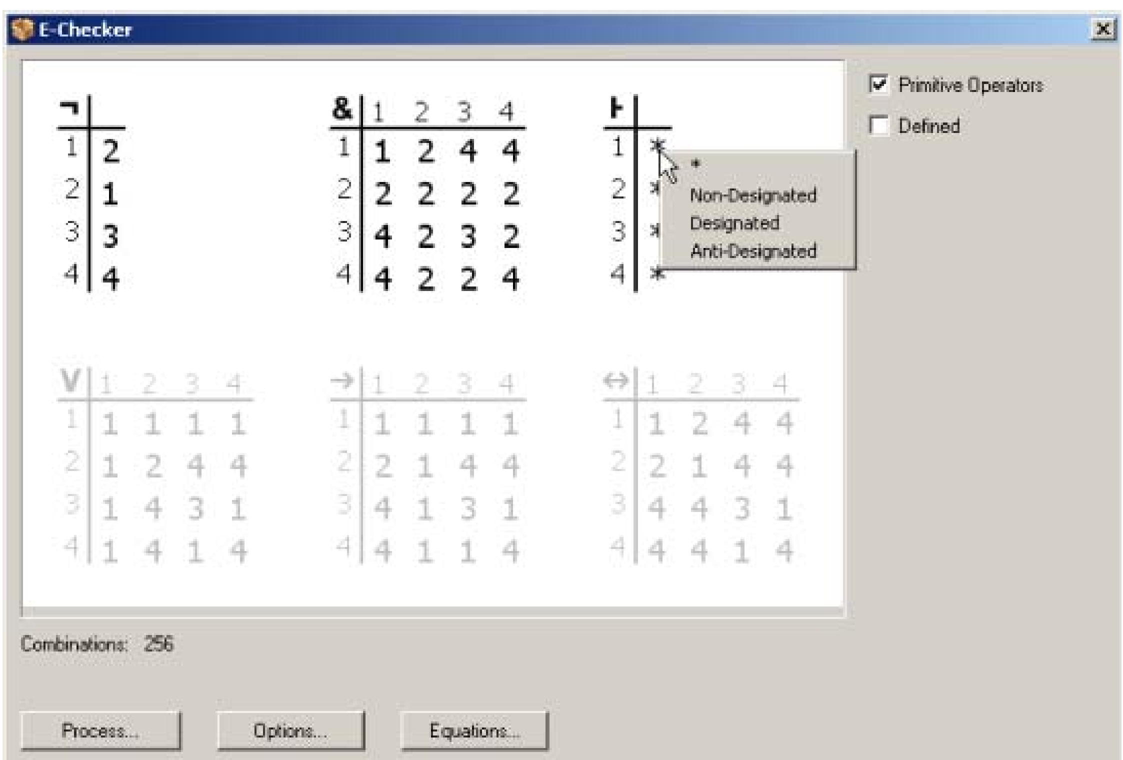


Fig 8: Epsilon 442 Set-Up

