Presented to the Institute of Actuaries Students' Society

on 24th March 1972

SOME PROPOSALS FOR A REVISION OF THE INTERNATIONAL ACTUARIAL NOTATION

by

P. J. Turvey

Some Proposals for a Revision of the International Actuarial Notation

by P.J. Turvey

To be

Presented to the Institute of Actuaries Students' Society

at Staple Inn Hall, High Holborn, W.C.1. on

Friday, 24th March 1972

at 5.30 p.m.

Contents

1. Introduction

2. Existing Notations

3. Practical Use of the Notation

4. Designing a Notation

   4.1. Actuarial Considerations

   4.2. Computer Considerations

   4.3. Practical Conclusions

5. The Proposed Notation

   5.1. The Function Name

   5.2. The Parameter List

   5.3. Global Values

   5.4. Some Examples

   5.5. Further Simplification

6. Extensions and Modification

7. Comparison with other systems

8. Some thoughts on implementation

Tables

1. Function Identification Codes

2. Indicator Codes

3. Comparison with existing notation

4. Comparison of three proposed notations

1. Introduction

Attention has recently been given to the problem of defining a new version of the International Actuarial Notation. The topic is also to be discussed at the Congress in Oslo.

The aims of this revision are twofold :

1) To simplify typesetting of texts involving actuarial functions.

2) To define a notation which can be used to communicate with electronic computers.

The relevance of the first reason is arguable - for example, the existence of a large volume of material containing the old notation would mean that a new notation could never entirely replace the old. However, anyone who has ever had to correct a typewritten document (such as this one) will be well familiar with the problems, and the author would like to record his appreciation of the help and understanding that he received from his typists!

The second reason is far more important, as the existing notation with its "halo of subscripts and superscripts" is not compatible with modern computers.

The aim of this paper is therefore to describe a notation which is suitable for the definition of actuarial terms and formulae in a way compatible with the requirements of a computer. With the increasing use of computers by the profession, the absence of a formal and generally accepted notation is felt to be a handicap to the further developments of their use. Many attempts (mostly unpublished) have been made to establish such a notation; the resulting systems have much in common. An attempt has been made to take the best features of each system and incorporate them into a general solution.

This paper does not attempt to fully discuss the case for a revised notation; the arguments have already been eloquently expressed (Ref. 1 and 2) and the body of existing work on this subject demonstrates that the need is widely recognised. Rather, it is intended that the paper be confined to the problem of designing the notation; nevertheless, the occasional red herring is pursued.

2. Existing Notations

Several notational systems have been published which attempt to solve these problems. All of them fail to satisfy certain of the criteria proposed below, but they have been a valuable source of inspiration in the design of the notation described in this paper. The author gratefully acknowledges that most of the ideas described are not new.

Two recent sets of proposals which have used the same general approach to the problem are the work of Boehm's committee - which presented a paper to the 18th Congress (and which will be presenting a further paper in Oslo), and a report by an Australian committee. The important common ground of these proposals is that they attempt to describe a consistent set of rules for rewriting functions such as $A_{x\,:\,\overline{n}|}$ by expressions such as B(x,n) (Australia) and AE(x,n) (Boehm). This general philosophy is also followed in this paper; the essential difference between the three sets of proposals is in the precise definitions of the way in which the functions are to be rewritten.

Section 7 compares the notation of this paper with certain other proposals.

3. Practical Use of the Notation

A notation constructed following this approach could be used in two ways :

1) In descriptive matter, which actuarial formulae could be typewritten or typeset using ordinary equipment without the need for any special techniques and/or manual amendments.

2) As the basis for a computer programming language. The expressions to be evaluated would be written onto an appropriate coding sheet, together with details of the actuarial bases and age ranges, etc. to be used. From the coding form, punched cards (or whatever) are created and submitted to the computer.

4. Designing a Notation

We start by discussing the properties that a "perfect notation" would have. In practice, it appears to be impossible for any notation to be perfect, and the problem becomes one of trying to decide which imperfections can be accepted.

