2.a. *Authors:* Colin James III; Garry Goodwin
2.b. *Title:* **Meth8 model prover for multivalued logic: Truth is as a white light.**
2.c. *Keywords:* Łukasiewicz variant, Meth8, modal theorem prover, multi-valued logic, VŁ4
2.d. *Abstract:* [words 131]

In this paper we look at two methods for modelling formal languages. We first look at a bivalent framework used to weaken a class of many valued logics with twin functors. We then introduce the idea of primary values. Primary values are the maximal number of contrary formulae expressible in the language. The set of primary values is equally as important as the set of axioms. In the spirit of Suszko's Thesis the set is evaluated as a two valued logic. As an example, we provide the primary set for S5's binary fragment. This approach informs an automated theorem prover / model checker presently under development named Meth8. In the second part of this paper, we show how Meth8 implements elements of the semantic framework as elucidated in the first part.

2.e. *Affiliations, respectively:* Ersatz Systems Machine Cognition, LLC; Semantic-Qube UK
2.f. 1440 Farnham Pt #203, Colorado Springs, CO 80904-5220, USA
2.g. info@cec-services.com, USA +011.719.210.9534

Size of the entire document herewith including this separate title page is:
24,437 characters out of 27,000; or about 15 pages of continuous text.

Table 15 should appear on its own page.

Title: Meth8 model prover for multivalued logic: Truth is as a white light.

INTRODUCTION

We characterize a formal language as a many valued logic with the generic structure:

General Structure: [T^{Π}, T^{V}: $\{V^{\top}, V^{n}, V^{\perp}\}$], ⊨, ~, &, v, →, ⥱, ↔, v̲, △, +△, −△]

T^{Π} is the set of truth possibilities. T^{V} is the set of truth values. $V^{\top}$ is the set of designated values, $V^{\perp}$ is the set of falsifying values, and V^{n} the set of non designating values not false. Tab. 1 and Tab. 2 offer two versions of validity.

Tab. 1

$_1$⊨	$V^{\top}$	V^{n}	$V^{\perp}$
$V^{\top}$			✗
V^{n}			
$V^{\perp}$			

Tab. 2

$_2$⊨	$V^{\top}$	V^{n}	$V^{\perp}$
$V^{\top}$			✗
V^{n}			✗
$V^{\perp}$			

Tab. 1 is the minimum threshold for validity. Tab. 1 in words:

Γ $_1$⊨ A   iff there are no models such that all values of Γ are true and A is false. (1.0)

Tab. 1 may prove insecure for many valued logic and is strengthened as Tab. 2. In words:

Γ $_2$⊨ A   iff there are no models such that all values of Γ are non falsifying and A is false. (1.1)

Other elements of the general structure are defined on Tab. 3. The triangular notation △ marks the presence of a functor.

BIVALENT FRAMEWORK

The bivalent framework evaluates the two truth possibilities *p is case* and *p is not the case*. Two valued logic is as Tab. 3.

Tab. 3

	p	~p	⊤	⊥
p is the case	T	F	T	F
p is not the case	F	T	T	F

On Tab. 3 the truth conditions are tautological. For example, to assert ‘p’ means *p is the case* is true when *p is not the case* is false.

The bivalent framework was originally designed to weaken the four valued modal logic of Łukasiewicz. [Łukasiewicz, J. (1953), 1, 111-149; Łukasiewicz, J. (1957), VII] Ł$_4$ is a B$_4$ algebra with twin modal functors as Tab. 4.

Tab. 4

		△		▽	
	~	□	◇	□	◇
1	0	2	1	3	1
2	3	2	1	0	2
3	2	0	3	3	1
0	1	0	3	0	2

Despite Ł$_4$'s conservatism it has multiple complaints. 2.0 is a noted egregious Ł$_4$ theorem.

$$\vDash_{Ł4} (\Diamond p \;\&\; \Diamond q) \rightarrow \Diamond(p \;\&\; q) \qquad (2.0)$$

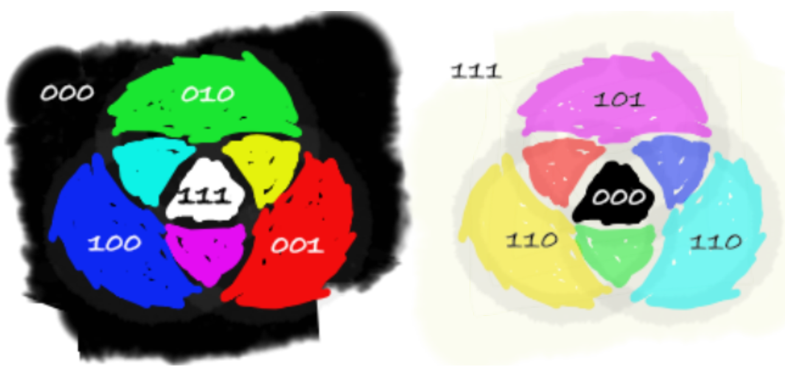
Béziau points out 2.0 proved a nightmare for Łukasiewicz. [Béziau, J-Y. (2011), 54] Consider the counter: *If it is possible the President is in Washington and possible the President is in London, then, it is possible the President is both in Washington and London*. It is clear Ł$_4$ is untenable as an alethic logic but we wonder how Ł$_4$ may be rehabilitated.

