Interpreting LAp into $V\oplus L$ and $V\#L$

Lila Fontes

February 11, 2010

This document's goal is to interpret Soltys' theory LAp [SK01, SC04] two times. Once over the field $\mathbb{F}_2$ into the theory $V\oplus L$, and once over the integral domain $\mathbb{Z}$ into the theory $V\#L$ [Fon09]. Both theories capture basic arithmetic and include a function expressing matrix powering.

Each interpretation is a mapping from LAp to $V\oplus L$ (respectively, $V\#L$) of terms and formulas such that provability is preserved. That is, theorems in LAp are mapped to theorems of $V\oplus L$ ($V\#L$). Such an interpretation has two uses. First, it allows for a shortcut when working in $V\oplus L$ ($V\#L$): no duplication of Soltys' work must be done in order to show that theorems proved in LAp are also provable in $V\oplus L$ ($V\#L$). Second, it fits Soltys' otherwise standalone hierarchy of theories LA $\subset$ LAp $\subset$ $\forall$LAp into the hierarchy of theories established in [CN10].

LAp has three sorts – indices ($\mathbb{N}$), field elements, and matrices, which we think of semantically as having field elements as entries. $V\oplus L$ ($V\#L$) has only two sorts, numbers ($\mathbb{N}$) and binary strings, which are used to encode matrices, lists, and other data structures, as required. The technical details of interpreting LAp into $V\oplus L$ ($V\#L$) arise from this difference in number of sorts. The number sort of LAp can be directly interpreted as the number sort of $V\oplus L$ ($V\#L$), but the field sort and the matrix sort present a non-obvious technical point. This document motivates, explains, and proves correct an interpretation whose main point of interest is this non-obvious interpretation of the field and matrix sorts into binary strings.

Contents

1 A summary of LAp

A survey of LAp is available in [SC04]; a fully-detailed explanation is available in [SK01].

LAp has three sorts: indices (represented by i, j, k), field elements (a, b, c), and matrices (A, B, C). Matrices have field elements as entries.

The language $\mathcal{L}_{LA}$ of LAp comprises the symbols:

$$\begin{gathered}0_{\text{index}}, 1_{\text{index}}, +_{\text{index}}, *_{\text{index}}, -_{\text{index}}, \text{div}, \text{rem}, \\ 0_{\text{field}}, 1_{\text{field}}, +_{\text{field}}, *_{\text{field}}, -_{\text{field}}, ^{-1}, \text{r}, \text{c}, \text{e}, \textstyle\sum, \\ \leq_{\text{index}}, =_{\text{index}}, =_{\text{field}}, =_{\text{matrix}}, \text{cond}_{\text{index}}, \text{cond}_{\text{field}}\end{gathered}$$

The symbol $-_{\text{index}}$ is cutoff subtraction; $\text{div}(i,j)$ and $\text{rem}(i,j)$ are the quotient and remainder functions; $0^{-1} =_{\text{field}} 0$; $\text{r}(A)$ and $\text{c}(A)$ return the number of rows and columns in A; $\sum(A)$ is the sum of all the entries of A; and for α a formula (with all atomic subformulas of the form $m \leq n$ and $m = n$), $\text{cond}(\alpha, i, j)$ is i if α is true and j otherwise (and similarly for field elements). The function $\text{e}(A, i, j)$ is the $(i,j)^{\text{th}}$ entry of matrix A. (See remark 1 below.)

There is an additional symbol, P, for matrix powering. The i^{th} power of matrix A is given by $\text{P}(i, A)$.

Terms of type index are represented by n and m; terms of type field are represented by t and u, and terms of type matrix are represented by T and U. Formulas are represented by α and β. (This holds throughout this section and the rest of the document.) Terms and formulas are defined inductively, as follows:

0_{index}, 1_{index}, 0_{field}, 1_{field}, and variables of all three types are terms. Terms of type index also include combinations of terms of type index using the $+_{\text{index}}$, $-_{\text{index}}$, $*_{\text{index}}$, div, rem, r, and c functions as expected. Terms of type field also include combinations of terms of type field using the $+_{\text{field}}$, $-_{\text{field}}$, $*_{\text{field}}$, and $^{-1}$ functions as usual, as well as the terms $\text{e}(T, m, n)$ and $\sum(T)$.

Terms of type matrix also include the constructed term $\lambda_{ij}\langle m, n, t\rangle$ (with the restriction that i and j are not free in m and n). It defines a $r \times c$ matrix with $(i,j)^{\text{th}}$ entry given by $t(i,j)$. Constructed λ_{ij} terms can be used to define many matrix functions — multiplication, addition, transpose, inverse, etc. — using a field function to compute each entry. This feature avoids the need for separately-defined function symbols.

Atomic formulas are of the form $m \leq_{\text{index}} n$, $m =_{\text{index}} n$, $t =_{\text{field}} u$, and $T =_{\text{matrix}} U$. Formulas include $\neg\alpha$, $\alpha \wedge \beta$, and $\alpha \vee \beta$. Notice that all formulas of LAp are open.

If α is a formula with atomic subformulas all of the form $m \leq_{\text{index}} n$ and $m =_{\text{index}} n$, then $\text{cond}_{\text{index}}(\alpha, m', n')$ is a term of type index and $\text{cond}_{\text{field}}(\alpha, t, u)$ is a term of type field.

Following the numbering from [SC04], the axioms of LAp are numbered **A1** through **A34**, together with **A35** and **A36** for matrix powering. The rules of inference are given by PK, with additional rules for induction and matrix equality [SK01, SC04]. These axioms and rules are all restated in section 3 below.

Remark 1 On a technical note, matrices in LAp have three "attributes:" number of rows, number of columns, and the values of matrix entries. Matrix entries are numbered from 1 to $\text{r}(A)$ and 1 to $\text{c}(A)$. If queried on i or $j = 0$ or $> \text{r}(A)$ or $\text{c}(A)$ (respectively), then $\text{e}(A, i, j) = 0$ by definition.

LAp takes advantage of this default behavior to allow non-standard mathematical operations: matrices of inappropriately-matched sizes can be haphazardly added and multiplied. (This works because an $n \times m$ matrix can be used as an $(n + k) \times (m + j)$ matrix by implicit padding with zeroes.) This feature is useful but requires additional notice; for example, non-square matrices can be raised to a power. See page 13 for more details.

2 A summary of $V\oplus L$

$V\oplus L$ has two sorts: numbers and strings. Numbers are denoted with lowercase letters x, y, z, and strings with uppercase X, Y, Z.

The language of 2-sorted arithmetic is $\mathcal{L}_A^2 = [0, 1, +, \cdot, |\,|, =_1, =_2, \leq, \in]$. 0 and 1 are constants of the number sort; $+$, $\cdot$, $=_1$, and $\leq$ are functions and predicate symbols over the number sort. $=_2$ is string equality. Binary strings are considered as the characteristic vectors of finite sets of natural numbers. $|X|$ is meant to represent the "length of X" and has value 1+ the largest element of X. (Thus the set $\{0, 1, 3\}$ is represented by string '1101' and has length 4.) Set membership is represented by $x \in X$, and is abbreviated $X(x)$.

There is an additional function symbol, Pow_2, for matrix powering over $\mathbb{F}_2$.

Terms of type number are represented by s and t; terms of type string are represented by S and T. Formulas are represented by φ and ψ.

0, 1, and number variables are number terms. Number terms also include combinations of number terms with the function symbols $+$ and $\cdot$, as well as $|T_1|$ for any string term T_1. String variables are string terms.

The logical constants $\top$ and $\bot$ are atomic formulas. Atomic formulas are also of the form $s =_1 t$, $s \leq t$, $S =_2 T$, and $S(t)$. Formulas include $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\forall x\varphi$, $\exists x\varphi$, $\forall X\varphi$, and $\exists X\varphi$.

The axioms of $V{\oplus}L$ are given by **B1** through **B12**, **L1**, **L2**, **SE**, and Σ_0^B -COMP (those for V^0 [CN10]), together with the axiom stating the existence of a string value for the matrix powering function Pow_2 [Fon09]. (On a technical note, matrices in $V{\oplus}L$ are encoded as strings *separately* from their sizes; see axioms A35 and A36 on page 13.)

The function $Parity(x, Y) = Z$ is defined in section 9D of [CN10]. The string Z witnesses the bit-by-bit computation of the parity of the first x bits of string Y, meaning that bit $Z(i)$ is true when there are an odd number of 1s in the first i bits of Y. Thus the parity of string Y is given by the boolean term $Parity(|Y|, Y)(|Y|)$. The expression PARITY(Y) will be used as shorthand for this in the discussion below.

3 Interpreting LAp over $\mathbb{F}_2$ into $V{\oplus}L$

The theories $V{\oplus}L$ and $\overline{V{\oplus}L}$ can be considered as having Boolean field elements. This observation is the basis of the interpretation below. The index sort is directly interpreted as the number sort, the field sort ($\mathbb{Z}_2$) is interpreted below as Boolean-valued formulas, and the matrix sort is interpreted as strings.

3.1 Interpreting terms and formulas

Preserving the notation from LAp in section 1 above, let i, j, and k represent indices, and m and n terms of type index; let a, b, and c represent field elements, and t and u terms of type field; let A, B, and C represent matrices, and T and U terms of type matrix; and let α and β represent formulas. When interpreted into $V{\oplus}L$, a superscript $^\sigma$ is added: i^σ, m^σ, T^σ, etc., are terms in $V{\oplus}L$. (Note that this breaks with the standard notation from $V{\oplus}L$, in which lowercase letters are numbers and uppercase letters are strings.)

This interpretation requires the introduction of predicate symbol eq, function symbols $\dot{-}$, f_{div}, f_{rem}, f_{r}, f_{c}, and functions f_φ and F_φ for every formula φ.[1] These functions have Σ_0^B definitions, and so they are defined in the universal conservative extension $\overline{V{\oplus}L}$. Aside from Σ and P, the functions of LAp are all definable in $\overline{V^0}$. Defining Σ requires $PARITY$; thus the base theory LA over $\mathbb{F}_2$ is interpretable into $\overline{V^0(2)}$. Conveniently, LAp is obtained from LA by addition of the two axioms defining powering function P, and $V{\oplus}L$ is obtained from $V^0(2)$ by addition of the axiom defining $PowSeq_2$.

Variables of the index sort are interpreted as variables of the number sort. Variables of the field and matrix sorts are interpreted as variables of the string sort.

3.1.1 Index sort

The index sort from LAp can be directly interpreted as the number sort in $V{\oplus}L$. In general, an index term with k free variables will be interpreted as a number term with k free variables. Free

[1]Using f_φ to interpret cond, we are only interested in the case where the formula φ is an interpretation of a formula in LAp all of whose atomic sub-formulas have the form $m = n$ or $m \leq n$. There is no such restriction for F_φ, which interprets λ terms.

variables of index terms are suppressed below for readability.