The criteria can be broadly classified as "actuarial" and "computer"; the computer considerations lead to a set of practical conclusions about the form that the notation must take.

4.1. Actuarial Considerations

A(1) The notation should cover all functions included in the existing International Actuarial Notation.

A(2) The notation should be capable of extension to cover other functions as and when the need arises.

A(3) It should follow the existing notation as closely as possible in order to minimise changeover difficulties.

A(4) It should be as simple as possible and not involve the writing of redundant information.

A(5) It should be possible to deduce the representation of a given function by following a set of consistent rules.

A(6) The notation should be unambiguous and easy to read.

4.2. Computer Considerations

C(1) It should be possible to implement the notation on a computer easily and cheaply.

C(2) Use of the implementation should not require extensive familiarity with computers.

C(3) The notation should be capable of being used on as wide a variety of types of computer as possible.

It is not the purpose of this paper to propose a specific computer implementation, but rather to describe a notation upon which a language could be based. Accordingly, the notation described deliberately omits to specify rules for input/output and for arithmetic. It is anticipated, however, that the notation for arithmetical expressions would probably follow that used in FORTRAN and a number of other languages - see P(2) below. It should prove a straightforward task to design these aspects with a specific computer in mind.

4.3. Practical Conclusions

Based upon these computer considerations, we can deduce various principles which control the design of the notation.

P(1) Computer input is generally treated as a series of consecutive characters, so that the notation should be linear - that is, any single function or formula is expressed as a string of symbols on one line. The length of a "line" will be subject to some overall limit dependant upon the specific implementation.

P(2) The character set used must be a subset of the characters available on a normal typewriter, and also a subset of the characters available on any computer upon which it is to be implemented : thus characters such as $\bar{a}$ $\ddot{a}$ $\bar{A}$ cannot be included. Furthermore, it should be noted that most computers cannot differentiate between upper and lower case letters (IBM 360 and 370 are exceptions).

The character set to be used is therefore a to z, 0 to 9, space, comma, fullstop and brackets, i.e.

abcdefghijklmnopqrstuvwxyz0123456789 ,.()

In writing, either upper or lower case letters (but not both) may be used; for computer input, upper case is implied even if lower case is written. In this note, lower case will be used throughout in the interest of simplicity.

Depending upon the specific implementation, the characters + - / * and possibly ↑ may also be used to denote arithmetic combination. / denotes division; * denotes multiplication. Exponentiation is denoted by either ** or ↑.

P(3) Functions of the form abc(x,y,z) are frequently used in computing languages. In this example :

"abc" is the function name,

(denotes the beginning of the parameter list,

"x" "y" and "z" are the parameters separated by commas, and

) denotes the end of the parameter list.

The number of parameters may vary from none (in which case the brackets are omitted) up to any number.

In order to facilitate implementation, this format will be used for the general representation of individual actuarial functions. To maintain the maximum compatibility with existing languages the further restriction will be applied that the number of parameters must be fixed for each function name, and that each parameter must be provided every time the function is written.

P(4) No attempt will be made to include some of the very complex functions - such as

$$A^{x}_{\overline{yz}\,2\,u}$$

in the notation because these rarely occur in practice and the minimal advantages of their inclusion would not appear to justify the consequent complications; we are trying to design a simple, useful and economical notation.

5. The Proposed Notation

Any particular actuarial function (for example $a^{(m)}_{[x]:\overline{n}|}$), is expressed as a function name followed by a parameter list (e.g. asnm (x,n,m).

The function name consists of a string of letters and numbers which uniquely identify the function and by implication specify the parameters required. The parameter list is used to specify the values of the variables involved (the parameters).

5.1. The Function Name

The function name consists of two parts. The beginning is an identification code consisting of one or two letters (e.g. 'a' for an assurance function such as A_x, 'at' for an annuity due $\ddot{a}_x$, 'm' for the commutation function M_x). Two letters are needed whenever the first letter can have more than one "obvious" meaning. The full list is given in Table 1.