Tab. 5 introduces the Łukasiewicz △ functor to the bivalent framework.

Tab. 5

	p	~p	□p	~□p	◇p	~p
p is the case	1	0	2	3	1	0
p is not the case	0	1	0	1	3	2

For Tab. 5 if 1 is interpreted as true and 0 is false, this begs the question as how to interpret 2 and 3. For an answer we refer to basic RGB color theory in Fig 1.



Additive Subtractive

Fig. 1

Basic color theory is an eight valued B_8 algebra. In the additive model the presence of a primary color is a denial of the minimal value black. In the subtractive model primary colors are contrary properties. For both models a primary color is a property of white light. The lesson is generalised:

> A primary value is a denial of the minimal zero, contrary to other primary values, and a property of the maximal value. (3.0)

Following 3.0, if the designated value of $Ł_4$ is interpreted as true, then the middle values 2 and 3 are properties of true and deny false. A class of contingent adjectives provides a solution e.g. {accidental, incidental, coincidental, marginal, temporary, extraneous, superfluous, etc.}. This class preserves truth. For example, if a state of affairs is accidental it is contingent yet also true. When the class is joined we name it C.

> C $=_{def}$ accidental or incidental or coincidental or marginal or temporary or superfluous, … etc. (4.0)

We name the series of negative conjunction N for non-contingent.

> N $=_{def}$ not accidental and not incidental and not coincidental and not marginal and not temporary and not superfluous, … etc. (4.1)

There is a possible world counterpart to the natural language definitions. W_1 is the start world and W_2 some world accessible from W_1.

C	N	
True in W_1 & False in W_2	True in W_1 & True in W_2	(5.0)

The set of values are false, contingent, non contingent, and true, viz., {F, C, N, T}. The basic non-modal and alethic propositions are defined as Tab. 6.

Tab. 6

	p	~p	□p	~□p	◇p	~◇p	Np	Cp
p is the case	T	F	N	C	T	F	N	C
p is not the case	F	T	F	T	C	N	N	C

If we replace {0, 3, 2, 1} with the B_4 set {00, 10, 01, 11} there is an intuition that says extremes of necessity ought to be held farthest apart, i.e. (00 01)(p) = □p and (10 00)(p) = ~◇p. We name this polarity. Polarity occurs if the ▽ functor applies to the positive case and the △ functor to the negative case as Tab. 7.

Tab. 7

	p	~p	□p	~□p	◇p	~◇p	
p is the case	11	00	01	10	11	00	+▽
p is not the case	00	11	00	11	01	10	−△

The set {F, C, N, T} proves an inconsistent interpretation of a polar system i.e. both 01 and 10 are interpreted as N. We introduce the alternative values {(U) unevaluated, (I) improper, (P) proper, (E) evaluated}. The values I and P are a conditional access between worlds.

$$\overset{\text{I}}{\text{True in } W_1 \rightarrow \text{False in } W_2} \qquad \overset{\text{P}}{\text{True in } W_1 \rightarrow \text{True in } W_2} \tag{6.0}$$

As a combined system {F, C, N, T} is Model 1 and {U, I, P, E} is Model 2. Tab. 8 makes clear how the B_4 set is interpreted in either model.

Tab. 8

11	1	T	E
01	2	N	P
10	3	C	I
00	4	F	U

Tab. 9 extends the interpretations of the non modal and propositions to both models.

Tab. 9

	p	~p	□p	~□p	◇p	~◇p
p is the case	T, E	F, U	N, P	C, I	T, E	F, U
p is not the case	F, U	T, E	F, U	T, E	C, P	N, I

In Model 2 the modal box is interpreted as correct and the lozenge as passable. Correct may mean unmistaken or appropriate.

A theorem in this *two-tone* variant of $Ł_4$ ($VŁ_4$) is valid in both models. Model 1 is equivalent to $Ł_4$ and harbors no further caveats. Model 2 qualifies Model 1, and so $VŁ_4$ theorems are a subset of $Ł_4$. Model 2 has additional technical framework because it is not clear which is the correct functor to apply when the number of propositions is greater than one. At such times middle rows of a table mix truth possibilities. Tab. 10 covers all of the available options.