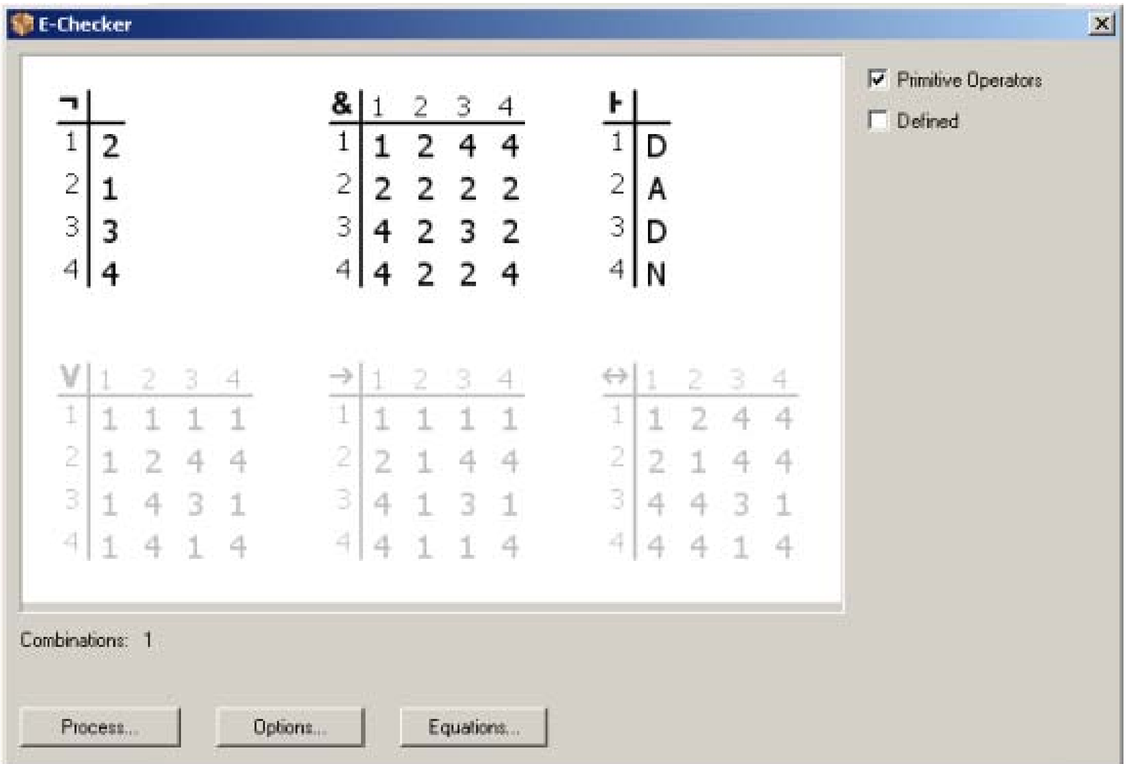


Fig 9: Process Screen