The 0 in LAP is interpreted as 0 in $V\oplus L$, 1 is interpreted as 1 in $V\oplus L$, etc.:

LAP	$V\oplus L$	
0_{index}	0	
1_{index}	1	
i	i	variables map to variables
$m +_{\text{index}} n$	$m^\sigma + n^\sigma$	
$m *_{\text{index}} n$	$m^\sigma \cdot n^\sigma$	
$m -_{\text{index}} n$	$m^\sigma \dot{-} n^\sigma$	'$\dot{-}$' is standard limited subtraction, defined: $x \dot{-} y = z \leftrightarrow (x = y + z) \vee (z = 0 \wedge x < y)$
$\text{div}(m, n)$	$f_{\text{div}}(m^\sigma, n^\sigma)$	f_{div} is a number function with graph: $f_{\text{div}}(x, y) = z \leftrightarrow y \cdot z \leq x \wedge x < y \cdot (z + 1)$
$\text{rem}(m, n)$	$f_{\text{rem}}(m^\sigma, n^\sigma)$	f_{rem} is a number function with graph: $f_{\text{rem}}(x, y) = z \leftrightarrow z + y \cdot f_2(x, y) = x$ or alternatively, $f_{\text{rem}}(x, y) = z \leftrightarrow z < y \wedge \exists d \leq x(x = d \cdot y + z)$
$\text{cond}_{\text{index}}(\alpha, m, n)$	$f_{\alpha^\sigma}(m^\sigma, n^\sigma)$	where for every formula φ, f_φ is defined: $f_\varphi(x, y) = z \leftrightarrow (\varphi \wedge x = z) \vee (\neg\varphi \wedge y = z)$
$m =_{\text{index}} n$	$m^\sigma =_1 n^\sigma$	
$m \leq_{\text{index}} n$	$m^\sigma \leq n^\sigma$	

3.1.2 Field sort

The field sort of LAP over $\mathbb{F}_2$ is binary-valued. Each term of type field is interpreted into a formula in $V\oplus L$ whose truth value corresponds to the binary value of the original term. (It was previously discussed that it might be necessary to interpret all field elements as strings of length one in $V\oplus L$, but this does not seem necessary.)

Variables of type field are interpreted as the first entry of a 1×1 matrix variable (that is, a string). Thus a field variable a is interpreted as the formula $X_a(1, 1)$, where X_a is a free string variable. (See section 3.1.3 below for explanatory details of the interpretation of matrices.)[2]

Thus a field term with k free variables will be interpreted as a formula with k free variables (each index variable corresponding to a number variable, and each field or matrix variable corresponding to a string variable). Free variables of field terms are suppressed below for readability.

For t and u terms of type field, we interpret:

LAP	$V\oplus L$	
0_{field}	$\bot$	
1_{field}	$\top$	
a	$X_a(1, 1)$	see above: variables map to string variables
$t +_{\text{field}} u$	$t^\sigma \oplus u^\sigma$	that is, $t^\sigma \,\text{XOR}\, u^\sigma$
$t -_{\text{field}} u$	$t^\sigma \oplus u^\sigma$	
$t *_{\text{field}} u$	$t^\sigma \wedge u^\sigma$	
t^{-1}	t^σ	
$\text{cond}_{\text{field}}(\alpha, t, u)$	$(\alpha^\sigma \wedge t^\sigma) \vee (\neg\alpha^\sigma \wedge u^\sigma)$	
$t =_{\text{field}} u$	$t^\sigma \leftrightarrow u^\sigma$	

3.1.3 Matrix sort

The matrix sort of LAP has entries from the field sort (in this instance, $\mathbb{F}_2$). Every matrix in LAP has three attributes: number of rows, number of columns, and matrix entries. The matrix sort is

[2]Alternatively, a field variable could be interpreted into a formula $x = 0$ for a free number variable x. This shorter interpretation "maps" field variables into number variables. This might be a better interpretation, since interpreting field variables into string variables as above implicitly assumes that the string variable X encodes a matrix appropriately. Strings X not encoding a matrix will not cause any problems with this interpretation – they are treated as strings encoding a 0×0 matrix.

interpreted as strings in $V\oplus L$. The string sort in $V\oplus L$ is used to encode many different formats of data. Here, the intention is to use a binary string to encode a triple: two numbers (the dimensions of the matrix) and a matrix of Boolean values. The pairing function will facilitate this encoding, by means of the same usage and encoding format as is used in [CN10, Fon09]. Additionally, since field elements in LAp are interpreted as Boolean-valued formulas in $V\oplus L$, the format of retrieving matrix entries is cleanly interpreted.

An $a \times b$ matrix A is interpreted as a string A^σ such that $A^\sigma(0, \langle a, b\rangle)$ is true, and for every i and j such that $\mathrm{e}(A, i, j) = A_{ij} = 1$, the bit $A^\sigma(i, j)$ is true. All other bits of A^σ are false. (Thus a string $A^\sigma = X$ encodes a matrix whose zero$^{\text{th}}$ row is all zeroes except the $\langle a, b\rangle^{\text{th}}$ place, with the rows of A stored in the other "rows" of X.) This interpretation preserves the feature from LAp that matrices, when queried out-of-bounds, return 0. (As in LAp, a matrix queried on its 0^{th} row must return 0; that row is reserved in our encoding, so this is a special case.)

Throughout this section, it is convenient to have a formula $isMatrix_2(X)$ which is true if and only if string X is a valid encoding of a matrix over the field $\mathbb{F}_2$. Let $isMatrix_2(X)$ be:

$$\begin{aligned}\exists x, y < |X| \big[\overbrace{X(0, \langle x, y\rangle)}^{\text{(a)}} \wedge \forall z < |X| \, \big[&\overbrace{(z = 0 \vee \neg X(z, 0)) \wedge (z = \langle x, y\rangle \vee \neg X(0, z))}^{\text{(b)}} \\ &\wedge \underbrace{\big(X(z) \to (Pair(z) \wedge \mathrm{left}(z) \leq x \wedge (\mathrm{left}(z) = 0 \vee \mathrm{right}(z)) \leq y\big)}_{\text{(c)}} \big]\big]\end{aligned} \tag{1}$$

This formula has three sections: section (a) guarantees that the matrix dimensions are encoded properly; section (b) checks that there are no stray matrix entries in the 0^{th} row or column; and section (c) checks that the matrix entries are encoded properly and within the bounds of the matrix dimensions. Matrices have many "wasted" bits of empty space, due to the use of pairing numbers to encode rows and columns of data.

The shorthand formula $isMatrix_2(X)$ simplifies the design of the following formulas of the interpretation. Strings not representing matrices are treated as 0×0 matrices when considered in the $V\oplus L$ "image" of an interpreted formula from LAp. That is, the interpretation below is arranged so that the axioms of LAp are theorems of $V\oplus L$ without the addition of a conditional clause for "when X is a valid encoding of a matrix, . . ." This is further explained in section 3.2.

LAp	$V\oplus L$
A	A

Variables of type matrix are interpreted as variables of type string. As with the other sorts, a matrix term with k free variables is interpreted as a string term with k free variables. Free variables for matrix terms are suppressed below for readability. Special consideration is required for the free variables of λ_{ij} terms.

LAp	$V\oplus L$	
$\mathrm{r}(T)$	$f_r(T^\sigma)$	f_r is a number function with graph: $f_r(X) = z \leftrightarrow$ $\underbrace{\big(isMatrix_2(X) \wedge \exists y \leq \lvert X\rvert(X(0, \langle z, y\rangle))\big)}_{\text{for matrices}} \vee \underbrace{\big(z = 0 \wedge \neg\, isMatrix_2(X)\big)}_{\text{for strings not encoding matrices}}$
$\mathrm{c}(T)$	$f_c(T^\sigma)$	where f_c is a number function with graph: $f_c(X) = z \leftrightarrow$ $\underbrace{\big(isMatrix_2(X) \wedge \exists y \leq \lvert X\rvert(X(0, \langle y, z\rangle))\big)}_{\text{for matrices}} \vee \underbrace{\big(z = 0 \wedge \neg\, isMatrix_2(X)\big)}_{\text{for strings not encoding matrices}}$

The r and c functions recover the size of the matrix from the first "row" of the string. Note that they also check that the string is a valid encoding of a matrix. These functions are well-defined: when X does not encode a matrix, $f_r(X) = 0 = f_c(X)$. The values of $\mathrm{r}(X)$ and $\mathrm{c}(X)$ are implicitly bounded by $|X|$.

LAp	$V\oplus L$
$\sum(T)$	$\underbrace{\neg\,\mathrm{PARITY}(T^\sigma)}_{\text{for matrices}} \wedge \underbrace{isMatrix_2(T^\sigma)}_{\text{true only for strings encoding matrices}}$

The $\sum$ function returns the sum of the entries of a matrix. This is of type field in LAp, and its interpretation is a formula in $V\oplus L$. Working over $\mathbb{F}_2$, the sum of field entries of a matrix is is simply the parity of the entries. If X is a string encoding a matrix T^σ, then the parity of X is the parity (that is, sum mod 2) of the entries of T *and* the single 1 bit encoding the size of the matrix. The PARITY of the string is reversed to correct for this extra bit and obtain the parity of the matrix entries.

String X encodes a matrix T^σ if and only if then $\mathit{isMatrix}_2(X)$ is true. Any string X not encoding a matrix is semantically interpreted as the 0×0 matrix; thus the sum of entries is 0. The interpretation maps 0 to a false formula; if X is not a matrix, then this formula is false, as required. This is useful in section 3.2.

LAp	$V\oplus L$
$\lambda_{ij}\langle m, n, t\rangle$	$F_{t^\sigma}(m^\sigma, n^\sigma)$

Constructed λ_{ij} matrix terms have the restriction that distinguished variables i and j are free in t and not free in m and n. In LAp, the λ operator binds i and j. Thus, when interpreted, t^σ has two reserved free variables i^σ and j^σ (the interpretations of i and j). The restrictions are preserved: i^σ and j^σ are free in t^σ and not free in m^σ and n^σ. Thus, suppressing free variables, $F_{t^\sigma}(m^\sigma, n^\sigma)$ is the string interpreting $\lambda_{ij}(m, n, t)$ in $V\oplus L$.

If $t(i, j)$ is a field term in LAp, then t^σ is a $V\oplus L$ formula $\varphi(i, j)$ with the same number of free variables. (Recall from section 3.1.2 that field variables are interpreted into string variables.) For every formula $\varphi(i, j)$, the string function $F_\varphi(m, n)(b)$ is bit-defined:

$$b = \langle 0, \langle m, n\rangle\rangle \vee \exists i \leq m \exists j \leq n(i > 0 \wedge j > 0 \wedge b = \langle i, j\rangle \wedge \varphi(i, j)) \tag{2}$$

Thus the output of $F_\varphi(m, n)$ is a string encoding a matrix with m rows and n columns, with entries determined by φ. This aligns with LAp's definition of λ_{ij} term, so that $F_{t^\sigma}(m^\sigma, n^\sigma)$ is the string interpretation in $V\oplus L$ of the matrix $\lambda_{ij}(m, n, t)$ in LAp.

Notice that $\mathit{isMatrix}_2(F_\varphi(m, n))$ is always true.