Tab. 10. Three modal options for mixed truth possibilities

□1	□2	□3	◇1	◇2	◇3
×E	×U	×P , ×I	+U	+E	+I , +P

On Tab. 10 option 1 is neutral, leaving the middle rows of a truth table unchanged. The box operator under option 2 returns U, the lozenge returns E. Option 3 evaluates twins functors separately. Given options 1 and 2, option 3 is redundant. Atomic formulae are unary and do not have a middle row. Hence the question of which option does not arise.

Along with many implausible theorems, Model 2 invalidates 2.0 as seen on Tab. 11.

Tab. 11

(◇A	&	◇B)	→	◇	(A	&	B)
◇p		◇q		Option1	p		q
PEPE	PEPE	EEEE	IIEE	UIPE	UIPE	UIPE	EEEE
PEPE	PEPE	EEEE	IUEP	UUPP	UIPE	UUPP	PPPP
PEPE	PPPP	PPPP	IIII	UIUI	UIPE	UIUI	IIII
PEPE	PPPP	PPPP	IIII	UUUU	UIPE	UUUU	UUUU

The one instance on Tab. 11 where E → U means the inference is not a valid consequence in V$Ł_4$ (see Tab. 1). More controversial is Model 2 which finds against axiom K.

Tab. 12

□	(A	→	B)	→	(□A	→	□B)
Option1					Option1		□q
EEEE	UIPE	EEEE	EEEE	EPEP	UIPE	EPEP	PPPP
EPEP	UIPE	EPEP	PPPP	EEEE	UIPE	EPEP	PPPP
EEII	UIPE	EEII	IIII	EPEP	UIPE	EPIU	UUUU
EPIU	UIPE	EPIU	UUUU	EEEE	UIPE	EPIU	UUUU

As K is not controversial we should not expect conspicuous counter examples. However, on Tab. 12 the condition I → U is a cause for concern, viz., □(A → B) $_2\nvDash$ (□A → □B).

Consider the following example first reading the modal box as 'correct': *If correct that banking regulations imply egregious losses mount up, then, correct banking regulations imply it is correct egregious losses mount up*. It may be correct the present state of regulation leads to egregious losses, but this does not mean correct regulation implies egregious losses.

Another example reads the modal box as necessity: *If it is necessarily the case freewill implies sometimes a person abstains, then freewill is necessarily the case implies sometimes a person abstains is necessarily the case*. If a person who never abstains entails the negation of freewill, then on that condition the antecedent is true. However, if the final consequent means sometimes abstinence is the only option then freewill is negated.

If we look again at Tab. 12 the inference fails where the consequent is unevaluated. The first example invokes regulation both 'correct and egregious' and the second example invokes an abstinence both 'necessary and optional'. Both examples invite oxymora that make little sense and hence the validity of K as a structural inference is threatened.
The bivalent framework is not limited to $Ł_4$. One well known set of four valued matrices is Lewis and Langford's Groups I-V. [Lewis, C. I., Langford, H. C. (1959), 493-494] On Tab. 13 we include the necessity operator.

Tab. 13

	I		II		III		IV		V	
	□	◇	□	◇	□	◇	□	◇	□	◇
1	2	1	1	1	1	1	1	2	2	1
2	4	1	4	2	4	1	3	2	4	2
3	4	1	3	1	4	1	3	2	3	1
4	4	3	4	4	4	4	3	4	4	3

Group III lacks a twin and cannot be weakened. The other groups do have twins if the second designated value is switched when the negative case. This is problematic in as far as it is uncertain whether it is 2 or 3 that is designated where truth possibilities are mixed. With that caveat, groups I, II, IV and V may be weakened using the bivalent framework. For axiom K groups I, II and V have a set of conditions such that 2 → 4 or 3 → 4. For group IV there is the set of conditions 2 → 3 or 3 → 2. Despite uncertain designation these conditions ensure the inference is invalid. Whilst Model 2 militates against Lewis' strict implication it is worth noting I and II preserve his amended postulates A1-A7 after weakening, but A8 is now invalid. However, Group V also originally failed to validate A8. [(1959), 493-495].

GENERAL STRATEGY FOR PARSING MINIMAL SETS

The objective is to take any logic with Boolean operations (&) and (~) and parse the minimal set of semantic elements. The minimal set is an intuitively easy concept to grasp. In color theory it is the set of primary colors, viz., {red, green, blue}. The set contains no subcontrary pair of elements, and no individual formula is a contradiction. In a formal language the minimal set has the maximal number of contrary elements expressible in the language. Suszko's Thesis is taken to mean "every logic is logically two valued". [Suszko, R. (1977), 378] The objective here is to give a zero-one evaluation of the minimal set.

We look at modal system S5. The S5 unary fragment is a simple B_4 algebra with four primary values, viz., (0001)(□p), (0010)(~□p &p), (0100)(◇p & ~p), (1000)(~◇p).

As the unary fragment is a B_4 logic the starting point for binary formula is a 4 × 4 grid. Extended analysis proves a simple 4 × 4 grid is insufficient and the final grid is as Tab. 14.