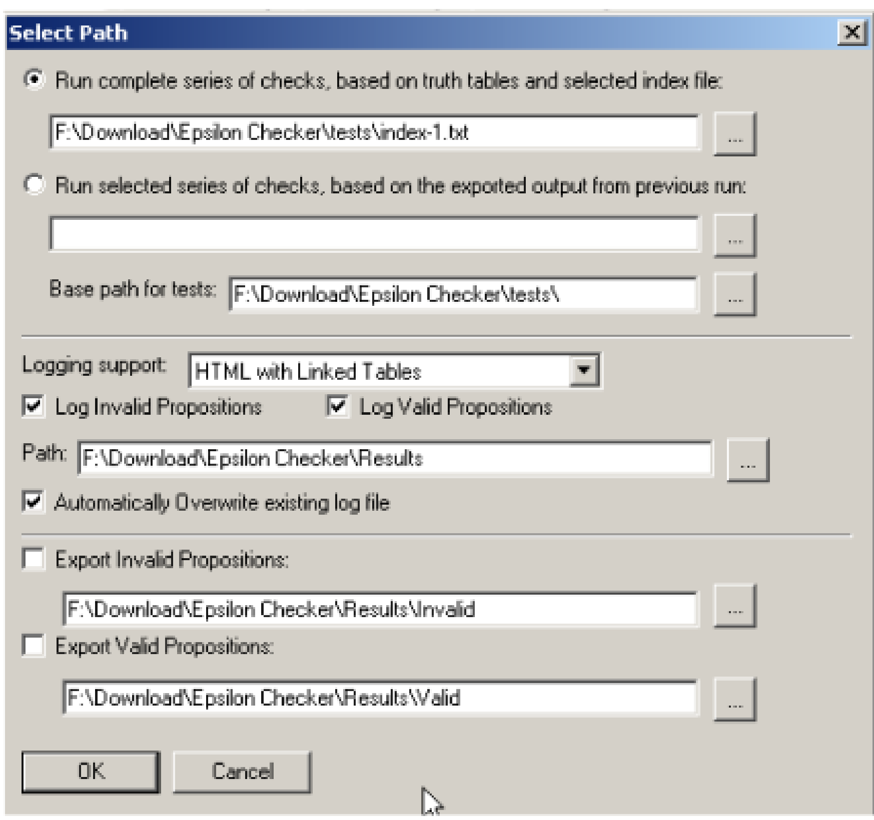


Fig 10: Sample HTML Log Screen