The interpretation of λ_{ij} terms is required when those terms are used inside formulas; the existence of such a string is guaranteed in $\overline{V\oplus L}$ by Σ_0^B **-COMP**. By general theorems, these quantifier-free Σ_0^B formulas in $\overline{V\oplus L}$ translate to Σ_1^B formulas in the base theory $V\oplus L$.

LAp	$V\oplus L$
$\mathrm{e}(T, m, n)$	$T^\sigma(m^\sigma, n^\sigma) \wedge m^\sigma > 0 \wedge n^\sigma > 0 \wedge \underbrace{\mathit{isMatrix}_2(T^\sigma)}_{\text{true only for valid strings encoding matrices}}$

The e function simply returns the matrix's entry at the requested coordinates; this is straightforward, given the encoding of matrices into strings. LAp specifies that matrices return 0 when queried out-of-bounds. As in LAp, matrix row and column numbering starts at 1 for strings encoding matrices. Hence querying at $(0, x)$ or $(x, 0)$ is out-of-bounds and returns 0. Conveniently, on all bits in a row or column larger than its dimensions, a string also returns 0 (false).

Strings which do not encode matrices are interpreted as 0×0 matrices, and accordingly return 0 (false) for every lookup.

LAp	$V\oplus L$
$T =_{\text{matrix}} U$	$(\mathrm{r}(T) = \mathrm{r}(U))^\sigma \wedge (\mathrm{c}(T) = \mathrm{c}(U))^\sigma \wedge \forall i, j \leq \lvert T^\sigma\rvert + \lvert U^\sigma\rvert(\mathrm{e}(i, j, T) = \mathrm{e}(i, j, U))^\sigma$

Two matrices are equal in LAp if and only if they have the same dimensions and entries. If two matrices are equal in LAp then their interpretations are equal as strings in $V\oplus L$. (The converse is not true because there are many different strings that "encode" the 0×0 matrix.) Specifying the above formula for matrix equality handles these cases; the formula evaluates two different strings as "equal as matrices" if they both encode the same matrix.

LAp	$V\oplus L$
$\mathrm{p}(i, A)$	$F_p(i, A^\sigma)$

The P function computes the power of a matrix. The $V\oplus L$ function Pow_2 (definable from $PowSeq_2$) also computes the power of a matrix.

A technical point obfuscates this otherwise-straightforward interpretation. Each matrix in LAP is associated with its dimensions r and c, and has entries starting with $(1,1)$. To preserve this information, matrices are interpreted as strings encoding *both* the matrix entries and dimensions. However, the Pow_2 function was defined for a different matrix encoding in $V\oplus L$, in which a string encodes the bits of a matrix starting with $(0,0)$; the size of the matrix is not included in the string.

Thus we need two Σ_0^B string functions. The first one strips the dimensions from the string, converting a matrix encoded according to our interpretation (of LAP into $V\oplus L$) into a matrix according to the standard in $V\oplus L$.

$$Strip(X)(i,j) \leftrightarrow X(i+1, j+1)$$

The second function adds the dimension-encoding "wrapper" back, converting a standard-form $V\oplus L$ matrix into the form of an interpreted matrix from LAP.

$$Wrap(r,c,X)(b) \leftrightarrow b = \langle 0, \langle r,c\rangle\rangle \vee \exists 0 < i,j < b(b = \langle i,j\rangle \wedge X(i,j))$$

Given these two functions, let $F_p(i, A^\sigma)$ interpret the LAP function $\mathrm{P}(i,A)$, and let $r_A = f_r(A^\sigma)$ be the interpretation of $\mathrm{r}(A)$, and $c_A = f_c(A^\sigma)$ be the interpretation of $\mathrm{c}(A)$. For strings X that validly encode a matrix A^σ, and powers $i > 0$,

$$F_p(i,X) = Wrap(r_A, c_A, Pow_2(\max(r_A, c_A), i, Strip(X)))$$

There is a special case: if $i = 0$, then LAP specifies that the zeroth power of a matrix A is the $\mathrm{r}(A) \times \mathrm{r}(A)$ identity matrix.

$$F_p(0,X) = Wrap(r_A, r_A, Pow_2(r_a, 0, Strip(X)))$$

Otherwise, for strings X that do not validly encode a matrix, we want F_p to behave as if X encoded the 0×0 matrix. The powers of this matrix are all identical: the 0×0 matrix. This is encoded as a string with only one true bit, the $\langle 0,0\rangle$ bit of its first row. Thus in this case,

$$F_p(i,X)(b) \leftrightarrow b = \langle 0, \langle 0,0\rangle\rangle$$

Combining these, we can bit-define F_p as follows:

$$\begin{aligned} F_p(i,A^\sigma)(b) \quad \leftrightarrow \quad & \big[isMatrix_2(X) \wedge Wrap(r_A, c_A, Pow_2(f_{r_A=0}(r_A, \max(r_A,c_A)), i, Strip(A^\sigma)))(b)\big] \\ & \vee \big[\neg\, isMatrix_2(X) \wedge b = \langle 0, \langle 0,0\rangle\rangle\big] \end{aligned}$$

3.2 Provability is preserved

This interpretation is useful because it preserves provability: for any sequent $\rightarrow \varphi$ provable in LAP, the interpreted formula φ^t is provable in $\overline{V\oplus L}$. Because $\overline{V\oplus L}$ is a conservative extension of $V\oplus L$, it is sufficient to show that the interpretation from LAP to $\overline{V\oplus L}$ preserves provability.

Deduction in LAP follows PK rules of inference for sequents: exchange; weakening; connectives $\neg$, $\wedge$, $\vee$; induction; substitution; and matrix equality. These deductive rules (except induction and matrix equality) are logically sound, so the same deductions can be used in $V\oplus L$ proofs. The induction rule is easily proven using the induction axiom schema in $V\oplus L$. The matrix equality rules are trivially true in $V\oplus L$, since the interpretation of matrix equality fits the rule exactly (matrices are equal iff they have the same dimensions and entries). Thus it suffices to show that the interpretations of LAP axioms are provable in $V\oplus L$.

Each LAP sequent of the form

$$\alpha_1, \ldots, \alpha_k \rightarrow \beta_1, \ldots, \beta_t \tag{3}$$

is interpreted as a $V\oplus L$ formula[3]

$$\bigwedge_{i=1}^{k} \alpha_i^\sigma \supset \bigvee_{j=1}^{t} \beta_j^\sigma \tag{4}$$

Provability is preserved if every sequent (3) provable in LAP translates to a formula (4) provable in $V\oplus L$.

The following lemma is useful in proving A32$^\sigma$ and A33$^\sigma$ below. It states that if two strings differ on exactly one bit, then they have opposite parities.

Lemma 2 *$V\oplus L$ proves:*

$$(X(k) \leftrightarrow \neg Y(k)) \wedge \big(\forall i < |X| + |Y|\; i \neq k \leftrightarrow (X(i) \leftrightarrow Y(i))\big) \supset \big(\text{PARITY}(X) \leftrightarrow \neg\,\text{PARITY}(Y)\big)$$

The proof of the lemma proceeds by induction on the bits of the witness strings computing the parities of X and Y. The inverse "if two strings are identical on every bit, then they have the same parity" is true by properties of equality.

$V\oplus L$ has axioms:

B1. $x + 1 \neq 0$	B7. $(x \leq y \wedge y \leq x) \supset x = y$
B2. $x + 1 = y + 1 \supset x = y$	B8. $x \leq x + y$
B3. $x + 0 = x$	B9. $0 \leq x$
B4. $x + (y + 1) = (x + y) + 1$	B10. $x \leq y \vee y \leq x$
B5. $x \cdot 0 = 0$	B11. $x \leq y \leftrightarrow x < y + 1$
B6. $x \cdot (y + 1) = (x \cdot y) + x$	B12. $x \neq 0 \supset \exists y \leq x(y + 1 = x)$
L1. $X(y) \supset y < \lvert X\rvert$	L2. $y + 1 = \lvert X\rvert \supset X(y)$

SE. $[|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i)) \supset X = Y$

The axiom scheme Σ_0^B−COMP: for every Σ_0^B−COMP formula $\varphi(z)$,

$$\exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z))$$

And the axiom for matrix powering.

In this section, each interpretation of an axiom from LAP is shown to be provable in $V\oplus L$. The axioms for LAP are numbered A1-A34, and grouped by type.

The equality axioms for LAP are A1–A5. The equality axioms are all valid in $V\oplus L$ since in every $\mathcal{L}_A^2$ structure, equality is interpreted as true equality.

A1. $\rightarrow x = x$ A1$^\sigma$. $x = x$

A2. $x = y \rightarrow y = x$ A2$^\sigma$. $x = y \rightarrow y = x$

A3. $(x = y \wedge y = z) \rightarrow x = z$ A3$^\sigma$. $(x = y \wedge y = z) \supset x = z$

A4. $x_1 = y_1, \ldots, x_n = y_n \rightarrow fx_1, \ldots, x_n = fy_1, \ldots, y_n$

A4$^\sigma$. $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \supset f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

A5. $i_1 = j_1, i_2 = j_2, i_1 \leq i_2 \rightarrow j_1 \leq j_2$ A5$^\sigma$. $i_1 = j_1 \wedge i_2 = j_2 \wedge i_1 \leq i_2 \supset j_1 \leq j_2$

The index axioms for LAP are A6-A17.

A6. $\rightarrow i + 1 \neq 0$ A6$^\sigma$. $x + 1 \neq 0$

This is axiom B1.

A7. $\rightarrow i * (j + 1) = (i * j) + i$ A7$^\sigma$. $x \cdot (y + 1) = x \cdot y + x$

This is axiom B6.

A8. $i + 1 = j + 1 \rightarrow i = j$ A8$^\sigma$. $x + 1 = y + 1 \supset x = y$

This is axiom B2.

[3]Recall that $\supset$ is sometimes written $\rightarrow$, and $(\alpha \supset \beta) \wedge (\beta \supset \alpha)$ can be written $\alpha \leftrightarrow \beta$ in $V\oplus L$.

A9. $\rightarrow i \leq i + j$ A9$^\sigma$. $x \leq x + y$

This is axiom B8.

A10. $\rightarrow i + 0 = i$ A10$^\sigma$. $x + 0 = x$

This is axiom B3.

A11. $\rightarrow i \leq j, j \leq i$ A11$^\sigma$. $x \leq y \vee y \leq x$

This is axiom B10.

A12. $\rightarrow i + (j + 1) = (i + j) + 1$ A12$^\sigma$. $x + (y + 1) = (x + y) + 1$

This is axiom B4.

A13. $i \leq j, j \leq i \rightarrow i = j$ A13$^\sigma$. $(x \leq y \wedge y \leq x) \supset x = y$

This is axiom B7.

A14. $\rightarrow i * 0 = 0$ A14$^\sigma$. $x \cdot 0 = 0$

This is axiom B5.