Tab. 14

1	2					3					4
5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28
29	30					31					32

We account for all 32 primary values in Tab. 15. The number corresponds to their location on the grid. These formula whilst syntactically complex are the S5 semantic atoms (primary *colors*).

Tab. 15. The minimal set for S5 has 32 primary values.

1. □p & □~q
2. □p & ◇q & ~q
3. □p & q & ◇~q
4. □p & □q
5. p & ◇~p & □~q
6. □(q → p) & p & ◇~p & ◇q & ~q
7. (□(p v q) v □(~p v ~q)) & ◇(p & q) & p & ◇~p & ◇q & ~q
8. (□(p v q) v □(~p v ~q)) & ◇(~p & ~q) & p & ◇~p & ◇q & ~q
9. □(p v q) & □(~p v ~q) & p & ◇~p & ◇q & ~q
10. (◇ (~p & q) ↔ ◇ (p & q)) & ◇ (~p & ~q) & p & ◇~p & ◇q & ~q
11. □(p v q) & p & ◇~p & ◇~q & q
12. (□(p v ~q) v □(~p v q)) & ◇ (~p & q) & p & ◇~p & ◇~q & q
13. (□(p v ~q) v □(~p v q)) & ◇(p & ~q) & p & ◇~p & ◇~q & q
14. □(p v ~q) & □(~p v q) & p & ◇~p & ◇~q & q
15. (◇(~p & ~q) ↔ ◇ (p & ~q)) & ◇(~p & q) & p & ◇~p & ◇~q & q
16. □q & p & ◇~p
17. ◇p & □~q & ~p
18. □(~p v ~q) & ~p & ◇p & ◇q & ~q
19. (□(p v ~q) v □(~p v q)) & ◇ (~p & q) & ~p & ◇p & ◇q & ~q
20. (□(p v ~q) v □(~p v q)) & ◇(p & ~q) & ~p & ◇p & ◇q & ~q
21. □(p v ~q) & □(~p v q) & ~p & ◇p & ◇q & ~q
22. (◇(p & q) ↔ ◇(p & ~q)) & ◇(~p & q) & ~p & ◇p & ◇q & ~q
23. □(p → q) & ~p & ◇p & ◇q & q
24. (□(p v q) v □(~p v ~q)) & ◇(p & q) & ~p & ◇p & ◇~q & q
25. (□(p v q) v □(~p v ~q)) & ◇ (~p & ~q) & ~p & ◇p & ◇~q & q
26. □(p v q) & □(~p v ~q) & ~p & ◇p & ◇~q & q
27. (◇(p & ~q) ↔ ◇(p & q)) & ◇(~p & ~q) & ~p & ◇p & ◇~q & q
28. □q & ◇p & ~p
29. □~p & □~q
30. □~p & ~q & ◇q
31. □~p & q & ◇~q
32. □~p & □q

The 32 primary values form a grid. However, the extra five formulae of the four central cells on Tab. 15 require additional 4 × 4 grids that qualify the formula as Tab. 16.

Tab. 16

6	7	8	9	10
1100	0010	0000	0001	0000
1100	0011	0000	0000	0000
0000	0000	1100	0000	0011
0001	0000	0100	1000	0010

a

11	12	13	14	15
0011	0100	0000	1000	0000
0011	1100	0000	0000	0000
0000	0000	0011	0000	1100
1000	0000	0010	0001	0100

b

18	19	20	21	22
0001	0100	0000	1000	0010
0000	1100	0000	0000	0011
1100	0000	0011	0000	0000
1100	0000	0010	0001	0000

c

23	24	25	26	27
1000	0000	0010	0001	0100
0000	0000	0011	0000	1100
0011	1100	0000	0000	0000
0011	0100	0000	1000	0000

d

Tab. 17 is a limited selection of binary grids sufficient for modeling S5's binary fragment. Formulae in which the scope of the modal operators extends to two variables incorporate grids a, b, c, d from Tab. 16, or their negations.

Tab. 17

1	2	3	4	5	6	7
□p	p	◇p	□q	q	◇q	□(p v q)
1111	1111	1111	0001	0011	0111	1111
0000	1111	1111	0001	0011	0111	0bb1
0000	0000	1111	0001	0011	0111	00b1
0000	0000	0000	0001	0011	0111	0001

8	9	10	11	12	13	14
□(q → p)	□(p → ~q)	□(p → q)	◇(~p &~q)	◇(q & ~p)	◇(p & q)	◇(p & ~q)
1111	1000	0001	0000	0000	0111	1110
1aa0	1c00	00d1	1bb0	0aa1	0c11	11d0
1a00	1cc0	0dd1	11b0	0a11	0cc1	1dd0
1000	1111	1111	1110	0111	0000	0000

The system of truth functional grids expands with each new variable considered. Hence the values are enumerable but potentially infinite. This point means a truth functional S5 complies with the result of Dugundji that establishes S1-S5 to have no finite matrix. [Dugundji, J. (1940), 5 (4), 150-151]

S5 is a normal modal logic with axiom K, but we have given reason to doubt K. A similar logic to S5 retains the basic grids 1-6 but additionally qualifies grids 7-14 in Tab. 18.