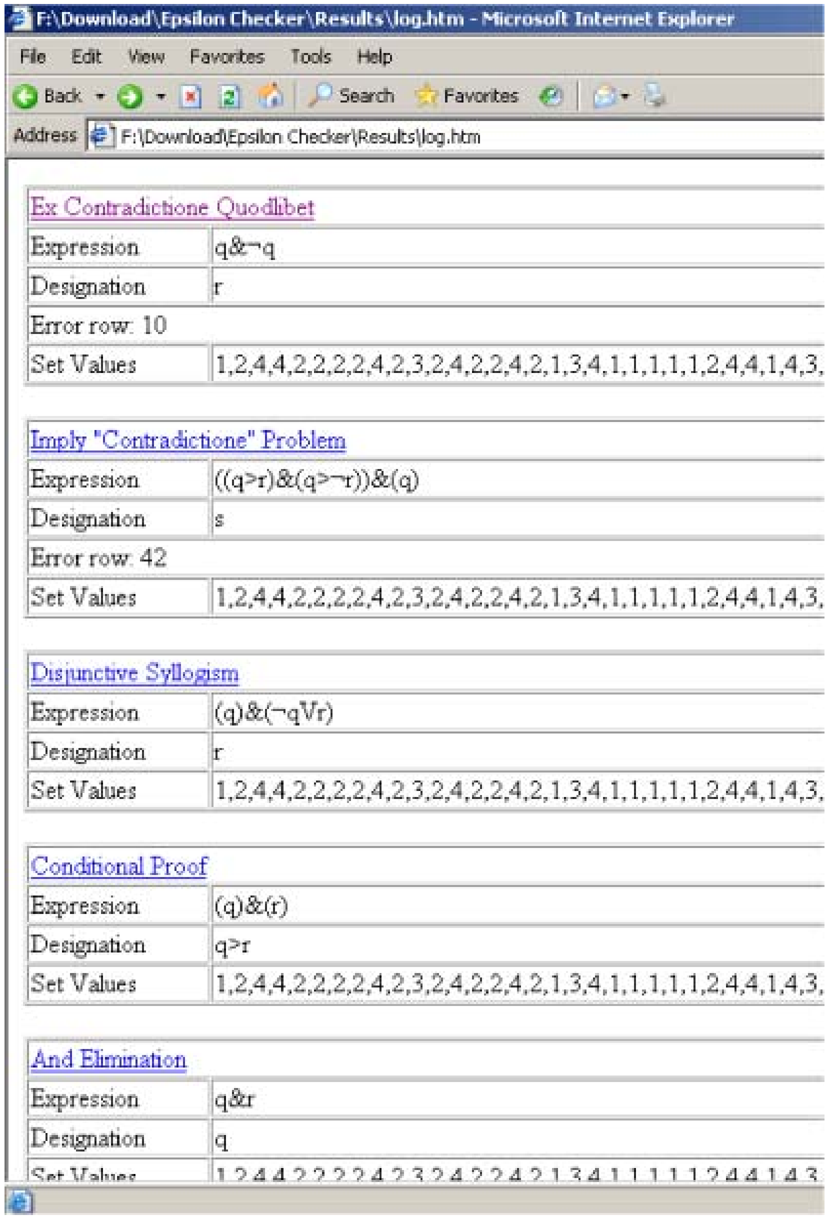
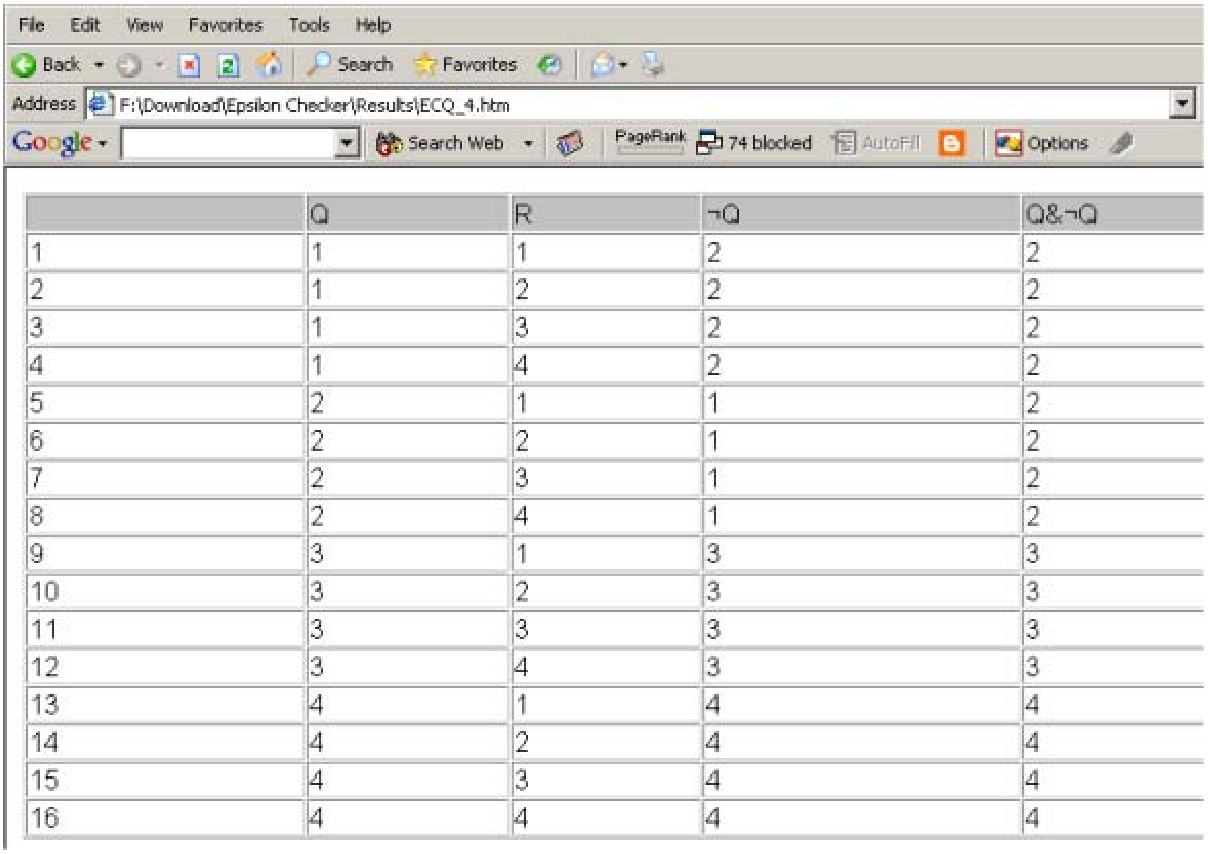