A15. $i \leq j, i + k = j \rightarrow j - i = k$ and $i \nleq j \rightarrow j - i = 0$

A15$^\sigma$. $x \leq y \wedge x + z = y \supset y \dot{-} x = z$ and $x \nleq y \supset y \dot{-} x = 0$

Axiom A15$^\sigma$ is provable by definition of $\dot{-}$.

A16. $j \neq 0 \rightarrow \mathrm{rem}(i, j) < j$ and $j \neq 0 \rightarrow i = j * \mathrm{div}(i, j) + \mathrm{rem}(i, j)$

A16$^\sigma$. $x \neq 0 \supset f_{\mathrm{rem}}(y, x) < x$ and $x \neq 0 \supset y = x \cdot f_{\mathrm{div}}(y, x) + f_{\mathrm{rem}}(y, x)$

A16$^\sigma$ is provable by the definitions of f_{div} and f_{rem}.

A17. $\alpha \rightarrow \mathrm{cond}(\alpha, i, j) = i$ and $\neg\alpha \rightarrow \mathrm{cond}(\alpha, i, j) = j$

A17$^\sigma$. $\alpha^\sigma \supset f_{\alpha^\sigma}(x, y) = x$ and $\neg\alpha^\sigma \supset f_{\alpha^\sigma}(x, y) = y$

A17$^\sigma$ is provable by the definition of f_φ.

The field axioms of LAp are A18-A27. For readability, we adopt the shorthand X_{11} for $X(1, 1)$. Since the interpretation maps terms of type field to formulas, the following axioms' interpretations are true in $V \oplus L$ because they are logical tautologies.

A18. $\rightarrow 0 \neq 1 \wedge a + 0 = a$ A18$^\sigma$. $(\bot \leftrightarrow \neg\top) \wedge ((X_{11} \oplus \bot) \leftrightarrow X_{11})$

A19. $\rightarrow a + (-a) = 0$ A19$^\sigma$. $(X_{11} \oplus \neg X_{11})) \leftrightarrow \bot$

A20. $\rightarrow 1 * a = a$ A20$^\sigma$. $(\top \wedge X_{1,1}) \leftrightarrow X_{11}$

A21. $a \neq 0 \rightarrow a * (a^{-1}) = 1$ A21$^\sigma$. $(X_{11} \leftrightarrow \neg\bot) \supset \big((X_{11} \wedge X_{11}) \leftrightarrow \top\big)$

A22. $\rightarrow a + b = b + a$ A22$^\sigma$. $(X_{11} \oplus Y_{11}) \leftrightarrow (Y_{11} \oplus X_{11})$

A23. $\rightarrow a * b = b * a$ A23$^\sigma$. $(X_{11} \wedge Y_{11}) \leftrightarrow (Y_{11} \wedge X_{11})$

A24. $\rightarrow a + (b + c) = a + (b + c)$ A24$^\sigma$. $\big(X_{11} \oplus (Y_{11} \oplus Z_{11})\big) \leftrightarrow \big((X_{11} \oplus Y_{11}) \oplus Z_{11}\big)$

A25. $\rightarrow a * (b * c) = (a * b) * c$ A25$^\sigma$. $\big(X_{11} \wedge (Y_{11} \wedge Z_{11})\big) \leftrightarrow \big((X_{11} \wedge Y_{11}) \wedge Z_{11}\big)$

A26. $\rightarrow a * (b + c) = a * b + a * c$ A26$^\sigma$. $\big(X_{11} \wedge (Y_{11} \oplus Z(_{11})\big) \leftrightarrow \big((X_{11} \wedge Y_{11}) \oplus (X_{11} \wedge Z_{11})\big)$

A27. $\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = a$ A27$^\sigma$. $\alpha^\sigma \supset \Big(\big((\alpha^\sigma \wedge X_{11}) \vee (\neg\alpha^\sigma \wedge Y_{11})\big) \leftrightarrow X_{11}\Big)$

and $\neg\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = b$ and $\neg\alpha^\sigma \supset \Big(\big((\alpha^\sigma \wedge X_{11}) \vee (\neg\alpha^\sigma \wedge Y_{11})\big) \leftrightarrow Y_{11}\Big)$

The matrix axioms of LAP are A28-A34.[4]

A28. $(i = 0 \lor \mathrm{r}(A) < i \lor j = 0 \lor \mathrm{c}(A) < j) \to \mathrm{e}(A, i, j) = 0$

A28$^\sigma$. $(x = 0 \lor f_r(X) < x \lor y = 0 \lor f_c(X) < y) \supset$
$\big[\big(X(x, y) \land x > 0 \land y > 0 \land \mathit{isMatrix}_2(X)\big) \leftrightarrow \bot\big]$

A29. $\to \mathrm{r}(\lambda_{ij}\langle m, n, t\rangle) = m$

A29$^\sigma$. $f_r(F_{\varphi(x,y,\vec{v},\vec{V})}(x, y, \varphi(x, y, \vec{v}, \vec{V}))) = x$

and $\to \mathrm{c}(\lambda_{ij}\langle m, n, t\rangle) = n$

and $f_c(F_\varphi(x, y, \varphi(x, y, \vec{v}, \vec{V}))) = y$

and $1 \le i, i \le m, 1 \le j, j \le n \to \mathrm{e}(\lambda_{ij}\langle m, n, t\rangle, i, j) = t$

and $(1 \le i \land i \le m \land 1 \le j \land j \le n) \supset \big((F_{\varphi(i,j,\vec{v},\vec{V})}(m, n))(i, j) \leftrightarrow \varphi(i, j, \vec{v}, \vec{V})\big)$

for any Σ_0^B formula $\varphi(x, y, \vec{v}, \vec{V})$

Axioms A28 and A29 are provable in $V{\oplus}L$ by definition of the functions f_r, f_c, and F_φ.

A30. $\mathrm{r}(A) = 1, \mathrm{c}(A) = 1 \to \sum(A) = \mathrm{e}(A, 1, 1)$

A30$^\sigma$. $f_r(X) = 1 \land f_c(X) = 1 \supset$
$\big((\neg\,\mathrm{PARITY}(X) \land \mathit{isMatrix}_2(X)) \leftrightarrow X(1, 1) \land 1 > 0 \land 1 > 0 \land \mathit{isMatrix}_2(X)\big)$

Consider two cases: if $\neg\,\mathit{isMatrix}_2(X)$, then by definition, $f_r(X) = 0 = f_c(X)$ and so A30$^\sigma$ is vacuously true. If $\mathit{isMatrix}_2(X)$ and $f_r(x) = 1 = f_c(X)$, then $X(0, \langle 1, 1\rangle)$ is true (by the definitions of f_r and f_c). By clauses (b) and (c) of $\mathit{isMatrix}_2(X)$, all other bits of X are false except $X(1, 1)$, which may be true. Thus $\neg\,\mathrm{PARITY}(X)$ is true iff $X(1, 1)$ is true. Thus A30$^\sigma$ is provable in $V{\oplus}L$.

A31. $\mathrm{r}(A) = 1, 1 < \mathrm{c}(A) \to \sum(A) = \sum(\lambda_{ij}\langle 1, \mathrm{c}(A) - 1, A_{ij}\rangle) + A_{1\,\mathrm{c}(A)}$

Let $\varphi(i, j, X)$ be the formula $X(i, j) \land i > 0 \land j > 0 \land \mathit{isMatrix}_2(X)$ interpreting $\mathrm{e}(A, i, j)$, and let Y be the string $F_{\varphi(i,j,X)}(1, f_c(X) \dot{-} 1)$. Then A31$^\sigma$ is:

$(f_r(x) = 1 \land 1 < f_c(X)) \supset$
$\big((\neg\,\mathrm{PARITY}(X) \land \mathit{isMatrix}_2(X)) \leftrightarrow (\neg\,\mathrm{PARITY}(Y) \land \mathit{isMatrix}_2(Y) \oplus X(1, f_c(X)))\big)$

To prove A31$^\sigma$, consider two cases: if $\neg\,\mathit{isMatrix}_2(X)$, then by definition, $f_r(X) = 0 = f_c(X)$, and A31$^\sigma$ is vacuously true.

Otherwise, assume $\mathit{isMatrix}_2(X)$. If $f_r(X) = 1$ and $1 < f_c(X)$, then $f_c(X) = (f_c(X) \dot{-} 1) + 1$. Note that by the definition of F_φ, $Y(b)$ is false for any b not a pairing number, and $Y(0, z)$ is true only for $z = \langle 1, f_c(X) \dot{-} 1\rangle$. Similarly, for $j \ge f_c(X)$, $Y(1, j)$ is false, and $Y(i, j)$ is false for all $i > 1$. Thus the only true bits of Y are $Y(0, \langle 1, f_c(X) \dot{-} 1\rangle)$, and (possibly) bits in the first row $Y^{[1]}$, so

$$\mathrm{PARITY}(Y) \leftrightarrow \neg\,\mathrm{PARITY}(Y^{[1]}) \tag{5}$$

By $\mathit{isMatrix}_2$, $X(b)$ is true only if b is a pairing number. By the definitions of f_r and f_c, $X(i, j)$ is true for $\langle i, j\rangle = \langle 0, \langle 1, f_c(X)\rangle\rangle$, and otherwise, only if $0 < i \le 1$ and $0 < j \le f_c(X)$. Thus $|X^{[1]}| \le f_c(X)$ and

$$\mathrm{PARITY}(X) \leftrightarrow \neg\,\mathrm{PARITY}(X^{[1]}) \tag{6}$$

Recall that $\mathrm{PARITY}(Z)$ is shorthand for $\mathit{Parity}(|Z|, Z)(|Z|)$, and for all $b \ge |Z|$, $\mathit{Parity}(|Z|, Z)(b)$ is equivalent to $\mathit{Parity}(|Z|, Z)(|Z|)$. By the definition of F_φ, $\forall j < f_c(X)(Y(1, j) \;\leftrightarrow\; X(1, j))$. Thus

$$\mathit{Parity}(f_c(X) \dot{-} 1, X^{[1]}) = \mathit{Parity}(f_c(X) \dot{-} 1, Y^{[1]}) \tag{7}$$

Since $\forall j > f_c(X) \dot{-} 1$, $Y(1, j)$ is false, $|Y^{[1]}| \le f_c(X) \dot{-} 1$. Thus

$$\mathrm{PARITY}(Y^{[1]}) \leftrightarrow \mathit{Parity}(f_c(X) \dot{-} 1, Y^{[1]})(f_c(X) \dot{-} 1) \tag{8}$$

[4]In [SK01] they are numbered A28-A33 and E; we take the numbering from [SC04], where the axiom E is renumbered as A34.

By the bitwise definition of its graph δ_{parity} in 9D of [CN10],

$$\mathit{Parity}(f_c(X), X^{[1]})(f_c(X)) \leftrightarrow \mathit{Parity}(f_c(X)\dot{-}1, X^{[1]})(f_c(X)\dot{-}1) \oplus X^{[1]}(f_c(X)) \qquad (9)$$

Since $|X^{[1]}| \leq f_c(X)$,

$$\text{PARITY}(X^{[1]}) \leftrightarrow \mathit{Parity}(f_c(X), X^{[1]})(f_c(X)) \qquad (10)$$

Combining (5)–(10), we obtain the consequent of A31$^\sigma$.