Tab. 18

*10**
□(p → q)
00d1
00d1
ddd1
1111

Grid 10* belongs to a system that invalidates K.

METH8 MODEL PROVER

Meth8 stands for *Me*chanical *th*eorem prover in *8*-bits. [James, 2015a, 50-51] It is a model prover for modal logic using the rules of VŁ4 in the sections above. The prover is driven by look up tables (lut) with calculation for intermediate results. The purpose of Meth8 is to invalidate models of logic systems.

The development language used is TrueBASIC®, an ANSI standard for educators. The source code is directly portable for embedded systems into VHDL (a subset of Ada 95) as for example in [James, 2015b, Appendix].

Programming constraints on large memory limit the number of literal variables to 24 propositions or 12 theorems. The propositions are named as the 24 lower case letters from a to z, but excluding the lower case letter of "l", as in lion, and lower case letter "o" as in ocean because they are easily confused with the ordinal digits of one and zero. The theorems are named as the 12 upper case letters from A to L. The operators supported are the modal box and lozenge, and negation here given in one character symbols as {#, %, ~}. The eight connectives supported are conjunction, disjunction, joint denial, converse implication, biconditional, implication, exclusive disjunction, and alternative denial in one character symbols as {&+-<=>@\}. The maximum number of characters in an input expression is 2^30 (1 B).

The model prover consists of three parts for parser, processor, prover as named with the acronym of p-cubed or P^3.

PARSER

The parser component requests input from the user for the logic system and parameter directives unique to that logic system and is stored in a file at the root directory. The parser requests input of an expression to be processed. It is checked for syntax compliance and semantic content. The syntax includes correct symbols within the allowed character sets for literal types, literal operators, and connectives. The semantic content includes: the order of operators, literals, and connectives; and the nesting of parentheses for argument. Sequential combinations of modal operators and negation to literals are automatically reduced to the minimal algebraic state. A novel approach to mapping matched parentheses uses a sliding window parser named SWP. Fig. 2 is a worked example.

			01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	
1.			□	(	□	(	◊	A	→	B	)	↔	□	(	A	→	□	B	)	)	S5:58.
				L		L					R			L					R	R	
				02		04					09			12					17	18	
2.	<<	01	02		04					09			12					17	18		<<
			L		L					R			L					R	R		
3.				**L**		*L*					R			L					R	R	
				02		*04*					09			12					17	18	
	<<	01	02		04					*09*			12					17	**18**		<<
			L		L					*R*			L					R	**R**		
4.	L			**02**		*04*								12							
	R					*09*								17					**18**		
5.	L			**02**		*04*								12							
	R			**18**		*09*								17							

Fig. 2

From Fig. 2, five steps match the pairs of nested parentheses, each of which is an argument:

1. Map all parentheses as L or R for left or right; there are three valid pairs.

2. Slide the window left, so that L in step 1 moves from character position 2 to position 1.

3. From the combined maps of steps 1 and 2 as on the top and bottom, tag the first adjacent L/R pair as [*04*, *09*]; then tag the next adjacent L/R pair as [12,17].

4. Write these tagged pairs to a first in, first out (FIFO) stack list as: [*04*, *09*], [12, 17].

5. Match remaining parentheses [**02**, **18**] and write to the stack: [*04*, *09*], [12, 17], [**02**, **18**].

Each argument within the expression is stored in a parse tree, with index keyed to the stack.

The parser is not relaxed but strict, as it makes no effort to second guess the input of the user. Explicit input assures correct parsing by using parentheses for order of precedence of arguments. For example, the formula

B & A + A & ~(A & ~B) = A & B + A & A + A & ~B; A & B = A (7.0)

reduces to a result of A & B = A, which probably is not the intended result. However, rewriting (7.0) using parentheses as the formula

(B & A) + (A & ~((A & ~B)) = (A & B) + (A & A) + (A & ~B); A = A (8.0)

assures an intended result of A = A. Meth8 rejects (7.0) as ambiguous and not a well formed function (wff), but accepts (8.0) as a wff.

PROCESSOR

A lut is based on three sources of data to populate it: 1. External files; 2. Data statements; and 3. Algorithmic calculation. Data read from external files is best suited in a small memory footprint of lut such as implementation in programmable hardware parts for speed. Software programs use self-contained data statements to build a lut in a larger memory space such as for desktop computing. Building a lut by calculation as needed on the fly is for hand held and portable devices such as tablets and cellphones. The software program relies on internal calculation and data statements to build a lut.