The remainder of the function name is a series of indicators which specify the parameters involved. These indicators are used to distinguish between the various forms which the function can take, and have been designed to minimise the amount of writing required. The indicator codes attempt to be meaningful, so that it is easy to understand an unfamilar function simply by referring to first principles.

The indicators (when coded) must always appear in the same order. The full list of possible indicators and their meanings are given in Table 2 - there are codes which indicate

- the number of lives involved and whether 'joint' or 'last-survivor'
- the 'selectness' of these lives
- whether a 'term' is involved, and how it is used
- the rate of interest
- the mortality basis
- the frequency of payment
- the conditions regarding payment of premiums

It should be noted that not all possible combinations of codes lead to valid functions. Conversely, certain combinations lead to functions which are actuarially meaningful but which can not be readily expressed in the existing notation ;

for example, $nn(x,n) = N_x - N_{x+n}$ (Boehm, Ref.1).

5.2. The Parameter List

The presence or absence of each indicator code determines the items for which values are to be furnished in the parameter list.

These parameters can be either variables (such as x or n), specific values (such as 50), or any mixture of these.

The parameters will always be integers, except for the rate of interest which will be expressed as a decimal, e.g. .05 for 5%. (In a particular implementation it might also be possible to use arithmetic expressions (such as x + n, x + 10) and such an extension would be desirable; however, to insist on it might unduly complicate implementation on a particular computer.)

5.3. Global Values

Where the items 'rate of interest' and/or 'mortality table' are essential to the meaning of a function, but are omitted from the representation, a predefined global value is used. The actual way in which these globals would be defined would depend upon the specific computer implementation, but one might have a statement such as :

call global (5, .04)

or alternatively :

mort = 5

int = .04

both of which define a global mortality table code of 5 and a global interest rate of 4%.

Thus, if code 5 refers to A49/52 and code 6 to a(55) males, we would write

a(x)	=	A_x	4% A49/52
ai(x,.03)	=	A_x	3% A49/52
aq(x,6)	=	A_x	4% a(55)
aqi(x,6,.03)	=	A_x	3% a(55)

Of course, aq(x,5), aqi(x,5,.04) and ai(x,.04) would all be equivalent to a(x); the simplest form would normally be used.

5.4. Some Examples

At first sight, the tables may appear to provide an unnecessarily complicated set of rules. To indicate why these rules are necessary, consider the function that might be verbally referred to as "A-twenty-twelve". Depending upon the context, this might be understood to mean any of the following

$a^{(12)}_{20}$ $\quad a_{20:\overline{12|}}$ $\quad a_{20:12}$ $\quad a_{\overline{20:12}}$

$A^{(12)}_{20}$ $\quad A^{1}_{20:\overline{12|}}$ $\quad A_{20:\frac{1}{\overline{12|}}}$ $\quad A_{20:\overline{12|}}$

$A^{(12)}_{[20]}$ $\quad$ .... and many more!

It is clear that all of these functions cannot be expressed as a(20,12) - the obvious translation - and we must try to design a scheme which permits all possible variations without ambiguity.

Let us see how variations on a simple function would be handled :

a(x)	$= A_x$
an(x,n)	$= A_{x:\overline{n|}}$
ani(x,n,i)	$= A_{x:\overline{n|}}$ at i%
anic(x,n,i)	$= \bar{A}_{x:\overline{n|}}$ at i%
a2nic(x,y,n,i)	$= \bar{A}_{xy:\overline{n|}}$ at i%
a2knic(x,y,n,i)	$= \bar{A}_{\overline{xy}:\overline{n|}}$ at i%
a2k(x,y)	$= A_{\overline{xy}}$

5.5. Further simplification of the written notation

In writing, we frequently wish to use x to denote an age and n to denote a term. Where this is the case, and x's and n's are the only parameters, we may - without ambiguity - omit the parameter list altogether.

It would be desirable to be able to use this convention in communication with the computer; in view of the probable resultant increase in the problems of implementing the language, this requirement is therefore omitted.