This axiom is also provable by induction.

A32. $\mathrm{c}(A) = 1 \rightarrow \sum(A) = \sum(A^t)$

A32$^\sigma$. $f_c(X) = 1 \supset \big((\neg\,\text{PARITY}(X) \wedge \mathit{isMatrix}_2(X)) \leftrightarrow (\neg\,\text{PARITY}(Y) \wedge \mathit{isMatrix}_2(Y))\big)$

where Y is the string: $F_{X(j,i)\wedge j>0\wedge i>0\wedge \mathit{isMatrix}_2(X)}(f_c(X), f_r(X))$

Recall that A^t is shorthand for the formula $\lambda_{ij}\langle c(A), r(A), A_{ji}\rangle$.

To prove A32$^\sigma$, consider two cases: if $\neg\,\mathit{isMatrix}_2(X)$, then $f_c(X) = 0$ and A32$^\sigma$ is vacuously true.

Otherwise, $\mathit{isMatrix}_2(X)$. We can also assume that X encodes a column matrix ($f_c(X) = 1$), as otherwise A32$^\sigma$ is trivially true.

We prove the theorem by induction on rows. Let $\varphi(k, X)$ be the formula for "for string X encoding a 1-column matrix, axiom A32 holds of the $k \times 1$ submatrix of the first k rows of X." To write this formula, we first define the Σ_0^B formula that extracts the submatrix. Let $\text{Abbrev}(k, X)$ be a string function that takes a string X encoding a matrix and outputs the first $k \times 1$ sub-matrix of that string (the first k entries of the first column):

$$\text{Abbrev}(k, X)(b) \leftrightarrow \mathit{isMatrix}_2(X) \wedge \big(b = \langle 0, \langle k, 1\rangle\rangle \vee (0 < \text{left}(b) \leq k \wedge \text{right}(b) = 1 \wedge X(b))\big)$$

By construction, $\text{Abbrev}(k, X)$ correctly encodes a matrix: $\mathit{isMatrix}_2(\text{Abbrev}(k, X))$. The induction proceeds on the formula $\varphi(k, X)$:

$$f_c(\text{Abbrev}(k, X)) = 1 \supset \big(\text{PARITY}(\text{Abbrev}(k, X)) \leftrightarrow \text{PARITY}(\overbrace{F_{\text{Abbrev}(k,X)(j,i)\wedge j>0\wedge i>0}(k, 1)}^{\text{transpose of Abbrev}(k,X)})\big)$$

Notice that this is a simplified version of A32$^\sigma$, with $\text{Abbrev}(k, X)$ substituted for X.

As a base case, $\varphi(0, X)$ is obviously true, as both $\text{Abbrev}(0, X)$ and the string encoding its transpose have exactly one bit true (the bit encoding the matrices' size), and no others.

Assume $\varphi(k, X)$. In order to prove the inductive step $\varphi(k, X) \supset \varphi(k+1, X)$, we need to prove three statements. In shorthand, they are:

(a) $\sum \text{Abbrev}(k+1, X) = \sum \text{Abbrev}(k, X) + X(k+1, 1)$

(b) $\sum \text{Abbrev}(k, X) = \sum \text{Abbrev}(k, X)^t$

(c) $\sum \text{Abbrev}(k+1, X)^t = \sum \text{Abbrev}(k, X)^t + X^t(1, k+1)$

<u>Proof of 1.</u> We want to show:

$$\neg\,\text{PARITY}(X)\wedge \mathit{isMatrix}_2(X) \leftrightarrow (\neg\,\text{PARITY}(\text{Abbrev}(k, X))\wedge \mathit{isMatrix}_2(\text{Abbrev}(k, X)))\oplus X(k+1, 1)$$

The possible "true" bits of X are, in order: $\langle 1, 1\rangle$, …, $\langle k, 1\rangle$, $\langle k+1, 1\rangle$, and $\langle 0, \langle k+1, 1\rangle\rangle$. The possible "true" bits of $\text{Abbrev}(k, X)$ are, in order: $\langle 1, 1\rangle$, …, $\langle k, 1\rangle$, and $\langle 0, \langle k, 1\rangle\rangle$. Note that on bits numbered $\langle *, 1\rangle$, the string $\text{Abbrev}(k, X)$ is true iff X is true, by construction. Thus for the first $\langle k, 1\rangle$ bits, X and $\text{Abbrev}(k, X)$ are identical, so the strings witnessing the computation of their parities are equal:

$$\mathit{Parity}(\langle k, 1\rangle, X) = \mathit{Parity}(\langle k, 1\rangle, \text{Abbrev}(k, X))$$

In fact, the strings are identical up until the bit $\langle k+1, 1\rangle$. At that bit, by the definition of *Parity*, and since $\text{Abbrev}(k+1, X)$ is false on all bits between $\langle k, 1\rangle$ and $\langle k+1, 1\rangle$,

$$\mathit{Parity}(\langle k+1,1\rangle, \text{Abbrev}(k+1,X))(\langle k+1,1\rangle) \quad\leftrightarrow\quad \mathit{Parity}(\langle k+1,1\rangle, \text{Abbrev}(k+1,X))(\langle k,1\rangle) \oplus X(k+1,1)$$

Since $\text{Abbrev}(k, X)(b)$ is false for all $b > \langle k, 1\rangle$ except $b = \langle 0, \langle k, 1\rangle\rangle$,

$$\mathit{Parity}(\langle k{+}1,1\rangle, \text{Abbrev}(k{+}1,X))(\langle k{+}1,1\rangle) \leftrightarrow \mathit{Parity}(\langle k,1\rangle, \text{Abbrev}(k,X))(\langle k,1\rangle) {\oplus} X(k{+}1,1)$$

Each of $\text{Abbrev}(k, X)$ and $\text{Abbrev}(k + 1, X)$ is true on exactly one bit $> \langle k + 1, 1\rangle$ (namely, the bit recording the matrix size, respectively $\langle 0, \langle k, 1, \rangle\rangle$ and $\langle 0, \rangle k + 1, 1\rangle\rangle$). Thus

$$\text{PARITY}(\text{Abbrev}(k, X)) \leftrightarrow \text{PARITY}(\text{Abbrev}(k + 1, X)) \oplus X(k + 1, 1)$$

<u>Proof of 2.</u> This is the inductive hypothesis, so it is true by assumption.

<u>Proof of 3.</u> This proof proceeds similarly to the proof of 1, above.

The inductive step is proven by combining 1, 2, and 3.

A33. $1 < \text{r}(A), 1 < \text{c}(A) \to \sum(A) = \text{e}(A, 1, 1) + \sum(R(A)) + \sum(S(A)) + \sum(M(A))$
Recall that, by definition:
$R(A) := \lambda_{ij}\langle 1, \text{c}(A) - 1, \text{e}(A, 1, i + 1)\rangle$
$S(A) := \lambda_{ij}\langle \text{r}(A) - 1, 1, \text{e}(A, i + 1, 1)\rangle$
$M(A) := \lambda_{ij}\langle \text{r}(A) - 1, \text{c}(A) - 1, \text{e}(A, i + 1, j + 1)\rangle$

These expressions nesting e inside λ make the translation of A33 very long. For readability, let $\eta(i, j, X)$ be the much-repeated formula $X(i, j) \wedge i > 0 \wedge j > 0 \wedge \mathit{isMatrix}_2(X)$ interpreting $\text{e}(A, i, j)$.

Then A33$^\sigma$ is the formula:

$$\begin{aligned} 1 < f_r(X) \wedge 1 < f_c(X) \supset \Bigg(&\neg\,\text{PARITY}(X) \wedge \mathit{isMatrix}_2(X) \leftrightarrow \\ &\Big(\Big\{\Big[X(1,1) \wedge 1 > 0 \wedge 1 > 0 \wedge \mathit{isMatrix}_2(X) \\ &\oplus(\neg\,\text{PARITY}(F_{\eta(i,j,X)}(1, f_c(X) \dot{-} 1)) \wedge \mathit{isMatrix}_2(F_{\eta(i,j,X)}(1, f_c(X) \dot{-} 1)))\Big] \\ &\oplus(\neg\,\text{PARITY}(F_{\eta(i,j,X)}(f_r(X) \dot{-} 1, 1)) \wedge \mathit{isMatrix}_2(F_{\eta(i,j,X)}(f_r(X) \dot{-} 1, 1))\Big\} \\ &\oplus(\neg\,\text{PARITY}(F_{\eta(i,j,X)}(f_r(X) \dot{-} 1, f_c(X) \dot{-} 1)) \wedge \mathit{isMatrix}_2(F_{\eta(i,j,X)}(f_r(X) \dot{-} 1, f_c(X) \dot{-} 1)))\Big)\Bigg) \end{aligned}$$

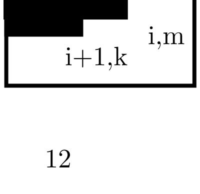
We will actually prove a stronger version: the consequent holds of all strings (even those that don't satisfy the antecedent). The proof will use lemma 2 from A33$^\sigma$ above: if two strings differ on a single bit, then they have opposite parities.

As above, for strings X not encoding matrices, the theorem is trivially true.

Recall that the intention of this theorem is to break a matrix into four parts:

e	R
S	M

We proceed by a double induction, first on rows, then on columns. It will be convenient to be able to denote a particular submatrix of $n \times m$ matrix X, the first $i \times m$ submatrix with an additional $k \leq m$ bits of the $i + 1$st row.



Call this string function Partial(i, k, X):

$$\begin{aligned}\text{Partial}(i,k,X)(b) \leftrightarrow\ & \mathit{isMatrix}_2(X) \wedge \mathit{Pair}(b) \wedge k \leq f_c(X) \wedge \\ & \big(b = \langle 0, \langle i+1, f_c(X)\rangle\rangle \vee && (11)\\ & [0 < \text{left}(b) \leq i \wedge \text{right}(b) \leq f_c(X) \wedge X(b)] \vee && (12)\\ & [\text{left}(b) = i+1 \wedge \text{right}(b) \leq k \wedge X(b)]\big) && (13)\end{aligned}$$

The case on line (11) is true for the bit encoding the size of the matrix; (12) is the case for bits within the $i \times m$ submatrix of X; and (13) is the case for bits in the final row.

As a base case, consider the first row of X. We proceed inductively by columns. For the first 1×1 submatrix of X, Partial$(0, 1, X)$, the theorem is easily true: the R, S, and M portions of the matrix have at least one dimension 0, so they are empty. Inducting on columns, for the first row of X (a $1 \times m$ matrix Partial$(0, m, X)$), the theorem is true by use of lemma 2. (This is similar to axiom A31.)