Two models are supported with optional variants named: M1; M2.1; M2.2; and M2.3. From Tab. 8 above, M1 is for propositions with the default quaternary logic of {F, C, N, T}; and M2.1, 2.2, 2.3 is for theorems with the quaternary logic of {U, I, P, E}.

The processor implements the rules of VŁ4 in six steps to build and calculate tables:

1. Read logical value equivalents and negations by model options:

   False = Unapplied = 00 = 0; [Not:] True = Evaluated = 11 = 1.

2. Read logical value modal conversions by model:

   (F) (U): FC UU EU UP UI; ... ; (T)(E): NT EE UE IE PE.

3. Read logical value connective truth table rows by model:

   &FCNT, FFFFF, CFCUC, NFUNN, TFCNT.

4. Read algebraic form of 4096 combinations for antecedent, conditional, and

consequent as literal propositions, theorems, and connectives:

~s & ~p; ~D & ~A.

5. Calculate atomic propositions and theorems as logical values:

for two propositions, p = FTFT and q = FFTT; for one theorem A = FCNT.

6. Calculate algebraic antecedent, conditional, and consequent into logic values for model options: for three propositions, ~r & ~q becomes

~(FFFFTTTT) & ~(FFTTFFTT) = (TTFFFFFF).

Step 6 uses a lut from each of steps 1-4 in order, with a result in the form of successive rows of a truth table. (Step 1 is useful in compact systems for translating the same truth tables from Model 1 to Model 2.x.)

The parsed input expression of interest is processed in respective iterations of three subsequent steps

7. An argument result as a truth table is stored from step 6 in the parse tree as the truth table of an intermediate result.

8. Intermediate results from step 7 are assigned as antecedent and consequent to a conditional in the lut of connectives in step 3. Each respective logic value in the argument is evaluated to produce another intermediate result as a truth table and stored back into the parse tree.

9. When a truth table of the final result is obtained, the constituent intermediate truth tables are retrieved from the parse tree to build a final truth table record of the logical value transactions. The format is that of which Tab. 11 and Tab. 12 are a fragment.

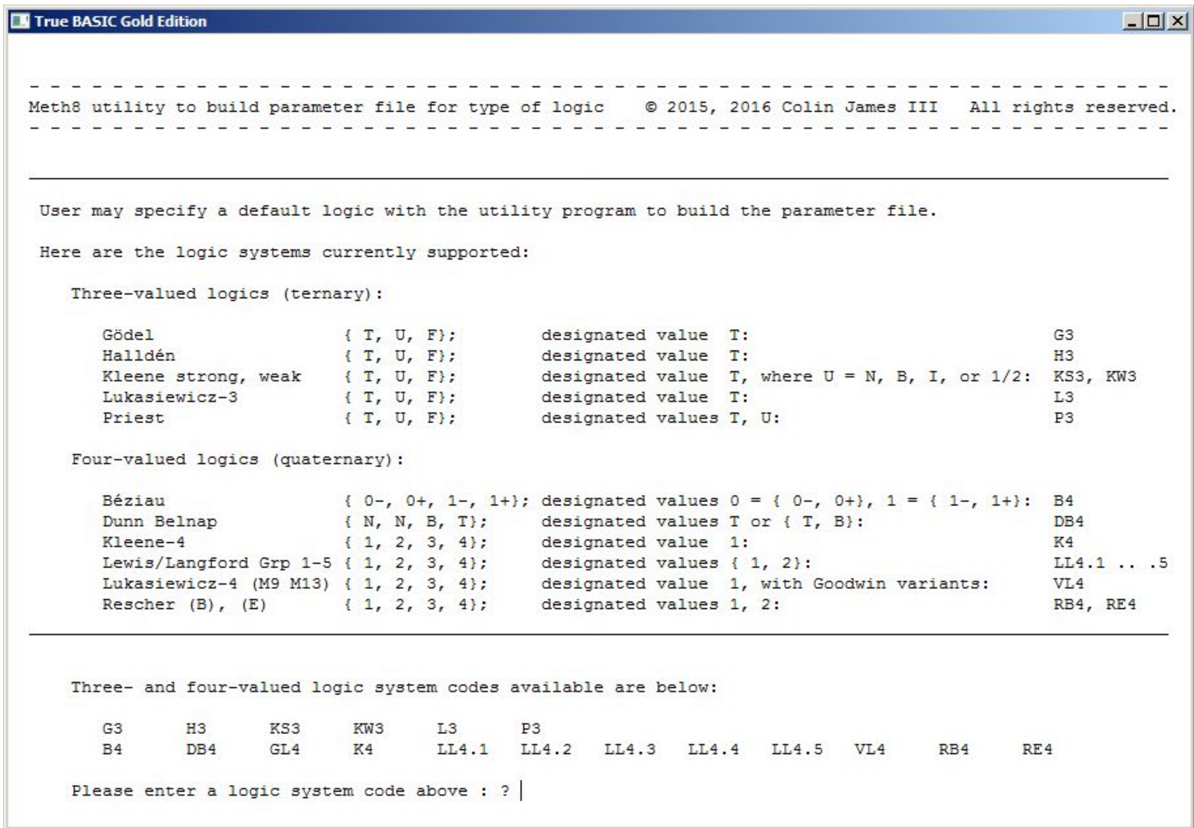
PROVER

The prover component evaluates the final truth table record for invalidation by model of the input expression. The final truth table record and invalidation by model is printed to the user screen and to an evaluation file.