Fig 11: Sample HTML Detail Screen (Ex Contradictione Quodlibet)



	Q	R	¬Q	Q&¬Q
1	1	1	2	2
2	1	2	2	2
3	1	3	2	2
4	1	4	2	2
5	2	1	1	2
6	2	2	1	2
7	2	3	1	2
8	2	4	1	2
9	3	1	3	3
10	3	2	3	3
11	3	3	3	3
12	3	4	3	3
13	4	1	4	4
14	4	2	4	4
15	4	3	4	4
16	4	4	4	4

Appendix 1: Initial Test set.

Lemmon 002 ((¬r => (¬q => r)) & (¬r)) |- ¬q => r
Lemmon 004 (((q => (r => s)) & (q => r)) & (q)) |- s
Lemmon 006 (((q => (r => s)) & (q)) & (¬s)) |- ¬r
Lemmon 007 ((q => ¬r) & (r)) |- ¬q
Lemmon 008 ((¬q => r) & (¬r)) |- q
Lemmon 010 (q => (r => s)) |- r => (q => s)
Lemmon 011 (r => s) |- (¬r => ¬q) => (q => s)
Lemmon 013 ((q & r) => s) |- q => (r => s)
Lemmon 015 (q & r) |- r
Lemmon 018 (r => s) |- (q & r) => (q & s)
Lemmon 020 (r => s) |- (q V r) => (q V s)
Lemmon 021 (s V (q V r)) |- q V (s V r)
Lemmon 022 ((q => r) & (q => ¬r)) |- ¬q
Lemmon 023 (q => ¬q) |- ¬q
Lemmon 024 (q=r) |- r=q
Lemmon 025 ((q) & (q=r)) |- r
Lemmon 026 ((q=r) & (r=s)) |- q=s
Lemmon 027 ((q & r)=q) |- q => r
Lemmon 028 (q & (q=r)) |- q & r
Lemmon 029 (q) |- q
Lemmon 031a (q & (q V r)) |- q
Lemmon 032a (q V (q & r)) |- q
Lemmon 032b (q) |- q V (q & r)
Lemmon 033a (q V q) |- q
Lemmon 034a ((q) & (¬(q & r))) |- ¬r
Lemmon 034b (¬r) |- q & (¬(q & r))
Lemmon 035b (¬(q & ¬r)) |- q => r
Lemmon 036a (q V r) |- ¬(¬q & ¬r)
Lemmon 036b (¬(¬q & ¬r)) |- q V r
Lemmon 045 (q) |- (q & r) V (q & ¬r)
Lemmon 046 (q => r) |- q & (r=q)
Lemmon 047b (q => r) |- (q & r)=q
Lemmon 048 (¬q V r) |- q => r
Lemmon 050 (q) |- r => q
Lemmon 051 (¬q) |- q => r
Lemmon 053 ((¬r) & (q V r)) |- q
Lemmon Exercise 1(a) ((q => (q => r)) & (q)) |- r
Lemmon Exercise 1(b) (((r => (q => s)) & (¬s)) & (r)) |- ¬q
Lemmon Exercise 1(c) ((q => ¬¬r) & (q)) |- r
Lemmon Exercise 1(d) ((¬¬r => q) & (¬q)) |- ¬r
Lemmon Exercise 1(e) ((¬q => ¬r) & (r)) |- q
Lemmon Exercise 1(f) (q => ¬r) |- r => ¬q
Lemmon Exercise 1(g) (¬q => r) |- ¬r => q

Lemmon Exercise 1(h)	(¬q => ¬r) |- r => q
Lemmon Exercise 1(j)	(q => (r => s)) |- (q => r) => (q => s)
Lemmon Exercise 1(l)	(q => r) |- (r => s) => (q => s)
Lemmon Exercise 1(m)	(q) |- (q => r) => r
Lemmon Exercise 1(n)	(q) |- (¬(r => s) => ¬q) => (¬s => ¬r)
Lemmon Exercise 1 p.27-8 (a)	(q) |- r => (q & r)
Lemmon Exercise 1 p.27-8 (b)	(q & (r & s)) |- r & (q & s)
Lemmon Exercise 1 p.27-8 (c)	((q => r) & (q => s)) |- q => (s & r)
Lemmon Exercise 1 p.27-8 (d)	(r) |- q V r
Lemmon Exercise 1 p.27-8 (e)	(q & r) |- q V r
Lemmon Exercise 1 p.27-8 (f)	((q => s) & (r => s)) |- (q V r) => s
Lemmon Exercise 1 p.27-8 (j)	(¬q => q) |- q
Lemmon Exercise 1 p.33 (a)	((r) & (q=r)) |- q
Lemmon Exercise 1 p.33 (b)	((q => r) & (r => q)) |- q=r
Lemmon Exercise 1 p.33 (c)	(q=r) |- ¬q=¬r
Lemmon Exercise 1 p.33 (d)	(¬q=¬r) |- q=r
Lemmon Exercise 1 p.33 (e)	((q V r)=q) |- q => r
Lemmon Exercise 1 p.33 (f)	((q=¬r) & (r=¬s)) |- q=s
Lemmon Exercise 1 p.41 (a)	(q V r) |- q V r
Lemmon Exercise 1 p.41 (b-1)	(q & q) |- q
Lemmon Exercise 1 p.41 (c-2)	((q & r) V (q & s)) |- q & (r V s)
Lemmon Exercise 1 p.41 (d-2)	((q V r) & (q V s)) |- q V (r & s)
Lemmon Exercise 1 p.41 (e-1)	(q & r) |- ¬(q => ¬r)
Lemmon Exercise 1 p.41 (e-2)	(¬(q => ¬r)) |- q & r
Lemmon Exercise 1 p.41 (f-1)	(¬(q V r)) |- ¬q V ¬r
Lemmon Exercise 1 p.41 (f-2)	(¬q V ¬r) |- ¬(q V r)
Lemmon Exercise 1 p.41 (g-1)	(¬q & ¬r) |- ¬(q V r)
Lemmon Exercise 1 p.41 (h-1)	(q & r) |- ¬(¬q V ¬r)
Lemmon Exercise 1 p.41 (h-2)	(¬(¬q V ¬r)) |- q & r
Lemmon Exercise 1 p.41 (j-1)	(¬q => r) |- q V r
Lemmon Exercise 1 p.41 (j-2)	(q V r) |- ¬q => r
Lemmon Exercise 2 p.63 (a)	(((q => r) => q) & (q => r)) |- q