For the inductive step on rows, we consider adding one row. Assume that the axiom holds for an $i \times m$ submatrix of the first i rows of X. From the proof for A32 above, we can denote this Abbrev(i, X); alternatively, this is Partial$(i, 0, X)$. We now consider adding the bits of the $i + 1^{\text{st}}$ row, one at a time. For each bit, Partial(i, k, X) differs from Partial$(i, k + 1, X)$ on at most one bit, and lemma 2 can be applied. If the bit is 0, then the parity remains unchanged. If the bit is 1, then by the lemma above, the parity is reversed.

A34. $\text{r}(A) = 0 \vee \text{c}(A) = 0 \rightarrow \sum(A) = 0$

A34$^\sigma$. $f_r(X) = 0 \wedge f_c(X) = 0 \supset \big(\neg\,\text{PARITY}(X) \wedge \mathit{isMatrix}_2(X) \leftrightarrow \bot\big)$

There are two cases: if $\neg\,\mathit{isMatrix}_2(X)$, then the theorem is trivially true. If $\mathit{isMatrix}_2(X)$ and $f_r(X) = f_c(X) = 0$, then the only true bit of X is $X(0, 0)$, the bit encoding the size of the matrix. Thus $\neg\,\text{PARITY}(X)$ is false, and the theorem is true.

A35. $\rightarrow \text{P}(0, A) = I$

A35$^\sigma$. $F_p(0, X) = \mathit{Wrap}(f_r(X), f_r(X), \mathit{ID}(f_r(X)))$

As defined in [Fon09], $\mathit{ID}(n)$ is the string representing the $n \times n$ identity matrix, with entries starting at $(0, 0)$.

To prove A35$^\sigma$, we need to show that the two strings are bit-by-bit identical.

Consider two cases. If $\neg\,\mathit{isMatrix}_2(X)$, then $F_p(i, X)$ is true only on bit $\langle 0, \langle 0, 0\rangle\rangle$. Also, by definition, $f_r(X) = 0$. Thus, by definition of the *Wrap* function, the right-hand side is true on bit $\langle 0, \langle 0, 0\rangle\rangle$. The right-hand side is also true on all bits (shifted) of the string $\mathit{ID}(f_r(X))$, but since $f_r(X) = 0$, there are no such bits.[5]

In the other case, let $\mathit{isMatrix}_2(X)$ hold. Then both sides are expressions of the form $\mathit{Wrap}(r, c, Z)$. On both sides, $r = c = f_r(X)$ and $Z = \mathit{ID}(f_r(X))$. Thus the strings are equal.

Remark 3 By the conventions from [SC04, SK01], the zeroth power of an $n \times m$ matrix is the $n \times n$ identity matrix. This is designed to work with the left-recursive definition of matrix powering (axiom A36), so that the i^{th} power of an $m \times n$ matrix is an $m \times n$ matrix (for every $i > 0$). (See remark 1 on page 2.) Notice that this axiom is meaningful in our interpretation as F_p treats any given matrix as a square matrix along its largest dimension. This allows for powering non-square matrices.

A36. $\rightarrow \text{P}(n + 1, A) = \text{P}(n, A) \times A$

[5]The string function $\mathit{ID}(n)$ is bit-defined in [Fon09]. It yields a string representing the $n \times n$ identity matrix, with entries starting at $(0, 0)$:

$$\mathit{ID}(n)(b) \leftrightarrow \text{left}(b) < n \wedge \mathit{Pair}(b) \wedge \text{left}(b) = \text{right}(b)$$

This concise axiom features convenient notational shorthand from LAp. In order to interpret A36, it is necessary to unravel this notation back to the basic predicates of LAp. This interpretation is not difficult, but lengthy, due to several layers of dependent notation definitions in [SC04].

This axiom explicitly uses matrix multiplication, defined:

$$A \times B = \lambda_{ij}\langle \mathrm{r}(A), \mathrm{c}(B), \lambda_{kl}\langle \mathrm{c}(a), 1, \mathrm{e}(A,i,k)\rangle \cdot \lambda_{kl}\langle \mathrm{r}(B), 1, \mathrm{e}(B,k,j)\rangle\rangle$$

Matrix multiplication uses the dot product, defined:

$$A \cdot B = \Sigma\lambda_{ij}\langle \max(\mathrm{r}(A), \mathrm{r}(B)), \max(\mathrm{c}(A), \mathrm{c}(B)), \mathrm{e}(A,i,j) * \mathrm{e}(B,i,j)\rangle$$

The dot product uses the index (integer) function max, defined:

$$\max(i,j) = cond(i \leq j, j, i)$$

Ultimately, the statement of A36$^\sigma$ and its proof in $V\oplus L$ are detailed but not interesting. Each term in LAp (matrix product, dot product, and maximum) corresponds to a term in $V\oplus L$ encoding the same information. The insight of the proof is limited to observing that the iterative process used to compute Pow_2 is the same as the interpretation of Solty's definition of matrix multiplication.

4 A summary of $V\#L$

The theory LAp can be considered over an arbitrary field. In fact, most results of [SK01, SC04] hold over an integral domain; multiplicative inverses are not required.

Like $V\oplus L$, the theory $V\#L$ is an extension of V^0. It has two sorts, numbers and binary strings. $V\#L$ differs from $V\oplus L$ in a single way: it is a theory for the integers (as opposed to $V\oplus L$'s field $\{0,1\}$). Each integer is encoded as a binary string. The first bit indicates the integer's sign (zero for integers ≥ 0, one for negative integers), and the following bits form a binary representation of the integer, from least to most significant bit. Along with this integer encoding, there are useful functions defined in [Fon09]. The function *intsize* returns the absolute value of the integer encoded in string X:

$$intsize(X) = \sum_i X(i+1) \cdot 2^i$$

Thus binary string X encodes the integer $(-1)^{X(0)} \cdot intsize(X)$. We also have the string functions $+_{\mathbb{Z}}$ for integer addition, $\times_{\mathbb{Z}}$ for integer multiplication, and the function $Sum_{\mathbb{Z}}(r, \ell, Z)$ which sums a list of r integers, each of length $\leq \ell$, given as rows of string Z.[6]

The theory $V\#L$ has the same axioms as $V\oplus L$, but its axiom for matrix powering accomodates this change in matrix encoding.

5 Interpreting LAp over $\mathbb{Z}$ into $V\#L$

As above, the superscript $^\sigma$ will be used here to denote interpretation.

The theory LAp over the integers has a fairly straighforward interpretation into $V\#L$, much of it identical or similar to the above interpretation into $V\oplus L$ (section 3). For example, the index sort in LAp can be interpreted, as before, into $V\#L$'s number sort. The "field"[7] sort $\mathbb{Z}$ is interpreted into binary strings, and the matrix sort is *also* interpreted into binary strings.

Because the integers are not a field, but an integral domain, there is no interpretation for multiplicative inverses over the "field" type. See section 5.2.1 for discussion of this issue.

[6]The function *Sum* is defined in [CN10] for binary strings encoding natural numbers, not integers. It can easily be modified to work for the integer encoding presented here; that modification is achieved in [Fon09] by the addition of a helper function, and need not be repeated here.

[7]The integers are, of course, not a field, but a principal ideal domain. See section 5.2.1.

5.1 Interpreting terms and formulas

Terms, formulas, and the index sort are treated exactly as above (sections 3.1 and 3.1.1).

5.1.1 "Field" sort

Each integer in LAP is interpreted into a binary string representing it. Integer variables are interpreted into string variables, and integer terms with k free variables are interpreted as string terms with k free variables. Free variables of field terms are suppressed below for readability.

For t and u integer terms in LAP, we interpret:

LAP	$V\#L$	
0_{field}	the empty string $\varnothing$	
1_{field}	the string "10",	i.e., X such that $X(i) \leftrightarrow i = 1$
a	X	see above: variables map to string variables
$t +_{\text{field}} u$	$t^\sigma +_{\mathbb{Z}} u^\sigma$	
$t -_{\text{field}} u$	$t^\sigma +_{\mathbb{Z}} (u^\sigma)'$	where $(u^\sigma)'$ is identical to u^σ, except on the first bit
$t *_{\text{field}} u$	$t^\sigma \times_{\mathbb{Z}} u^\sigma$	
$\text{cond}_{\text{field}}(\alpha, t, u)$	$F^{\text{cond}}_{\alpha^\sigma}(t^\sigma, u^\sigma)$	where for each formula φ, define: $F^{\text{cond}}_{\varphi}(X,Y) = Z \leftrightarrow (\varphi \wedge X = Z) \vee (\neg\varphi \wedge Y = Z)$
$t =_{\text{field}} u$	$t^\sigma = u^\sigma$	

No interpretation is provided for multiplicative inverses t^{-1}. See section 5.2.1.

5.1.2 Matrix sort

This treatment is similar to $V\oplus L$, but more complicated due to the difference in encoding matrices as strings.

The matrix sort of LAP has integer entries. Every matrix in LAP has three attributes: number of rows, number of columns, and matrix entries. The matrix sort is interpreted as strings in $V\#L$; as before, the string sort in $V\oplus L$ is used to encode many different formats of data. Here, the intention is to use a binary string to encode a triple: two numbers (the dimensions of the matrix) and a matrix of *other binary strings*. The pairing function will facilitate this encoding, by means of the same usage and encoding format as is used in [CN10, Fon09].

An $a \times b$ matrix A of integers is interpreted as a string A^σ such that $A^\sigma(0, \langle a, b\rangle)$ is true. Each integer entry A_{ij} is interpreted as a string A^σ_{ij}; each row of the matrix is encoded as a string (a list of other strings), and the list of rows of the matrix is encoded in the string A^σ. (Notice that, as before, the entries begin numbering at $(1,1)$, so that the first column of A^σ is empty.) All other bits of A^σ are false.

This interpretation preserves the feature from LAP that matrices, when queried out-of-bounds, return 0. (As in LAP, a matrix queried on its 0th row must return 0; that row is reserved in our encoding, so this is a special case.)

Throughout this section, it is convenient to have a formula $isMatrix_{\mathbb{Z}}(X)$ which is true if and only if X is a valid encoding of a matrix of integers. Let $isMatrix_{\mathbb{Z}}(X)$ be:

$$\begin{aligned}\exists x, y < |X| \quad \Big[\ & \overbrace{X(0,\langle x,y\rangle)}^{\text{(a)}} \wedge \overbrace{\forall z < |X|\big(\neg X(z,0) \wedge (z = \langle x,y\rangle \vee \neg X(0,z)) \wedge (X(z) \to Pair(z))\big)}^{\text{(b)}} \\ & \underbrace{\wedge \forall z \le x \forall i < |X^{[z]}|\big(X^{[z]}(i) \to Pair(i) \wedge \text{left}(i) \le y\big)}_{\text{(c)}}\Big]\end{aligned} \tag{14}$$

This formula has three sections: section (a) guarantees that the matrix dimensions are encoded properly; section (b) checks that there are no matrix entries in the 0th row or column, or in any non-pair-numbered bit; section (c) checks that the matrix entries are encoded properly and within the bounds of the matrix dimensions. Matrices have many "wasted" bits of empty space, due to the use of pairing numbers to encode rows and columns of data.

The interpretation of LAp matrix functions into $V\#L$ is similar, and in some places, identical, to the above interpretation (section 3.1.3). Only the differences are noted in this section.