OPERATION

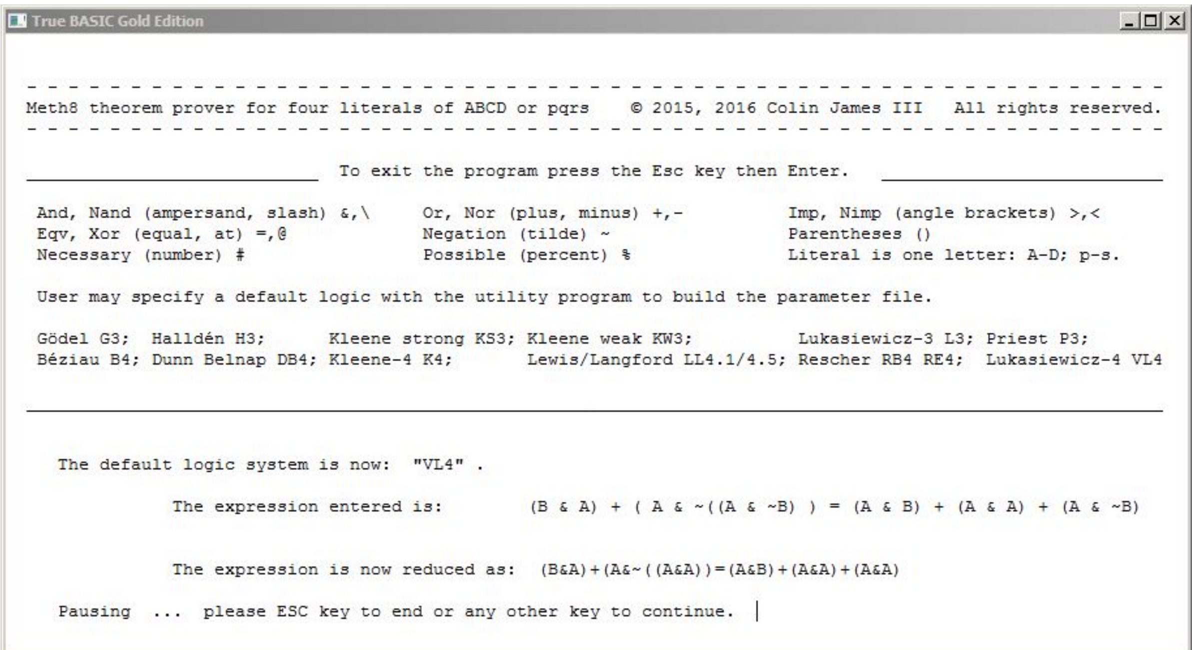
The demonstration version of Meth8 is limited to four propositions (p, q, r, s) and four theorems (A, B, C, D). The systems to be supported are ternary logics [Gödel, K. (1932), Halldén, S. (1949), Kleene, S.C. (1938), Kleene, S.C. (1950), Łukasiewicz, J. (1920), Priest, G. (1979)] and quaternary logics [Belnap, N.D. (1977), Béziau, J-Y. (2011), Lewis, C. I., Langford, H. C. (1959), Kleene, S. C. (1950), Łukasiewicz, J. (1953), Rescher, N. (1965)]

Fig. 3 is the screen of the utility program of Meth8 to specify the logic system and save parameters to a file in root directory of the computer; Fig. 4 is the demo input screen.



```
True BASIC Gold Edition

Meth8 utility to build parameter file for type of logic    © 2015, 2016 Colin James III   All rights reserved.

 User may specify a default logic with the utility program to build the parameter file.

 Here are the logic systems currently supported:

    Three-valued logics (ternary):

       Gödel                  { T, U, F};        designated value  T:                           G3
       Halldén                { T, U, F};        designated value  T:                           H3
       Kleene strong, weak    { T, U, F};        designated value  T, where U = N, B, I, or 1/2: KS3, KW3
       Lukasiewicz-3          { T, U, F};        designated value  T:                           L3
       Priest                 { T, U, F};        designated values T, U:                        P3

    Four-valued logics (quaternary):

       Béziau                 { 0-, 0+, 1-, 1+}; designated values 0 = { 0-, 0+}, 1 = { 1-, 1+}: B4
       Dunn Belnap            { N, N, B, T};     designated values T or { T, B}:                DB4
       Kleene-4               { 1, 2, 3, 4};     designated value  1:                           K4
       Lewis/Langford Grp 1-5 { 1, 2, 3, 4};     designated values { 1, 2}:                     LL4.1 .. .5
       Lukasiewicz-4 (M9 M13) { 1, 2, 3, 4};     designated value  1, with Goodwin variants:    VL4
       Rescher (B), (E)       { 1, 2, 3, 4};     designated values 1, 2:                        RB4, RE4

    Three- and four-valued logic system codes available are below:

       G3     H3     KS3    KW3    L3     P3
       B4     DB4    GL4    K4     LL4.1  LL4.2  LL4.3  LL4.4  LL4.5  VL4   RB4   RE4

    Please enter a logic system code above : ?
```