6. Extensions and Modifications to the Notation

The notation described should be considered as a framework for further development. It may well be found that certain aspects can be better handled, and suggestions will be welcomed.

In addition, the notation may be extended to include further actuarial expressions - such as contingent assurances or those arising in connection with Group Pensions, Sickness etc., simply by extending Tables 1 and 2. These aspects have been omitted from the current Paper in the interests of simplicity.

It should be emphasised that the author, whilst attempting to put forward a consistent set of rules for a notation, does not claim to have described an immutable system. Several of the problems could well be tackled in alternative ways, and the author is open to suggestions as to how the system might be improved.

For example (see P(4)) contingent functions involving two lives have been handled; in general, contingent functions on more than two lives have been omitted. It could be argued that, in the interests of simplicity, all contingent functions should be omitted - this would permit simplification of the function name structure. Conversely, by further complicating that structure, more complex contingent functions could be handled. Where should the line be drawn?

7. Comparison with other systems

A number of other systems have been proposed in an attempt to tackle the basic problem, and it may be helpful to describe the principal differences between the proposals in this paper and these other systems.

1) Boehm's Committee (Ref. 1)

The notation of Boehm is comprehensive and includes equivalents of nearly all of the elements of the existing notation, together with extensions thereto. As a result, the notation is more complicated. The notation has been extended to allow for the use of the concept of 'terminal age'. The principal difference between the two notations is that Boehm has a much simpler 'function name', but a much more complicated parameter list. The latter depends, inter alia, upon whether one is using variables (such as x) or discrete values (such as 20). This results in a structure which is likely to give rise to difficulties at the computer implementation stage.

One aspect of Boehm's work, not considered in the present paper , is the redefinition of the meanings of certain functions.

2) Australian Committee (Ref. 2)

The report of this committee described both a notation and an implementation. The notation was apparently designed for a computer which would accept both upper and lower case letters, and hence could not be used elsewhere in its present form. The notation has been extended in a rather neat fashion to include summation, integration etc.

The function name structure is similar to that described herein; it is simpler because variable parameter lists are permitted. The parameter list may contain 'compound parameters' such as .04(4) or .07C (p.179); this could give rise to further difficulties on implementation.

3) Columnar Systems

Various systems of columnar manipulation have been published. These systems do not need a formalised notation, as the user is responsible for generating his own names.

8. Some thoughts on Implementation

Despite having denied the intention of specifying how the notation would be implemented, the author is unable to resist the temptation of indicating how he feels this might be done.

A notation, however good, is of no use unless there is a computer which can understand it. There are severalways in which a notation can be implemented for a computer :

- as the basis of a new language
- as an addition to an existing language
- as an interpretive system (preprocessor or precompiler).

The implementation may be performed by computer manufacturers, software houses, or by individual users; ther may be several different implementations applicable to different computer configurations. The proposals have been drawn up with these considerations in mind, and it is hoped that it would be possible for a practical implementation to appear following the adoption of a notation.

As in the case of other computer languages, early implementations need not cover the whole of the notation - for example joint-life functions might be excluded - but whatever subset was available would still be extremely useful. In fact, it is extremely unlikely that the whole of the notation could be covered.

The remainder of this section may be omitted by the reader without practical experience of computer programming.

8.1. Function Subroutines

One approach which appears to have encouraging possibilities is that of providing a series of FUNCTION SUBROUTINES for use with FORTRAN programmes (or for other similar languages). Corresponding to each possible function name, we write a simple subroutine which evaluates the function in terms of commutation or other functions. This approach has been successfully used in practice.

8.2. Arrays in storage

Another approachis to establish arrays (one, two or even three-dimensional) in storage by means of a standard sub-programme. The arrays would have names corresponding to the function to be used, e.g. a, an, etc.

This approach has frequently been used; it is of limited power because of the limits imposed on the number of parameters and the size of core storage. It is nevertheless a very useful tool.