LAp	$V\#L$
$\sum(T)$	$F_{\sum}(T^\sigma)$

In order to interpret $\sum(T)$, we require a string-valued function that, given an input string X validly encoding a matrix of integers, outputs the string Y encoding the integer sum of the entries of the matrix. The function *Sum* is defined in [CN10] to sum a list of (positive) numbers encoded in binary notation. In [Fon09], this is extended to the function $Sum_{\mathbb{Z}}$, which sums a list of integers. (These integers are encoded in binary strings exactly as here.) Thus all that is required is a string function to convert an integer matrix into a list of integers:

$$List(Z)^{[k]}(b) \leftrightarrow isMatrix_{\mathbb{Z}}(Z) \wedge 0 < r \leq f_r(Z) \wedge 0 < c \leq f_c(Z) \wedge Z(\operatorname{div}(k,c), \operatorname{rem}(k,c))(b)$$

Thus $List(Z)$ is a list of the integer entries in the matrix Z, in "reading order": the first c elements of the list are the elements from the first row of matrix Z, the next c elements are from the second row, and so on. (As is standard, $List(Z)$ is zero on all unspecified bits.)

Define

$$F_{\sum}(T^\sigma)(b) \leftrightarrow Sum_{\mathbb{Z}}(f_r(T^\sigma) \times f_c(T^\sigma), |T^\sigma|, List(T^\sigma))(b) \wedge \underbrace{isMatrix_{\mathbb{Z}}(T^\sigma)}_{\text{true only for strings encoding matrices}}$$

We specify $F_{\sum}$ in this way so that universal claims about $\sum(T)$ in LAp can be interpreted into universal claims in $V\#L$ of the same provability, even though not every string in $V\#L$ encodes a matrix.

LAp	$V\#L$
$\lambda_{ij}\langle m, n, t\rangle$	$F_{t^\sigma}(m^\sigma, n^\sigma)$

Constructed λ_{ij} matrix terms have the restriction that i and j are free in t and not free in m and n. In LAp, the λ operator binds i and j. Thus, when interpreted, t^σ has two reserved free variables i^σ and j^σ (the interpretations of i and j). The restrictions are preserved: i^σ and j^σ are free in t^σ and not free in m^σ and n^σ. Thus, suppressing free variables, $F_{t^\sigma}(m^\sigma, n^\sigma)$ is the string interpreting the matrix term $\lambda_{ij}(m, n, t)$ in $V\#L$.

If $t(i, j, \vec{v}_{\text{index}}, \vec{V}_{\text{field}}, \vec{V}_{\text{matrix}})$ is a field term in LAp, then t^σ is a string in $V\#L$. Let $t^\sigma(k)$ be the formula $\varphi(i, j, \vec{v}, \vec{V})(k)$ with the same number of free variables. (Recall that field variables are interpreted into string variables.) For every formula $\varphi(i, j, \vec{v}, \vec{V})(k)$, the string function $F_\varphi(m, n, \vec{v}, \vec{V})(b)$ is bit-defined:

$$b = \langle 0, \langle m, n\rangle\rangle \vee \exists i \leq m \exists j \leq n (i > 0 \wedge j > 0 \wedge \operatorname{left}(b) = \langle i, j\rangle \wedge \varphi(i, j, \vec{v}, \vec{V})(\operatorname{right}(b))) \tag{15}$$

Thus the output of $F_\varphi(m, n)$ is a string encoding a matrix with m rows and n columns, with entries strings whose bits are determined by φ. This aligns with LAp's definition of λ_{ij} term, so that $F_{t^\sigma}(m^\sigma, n^\sigma)$ is the string interpretation in $V\#L$ of the matrix $\lambda_{ij}(m, n, t)$ in LAp.

Notice that $isMatrix_{\mathbb{Z}}(F_\varphi(m, n))$ is true for all for all formulas φ with at least two free number variables. This is by construction, as the bit-definition of F_φ matches the clauses of $isMatrix_{\mathbb{Z}}$.

The interpretation of λ_{ij} terms is required when those terms are used inside formulas; the existence of string $F_{t^\sigma}(m^\sigma, n^\sigma)$ is guaranteed by Σ_0^B-COMP.

LAp	$V\#L$
$\mathrm{e}(T, m, n)$	$F_e(T^\sigma, m^\sigma, n^\sigma)$

The e function returns the matrix's entry at the requested coordinates; thus its interpretation is *string*-valued in $V\#L$. We bit-define $F_e(X,i,j)$:

$$F_e(X,i,j)(b) \leftrightarrow X(i,j)(b) \wedge i > 0 \wedge j > 0 \wedge \underbrace{isMatrix_{\mathbb{Z}}(X)}_{\text{true only for valid strings encoding matrices}}$$

LAP specifies that matrices return 0 when queried out-of bounds. As in LAP, matrix row and column numbering starts at 1 for strings encoding matrices. Hence querying at $(0,x)$ or $(x,0)$ is out-of-bounds and returns the string of all false bits, representing integer 0. Conveniently, on all bits in a row or column larger than its dimensions, a matrix encoded as a string also returns false. Strings which do not encode matrices are interpreted as 0×0 matrices, and accordingly return false for every lookup.

This interpretation of $e(T,m,n)$ is convenient; it allows us to use the same interpretation of matrix equality as above, since again matrices that have the same number of rows, columns, and entries (strings) will be equal.

The interpretation of powering $\mathrm{P}(i,A)$ into $V\#L$ is identical to the interpretation into $V\oplus L$ above, except that $Pow_{\mathbb{Z}}$ replaces Pow_2.

5.2 Provability is preserved

This interpretation is useful because it preserves provability: for any sequent $\rightarrow \varphi$ provable in LAP without using field inverses, the interpreted formula φ^t is provable in $V\#L$. See the note on inverses (section 5.2.1).

As above in section 3.2, we will have a lemma to simplify upcoming proofs. This lemma states that if two integers differ by 1, then their sums also differ by 1.

Lemma 4 $V\#L \vdash \left(X^{[i]} = Y^{[i]} +_{\mathbb{Z}} Z \wedge \forall j \neq i\ X^{[j]} = Y^{[j]}\right) \rightarrow Sum_{\mathbb{Z}}(1,|X|,X) = Sum_{\mathbb{Z}}(1,|Y|,Y) +_{\mathbb{Z}} Z$

This lemma follows directly from definitions.

$V\#L$ has axioms:

B1. $x+1 \neq 0$	B7. $(x \leq y \wedge y \leq x) \supset x = y$
B2. $x+1 = y+1 \supset x = y$	B8. $x \leq x+y$
B3. $x+0 = x$	B9. $0 \leq x$
B4. $x+(y+1) = (x+y)+1$	B10. $x \leq y \vee y \leq x$
B5. $x \cdot 0 = 0$	B11. $x \leq y \leftrightarrow x < y+1$
B6. $x \cdot (y+1) = (x \cdot y)+x$	B12. $x \neq 0 \supset \exists y \leq x(y+1 = x)$
L1. $X(y) \supset y < \|X\|$	L2. $y+1 = \|X\| \supset X(y)$

SE. $[|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i)) \supset X = Y$
The axiom scheme Σ_0^B-COMP: for every Σ_0^B-COMP formula $\varphi(z)$,
$$\exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z))$$
And the axiom for matrix powering.

In this section, each interpretation of an axiom from LAP is shown to be provable in $V\#L$. Since $V\#L$'s set of axioms is nearly identical to those of $V\oplus L$ (the exception is that matrix powering is here over integer matrices, not binary matrices), many of these proofs are identical to those in section 3.2 above. The interpretation into $V\#L$ differs only on interpretations of strings, as well as functions $\sum$, λ_{ij}, and e. It suffices to re-prove only those axioms of LAP whose interpretations have changed. (This is a superset of all axioms involving PARITY in their proofs; note that the theory $V\#L$ does not have PARITY.)

The equality and field axioms remain the same, but the field axioms must be reproven.

The field axioms of LAP are A18-A27. For readability, let U be the all-zero string, and V be the string such that $V(i) \leftrightarrow i = 1$.

A18. $\rightarrow 0 \neq 1 \wedge a+0 = a$ A18$^\sigma$. $U \neq V \wedge X +_{\mathbb{Z}} U = X$

This is provable since string equality is true equality in $V\#L$, and by the definition of $+_{\mathbb{Z}}$.

A19. $\rightarrow a + (-a) = 0$ A19$^\sigma$. $X +_\mathbb{Z} Y = U$

Where Y is a string that differs from X only on $Y(0)$. Again, this is provable from the definition of $+_\mathbb{Z}$.

A20. $\rightarrow 1 * a = a$ A20$^\sigma$. $V \times_\mathbb{Z} X = X$

This is provable by the definition of $\times_\mathbb{Z}$.

A21. $a \neq 0 \rightarrow a * (a^{-1}) = 1$

This axiom is not provable in the interpretation as described above, since no interpretation was provided for multiplicative inverses in the "field" $\mathbb{Z}$. See section 5.2.1 for further discussion of inverses.

A22. $\rightarrow a + b = b + a$ A22$^\sigma$. $X +_\mathbb{Z} Y = Y +_\mathbb{Z} X$

This is provable from the definition of $+_\mathbb{Z}$, since it is commutative.

A23. $\rightarrow a * b = b * a$ A23$^\sigma$. $X \times_\mathbb{Z} Y = Y \times_\mathbb{Z} X$

This is provable from the definition of $\times_\mathbb{Z}$.

A24. $\rightarrow a + (b + c) = a + (b + c)$ A24$^\sigma$. $X +_\mathbb{Z} (Y +_\mathbb{Z} Z) = (X +_\mathbb{Z} Y) +_\mathbb{Z} Z$

This is provable from the definition of $+_\mathbb{Z}$.

A25. $\rightarrow a * (b * c) = (a * b) * c$ A25$^\sigma$. $X \times_\mathbb{Z} (Y \times_\mathbb{Z} Z) = (X \times_\mathbb{Z} Y) \times_\mathbb{Z} Z$

This proof is detailed but uninteresting, and simply follows the definition of $\times_\mathbb{Z}$.

A26. $\rightarrow a * (b + c) = a * b + a * c$ A26$^\sigma$. $X \times_\mathbb{Z} (Y +_\mathbb{Z} Z) = X \times_\mathbb{Z} Y +_\mathbb{Z} X \times_\mathbb{Z} Z$

This proof is also intricately detailed, but simply traces the interacting definitions of $+_\mathbb{Z}$ and $\times_\mathbb{Z}$.

A27. $\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = a$ A27$^\sigma$. $\alpha^\sigma \supset F^{\mathrm{cond}}_{\alpha^\sigma}(X, Y) = X$

and $\neg\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = b$ and $\neg\alpha^\sigma \supset F^{\mathrm{cond}}_{\alpha^\sigma}(X, Y) = Y$

These follow directly from the definition of $F^{\mathrm{cond}}_{\alpha^\sigma}$.