Fig. 3



```
True BASIC Gold Edition

Meth8 theorem prover for four literals of ABCD or pqrs    © 2015, 2016 Colin James III   All rights reserved.

                           To exit the program press the Esc key then Enter.

 And, Nand (ampersand, slash) &,\     Or, Nor (plus, minus) +,-          Imp, Nimp (angle brackets) >,<
 Eqv, Xor (equal, at) =,@             Negation (tilde) ~                 Parentheses ()
 Necessary (number) #                 Possible (percent) %               Literal is one letter: A-D; p-s.

 User may specify a default logic with the utility program to build the parameter file.

 Gödel G3;  Halldén H3;      Kleene strong KS3; Kleene weak KW3;          Lukasiewicz-3 L3; Priest P3;
 Béziau B4; Dunn Belnap DB4; Kleene-4 K4;       Lewis/Langford LL4.1/4.5; Rescher RB4 RE4;  Lukasiewicz-4 VL4

   The default logic system is now:  "VL4" .

              The expression entered is:       (B & A) + ( A & ~((A & ~B) ) = (A & B) + (A & A) + (A & ~B)

              The expression is now reduced as:  (B&A)+(A&~((A&A))=(A&B)+(A&A)+(A&A)

   Pausing  ...  please ESC key to end or any other key to continue.
```

Fig. 4

FINAL REMARKS

Whilst the bivalent framework is a model to weaken a class of logics with twin functors, Meth8 is also capable of testing a range of well known many valued logics. Significantly, a later version will approach different logics as alternative classes of minimal sets. It is intended Meth8 will allow the user to explore many valued logic and test practical examples of logics that comply to Suszko's Thesis. [Suszko, R. (1977)]

ACKNOWLEDGEMENTS

Thanks are due to the anonymous referees for helpful comments.

REFERENCES

Belnap, N.D. (1977). A useful four-valued logic, in J.M. Dunn, G. Epstein (eds.), *Modern Uses of Multiple-Valued Logic*, Dordrecht: Reidel, 8–37.

Béziau, J-Y. (2011). A New Four Valued Approach to Modal Logic *Logique et Analyse,* 54.

Dugundji, J. (1940). Note on a Property of Matrices for Lewis and Langford's Calculi of Propositions. The Journal of Symbolic Logic, 5 (4), 150-151.

Gödel, K. (1932). Zum intuitionischen Aussagenkalkül. Anzeiger der Akademie der Wissenschaften in Wien 69, 65–66.

Halldén, S. (1949). The logic of nonsense. Uppsala University, Uppsala.

James, C. (2015a). First World Conference: Analogy. Beneméita Universidad Autónoma de Puebla, Mexico, November 4-6, 20195, Handbook, ISBN 978-83-65273-01-1, 50-51.

James, C. (2015b). U.S. Patent No. 9,202,166, Method and system for Kanban cell neuron network, December 1, 2015.

Kleene, S.C. (1938). On a Notation for Ordinal Numbers, The Journal of Symbolic Logic, 50–155.

Kleene, S. C. (1950) Introduction to Metamathematics. D. Van Nostrand, Princeton, NJ.

Lewis, C. I., Langford, H. C. (1959). Symbolic Logic (Second Edition). New York: Dover Publications, 493-494.

Łukasiewicz, J. (1920). On Three-valued Logic, in L. Borkowski (ed.), Amsterdam, North-Holland, 1970, pp. 87-88.

Łukasiewicz, J. (1953). A system of Modal Logic. The Journal of Computing Systems, 1, 111-149.

Łukasiewicz, J. (1957). Aristotle's Syllogistic Logic (Second Edition). Clarendon Press, Chapter VII.

Priest, G. (1979). The Logic of Paradox. *Journal of Philosophical Logic*, Vol. 8, No. 1, Jan, 219-241.

Rescher, N. (1965), Notre Dame Journal of Formal Logic Volume VI, Number 2, April, 154-156.

Suszko, R. (1977). The Fregean axiom and Polish mathematical logic in the 1920's. *Studia Logica*, 36:373–380.