8.3. Columnar approach

This method also has been used with a considerable degree of success. Languages based on this method could be adapted to the proposed notation by establishing 'operations' corresponding to the function names which set up the values of that function for a range of 'ages' in a column.

Table 1 : Function Identification Codes

The equivalent notations in the International Actuarial Notation are given in their simplest forms; thus A_x includes $A_{x:\overline{n}|}$, A_{xx}, etc.

If the second letter of the code is 't', the function is 'due', corresponding to the use of the trema, ¨ . If the second letter is 'p', the corresponding function in the International Notation is that using the lower case (French-petit).

Code	Name	International Notation
a	Assurance	A_x
ap	Annuity in arrears	a_x
at	Annuity due	$\ddot{a}_x$
c	Commutation Function	C_x
d	Commutation Function	D_x
dp	Commutation Function	d_x
i	Rate of interest	i
it	Rate of discount	d
j	Increasing Assurance	$(IA)_x$
jp	Increasing Annuity	$(Ia)_x$
jt	Increasing Annuity (due)	$(I\ddot{a})_x$
l	Commutation Function	l_x
m	Commutation Function	M_x
mp	Commutation Function	m_x
n	Commutation Function	N_x
pp	Commutation Function (probability of survival)	p_x
p	Pure Premium for an Assurance	P_x
pr	Pure premium for an Annuity (French-rente)	${}_nP(n\|a_x)$
q	Commutation Function	q_x
r	Commutation Function	R_x
rp	Rate of Accumulation	(1+i)
s	Commutation Function	S_x
sp	Accumulation in arrears	$s_{\overline{n}\|}$
st	Accumulation in advance	$\ddot{s}_{\overline{n}\|}$

Code	Name	International Notation
v	Reserve	${}_tV_x$
vp	Present value	$v = 1 - d$
w	Paid-up Policy Value	${}_tW_x$

Table 2 - Indicator Codes

Each indicator code, if present, must appear in the order in which they are described below.

Number of Lives Indicator

This code indicates the number of lives involved; for single life functions the code is omitted and one age parameter must be coded. For functions which cannot depend upon a life, the 0 is omitted.

Possible Values	Parameters Required
0 = interest only function	nil
2 = two lives	2 ages
3 = three lives	3 ages
4 = four lives	4 ages
etc.	

Examples :

$$a(x) = A_x$$

$$a2(x,y) = A_{xy}$$

$$a3(x,y,z) = A_{xyz}$$

Select Age Indicator

This code indicates whether the lives involved are select or ultimate; for ultimate functions the code is omitted, and only the ultimate age need be coded. This indicator may only be coded if the number-of-lives indicator is not zero.

Possible Values	Parameters Required
s = Select, duration equal to zero	Select age
d = Select, non-zero duration	Select age; duration

(Note : if select and ultimate lives are to be included in the same expression, the ultimate lives may be treated as select by putting the select age equal to zero and the duration equal to the ultimate age. Alternatively an 'ultimate only' mortality table may be used for the ultimate age).

Examples :

as(x)	=	$A_{[x]}$
ad(x,t)	=	$A_{[x]+t}$
a2d(x,t,y,u)	=	$A_{[x]+t:[y]+u}$
a3d(n,0,y,0,0,z)	=	$A_{[x]:[y]:z}$

Joint Life Status Indicator

This code indicates whether a joint-life function depends upon the joint lives or upon the last surviving life. If omitted, joint status is assumed. It may only be coded if the number-of-lives indicator is 2 or more.

Possible Value	Parameters Required
k = last survivor	nil

(Note : the letter l would be preferred, but would be ambiguous with the number one.).

Examples :

a2(x,y)	=	A_{xy}
a2k(x,y)	=	$A_{\overline{xy}}$

Function Status Indicator

This code indicates that the status of thefunction changes upon the failure of the life or joint lives. If coded, a second set of Number-of-lives, Select-age and Joint-life-status indicators (and their associated parameters) are also required.

Possible Values	Parameters Required
a = status commences after death of lives	Definition of life (lives upon which continuation of the status will then depend.
b = status ceases after death of lives (i.e. in force only before)	Definition of life (lives) which must not fail before end of status.