The matrix axioms of LAp are A28-A34; these are the axioms most different in interpreted form in $V \oplus L$ and $V\#L$. As above, let U be the all-zero string, and V be the string such that $V(i) \leftrightarrow i = 1$.

A28. $(i = 0 \vee \mathrm{r}(A) < i \vee j = 0 \vee \mathrm{c}(A) < j) \rightarrow \mathrm{e}(A, i, j) = 0$

A28$^\sigma$. $(x = 0 \vee f_r(X) < x \vee y = 0 \vee f_c(X) < y) \supset F_e(X, x, y) = U$

A29. $\rightarrow \mathrm{r}(\lambda_{ij}\langle m, n, t\rangle) = m$ A29$^\sigma$. $f_r(F_{\varphi(x,y,\vec{v},\vec{V})}(x, y, \varphi(x, y, \vec{v}, \vec{V}))) = x$

and $\rightarrow \mathrm{c}(\lambda_{ij}\langle m, n, t\rangle) = n$ and $f_c(F_{\varphi(x,y,\vec{v},\vec{V})}(x, y, \varphi(x, y, \vec{v}, \vec{V}))) = y$

and $1 \leq i, i \leq m, 1 \leq j, j \leq n \rightarrow \mathrm{e}(\lambda_{ij}\langle m, n, t\rangle, i, j) = t$

and $(1 \leq i \wedge i \leq m \wedge 1 \leq j \wedge j \leq n) \supset \big((F_{\varphi(i,j,\vec{v},\vec{V})}(m, n))(i, j) = \varphi(i, j, \vec{v}, \vec{V})\big)$

for any appropriate formula $\varphi(x, y, \vec{v}, \vec{V})$

Axioms A28 and A29 are provable in $V\#L$ by definition of the functions f_r, f_c, and F_φ.

A30. $\mathrm{r}(A) = 1, \mathrm{c}(A) = 1 \rightarrow \sum(A) = \mathrm{e}(A, 1, 1)$

A30$^\sigma$. $f_r(X) = 1 \wedge f_c(X) = 1 \supset F_{\sum}(X) = F_e(X, 1, 1)$

This is provable from definitions, though the details are messy: by $f_r(X) = 1$ and $f_c(X) = 1$, X properly encodes a matrix (i.e., $\mathit{isMatrix}_\mathbb{Z}(X)$), and the only true bits of X are: $X(0, \langle 1, 1\rangle)$, and possibly $X(1, \langle 1, b\rangle)$. Those are exactly the bits in the string returned by the function $F_e(X, 1, 1)$.

A31. $\mathrm{r}(A) = 1, 1 < \mathrm{c}(A) \to \sum(A) = \sum(\lambda_{ij}\langle 1, \mathrm{c}(A) - 1, A_{ij}\rangle) + A_{1\,\mathrm{c}(A)}$

A31$^\sigma$. $f_r(X) = 1 \wedge 1 < f_c(X) \supset F_{\sum}(X) = F_{\sum}(F_{F_e(X,i,j)}(1, f_c(X))) +_{\mathbb{Z}} F_e(X, 1, f_c(X))$

This is provable by induction on columns, since $Sum_{\mathbb{Z}}$ is defined using $+_{\mathbb{Z}}$ [Fon09].

A32. $\mathrm{c}(A) = 1 \to \sum(A) = \sum(A^t)$

A32$^\sigma$. $f_c(X) = 1 \supset \big(F_{\sum}(X) = F_{\sum}(Y)\big)$

where Y is the string: $F_{F_e(X,j,i)}(X, f_c(X), f_r(X))$

This can be proven in the same way as previously, in the mod2 case.

A33. $1 < \mathrm{r}(A), 1 < \mathrm{c}(A) \to \sum(A) = \mathrm{e}(A, 1, 1) + \sum(R(A)) + \sum(S(A)) + \sum(M(A))$ Recall that, by definition:
$R(A) \colon = \lambda_{ij}\langle 1, \mathrm{c}(A) - 1, \mathrm{e}(A, 1, i + 1)\rangle$
$S(A) \colon = \lambda_{ij}\langle \mathrm{r}(A) - 1, 1, \mathrm{e}(A, i + 1, 1)\rangle$
$M(A) \colon = \lambda_{ij}\langle \mathrm{r}(A) - 1, \mathrm{c}(A) - 1, \mathrm{e}(A, i + 1, j + 1)\rangle$

These expressions nesting e inside λ make the translation of A33 very long.

Thus A33$^\sigma$ is:

$$\begin{aligned} 1 < f_r(X) \wedge 1 <, f_c(X) \to F_{\sum}(X) = \quad & F_e(X, 1, 1) \\ & +_{\mathbb{Z}} F_{\sum}(F_{F_e(X,i,j)}(1, f_c(X) - 1, F_e(X, 1, i + 1))) \\ & +_{\mathbb{Z}} F_{\sum}(F_{F_e(X,i,j)}(f_r(X) - 1, 1, F_e(X, i + 1, 1))) \\ & +_{\mathbb{Z}} F_{\sum}(F_{F_e(X,i,j)}(f_r(X) - 1, f_c(X) - 1, F_e(X, i + 1, j + 1))) \end{aligned}$$

A34. $\mathrm{r}(A) = 0 \vee \mathrm{c}(A) = 0 \to \sum(A) = 0$

A34$^\sigma$. $f_r(X) = 0 \wedge f_c(X) = 0 \supset \forall i \le |F_{\sum}(X)|\ \neg F_{\sum}(X)(i)$

A35. $\to \mathrm{P}(0, A) = I$ A35$^\sigma$. $F_p(0, X) = \mathit{Wrap}(f_r(X), f_r(X), ID_{\mathbb{Z}}(f_r(X)))$

The string function $ID_{\mathbb{Z}}(n)$ returns the $n \times n$ identity matrix of integers, each encoded by a string. It is defined in [Fon09]. Here again the difference in matrix encodings is evident, as standard matrices in $V \oplus L$ are encoded separately from their size, but interpreted matrices from LAP require the *Wrap* function to mark their dimensions. Note that non-square matrices can be powered; an $n \times m$ matrix raised to the 0th power will yield a $\max(n, m) \times \max(n, m)$ identity matrix (recall remark 1).

A36. $\to \mathrm{P}(n + 1, A) = \mathrm{P}(n, A) \times A$

As before, the statement of A36$^\sigma$ is lengthy due to the notational shorthand $\times$ in LAP, which condenses a much longer formula involving nested λ_{ij} and $\sum$ constructions.

The proofs of axioms A32$^\sigma$, A33$^\sigma$, A34$^\sigma$, A35$^\sigma$, and A36$^\sigma$ are similar to their proofs when interpreted into $V \oplus L$; note that we will use lemma 4 (for integers) in place of lemma 2 (for $\mathbb{F}_2$).

5.2.1 A note on inverses

The integral domain $\mathbb{Z}$ does not have multiplicative inverses, so no interpretation has been provided above for LAP terms of the form a^{-1}. Thus this interpretation is incomplete. It remains a useful tool, and the main theorem – that results provable in LAP are also provable in $V\#L$ – is still attainable. As noted in [SK01, SC04], multiplicative inverses are only used to prove two results in LAP.

These two results concern the "hard matrix identities":

$$AB = I,\, AC = I,\ \to B = C \tag{I}$$
$$AB = I \to AC \neq 0,\, C = 0 \tag{II}$$
$$AB = I \to BA = I \tag{III}$$
$$AB = I \to A^t B^t = I \tag{IV}$$

If A, B are $n \times n$ and the last column of A is 0, then $AB \neq I$. (V)

Theorem 3.1 of [SC04] proves that LA (without matrix powering or multiplicative inverses) proves $I \Leftrightarrow II \Leftrightarrow III \Leftrightarrow IV$.

Theorem 5 (4.1 of [SC04]) *LAP (over any field) proves that the Cayley-Hamilton theorem implies the hard matrix identities I-IV.*

This result is provable over an integral domain.

Proof of theorem 4.1: This $V\#L$ proof is only a slight modification of the original LAP proof [SC04].

Since theorem 3.1 [SC04] shows that the hard matrix identities I-IV are equivalent (without using inverses), it suffices to consider the identity III:

$$AB = I \rightarrow BA = I$$

Using the C-H theorem, $p(A) = 0$ where p is the characteristic polynomial of A. Since p has leading coefficient 1, it is not the zero polynomial. There must be $k \geq 0$ and a polynomial q with non-zero constant term q_0 such that

$$0 = p(A) = q(A)A^k$$

From $AB = I$, using induction, $A^kB^k = I$. Thus

$$0 = 0B^k = q(A)A^kB^k = q(A)I = q(A)$$

Separating the constant and non-constant parts of q,

$$\hat{q}(A)A = -q_0I$$

Multiplying on the right by B and using the assumption that $AB = I$,

$$\hat{q}(A) = \hat{q}(A)I = \hat{q}(A)AB = -q_0B$$

Thus $-q_0BA = -q_0I$; by the properties of a module (without using multiplicative inverses), $q_0(I - BA) = 0$. In an integral domain, there are no zero divisors; either q_0 or $I - BA$ must be zero. But $q_0 \neq 0$, so it must be that $BA = I$. ■

The second result from [SK01, SC04] states that:

Lemma 6 (3.1 of [SC04]) *LA proves the equivalence of* (V) *and* $(I) - (IV)$.

The proof that III implies V is straightforward and does not require inverses.

In the other direction, the proof in [SK01, SC04] uses multiplicative inverses in order to cancel terms, and prove that V implies II. Although the inverse axiom cannot be interpreted, we can add a cancellation axiom. (A cancellation law is clearly implied by the inverse axiom.) By extending $V\#L$ with a cancellation axiom, we can obtain the same result in a slightly larger theory.

(Steve suggested that there should be an easy way to prove this without using multiplicative inverses, since it seems obviously true in an integral domain. After some examination, the proof without inverses is not apparent. Email follow-up with Michael ensues. Dec 8 2009)

6 Conclusion

Two mappings are provided above for interpreting Soltys' theory LAP over $\mathbb{F}_2$ and $\mathbb{Z}$ into the Cook-Nguyen-style theories $V\oplus L$ and $V\#L$, respectively. These interpretations are useful tools, as they preserve provability; anything provable in LAP is provable (in translation) in $V\oplus L$ and $V\#L$. This increases the ease of working in $V\oplus L$ and $V\#L$, since LAP has more convenient and natural mathematical notation; thus $V\oplus L$ and $V\#L$ are more useful theories. Additionally, these interpretations fit LAP into the hierarchy of theories established in [CN10].

References

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, 2010. Draft available from URL `http://www.cs.toronto.edu/~sacook`.

[Fon09] Lila Fontes. Formal Theories for Logspace Counting. Master's thesis, University of Toronto, 2009.

[SC04] Michael Soltys and S. A. Cook. The Proof Complexity of Linear Algebra. *Annals of Pure and Applied Logic*, 130:277–323, 2004.

[SK01] Michael Soltys-Kulinicz. *The Complexity of Derivations of Matrix Identities.* PhD thesis, University of Toronto, 2001.