Appendix 2: Epsilon 442 & 444 Matrices

Epsilon 442

NOT	
1	2
2	1
3	3
4	4

AND	1	2	3	4
1	1	2	4	4
2	2	2	2	2
3	4	2	3	2
4	4	2	2	4

¬(¬P & ¬Q)

OR	1	2	3	4
1	1	1	1	1
2	1	2	4	4
3	1	4	3	1
4	1	2	1	4

¬(P & ¬Q)

COND	1	2	3	4
1	1	2	4	4
2	1	1	1	1
3	1	4	3	1
4	1	2	1	4

Epsilon 444

NOT	
1	2
2	1
3	3
4	4

AND	1	2	3	4
1	1	2	4	4
2	2	2	2	2
3	4	2	3	4
4	4	2	4	4

¬(¬P & ¬Q)

OR	1	2	3	4
1	1	1	1	1
2	1	2	4	4
3	1	4	3	4
4	1	2	4	4

¬(P & ¬Q)

COND	1	2	3	4
1	1	2	4	4
2	1	1	1	1
3	1	4	3	4
4	1	2	4	4

Footnotes:

[1] For example, Priest G., & Tanaka K., in the Stanford Encyclopaedia of Philosophy, http://plato.stanford.edu/entries/logic-paraconsistent/

[2] In fact |- may behave explosively even in systems which do not permit (q & ¬q) |- r. For an example see the LM_4 system of Anderson C.D.P., in "A solution to the problem of contradiction in knowledge discovery applications." Proceedings AISB 2002. A similar problem occurs with the C_1 system of da Costa., where A & ¬A |- B is invalid but a substitution instance of the same expression (A & ¬A) & ¬(A & ¬A) |- B is valid.

[3] For example see Quine W.V.O., 'On what there is', reprinted in From a logical Point of View, Cambridge, Mass. Harvard, 1980 p.18

[4] Asenjo, F.G. "A Calculus of Antinomies", *Notre Dame Journal of Formal Logic*, Vol. XVI, pp. 103-5, 1966

[5] da Costa. N.C.A., "On the theory of inconsistent formal systems", Notre Dame Journal of Formal Logic, Vol XV, No.4 October 1974 pp.497-510

[6] Dunn, J.M. "Intuitive Semantics for First Degree Entailment and Coupled Trees", *Philosophical Studies*, Vol. XIX, pp. 149-68, 1976

[7] Belnap, N.D., "A Useful Four-valued Logic", Modern Use of Multiple-valued Logic, J.M. Dunn and G. Epstein (eds.), D.Reidel Publishing Company, Dordrecht, 1977.

[8] Priest, G. "Logic of Paradox", *Journal of Philosophical Logic*, Vol. VIII, pp. 219-241, 1979

[9] See da Costa, op cit where the C_1 system does not employ the usual equivalences and furthermore employs a non-truth functional form of negation one consequence of which is the invalidity of A |- ¬¬A.

[10] First suggested by Asenjo, F.G., op.cit.

[11] Clearly this decision will have a profound effect on the logical characteristics of a system. For example Kleene's three valued system, becomes non-explosive if both T and I are treated as designated and F as anti-designated.

[12] For example, Anderson, C.D.P., "A solution to the problem of contradiction in knowledge discovery applications.", 9^{th} Workshop on Automated Reasoning, Imperial College London, 2002

[13] See Anderson, C.D.P., "Developing a framework for investigating inconsistency handling in automated reasoning.", 6th World Multi-Conference on Systemics, Cybernetics and Informatics, Florida, USA, 2002