Examples :

aa(x,y)	=	$A_{x\,\frac{2}{y}}$	y after x
ab(x,y)	=	$A\frac{1}{xy}$	x before y
ap2ka(x,y,z)	=	$a_{\overline{xy}\|z}$	on z after xy

Term Indicator

This code indicates whether the function depends upon a term of years; for "throughout life" functions, or for functions which must depend upon a term, the code is omitted.

Possible Values	Parameters Required
n = depends on a term	one term

Examples :

$$an(x,n) = A_{x:\overline{n}|}$$

$$at2n(x,y,n) = \ddot{a}_{x:y:\overline{n}|}$$

Term Status Indicator

This code indicates the relationship between the lives and the term, the latter being thought of as a life who survives n years and then dies. If omitted, the status is "joint". It may only be coded if the term indicator is n. If the code has the value 'a', a further term indicator 'n' may be used to denote a second term commencing at the end of the first.

Possible Values	Parameters Required
a = status commences after the end of the term	nil
b = status applies only before the end of the term	nil
k = status is "last survivor"	nil

Examples :

$$atn(x,n) = \ddot{a}_{x:\overline{n}|}$$

$$atnk(x,n) = \ddot{a}_{\overline{x:\overline{n}|}}$$

$$at2kn(x,y,n) = \ddot{a}_{\overline{xy}:\overline{n}|}$$

$$at2nk(x,y,n) = \ddot{a}_{\overline{x:y:\overline{n}|}}$$

$$ana(x,n) = {}_{n|}A_x$$

$$anb(x,n) = {}_{n}A_x$$

$$apna(x,n) = {}_{n|}a_x$$

$$apnan(x,n,m) = {}_{n|}a_{x:\overline{m}|}$$

Duration

Where the function is one that implicitly depends upon an elapsed duration (v,w) the duration (in years) parameter is coded immediately after the 'term' parameters. No corresponding indicator code is required.

Examples :

$v(x,t) = {}_tV_x$

$vn(x,n,t) = {}_tV_{x:\overline{n}|}$

Interest Indicator

This code indicates whether the interest rate is expressly specified. If omitted the global interest rate is assumed.

Possible Values	Parameters Required
i = interest rate is not the global rate	The annual interest rate, convertible yearly.

Examples :

$ai(x,i) = A_x$ at i%

$ani(x,n,.03) = A_{x:\overline{n}|}$ at 3%

Mortality Indicator

This code indicates whether the mortality basis is expressly specified. If omitted, the global mortality table is assumed. Each mortality table available is allocated a reference number, which is the parameter required. This reference number could refer to a computer file of mortality tables or to a table in internal programme storage.

Possible Value	Parameters Required
q = mortality table is not the global table	One mortality table reference code for each age

Examples if global mortality = a(55) males = ref. 9
and a(55) females = ref. 10

then

$ap(x) = a_x$ male

$apq(y,10) = a_y$ female

$ap2q(x,y,9,10) = a_{xy}$ female and male

$ap2q(x,y,10,10) = a_{xy}$ 2 females

$ap2(x,y) = a_{xy}$ 2 males

Frequency Indicator

This code indicates the "frequency of payment". If omitted, yearly is assumed.

Possible Values	Parameters Required
c = continuous function	Nil
m = other frequency	Number of payments per annum

Examples :

ap(x)	=	a_x
apc(x)	=	$\bar{a}_x$
apm(x,12)	=	$a_x^{(12)}$

Premium Indicator

This code is used to indicate how premiums are paid. It is omitted if the function does not consider the payment of premiums, or if premiums are payable for the whole currency of the status. If coded, a further set of indicator codes corresponding to function 'at' are required in respect of these items which are not the same.

Possible Values	Parameters Required
g = non-standard premiums, annual or 'true'	Description of the conditions for payment of premiums
h = non-standard premiums, by 'instalments'	Description of the conditions for payment of premiums

Examples :

p(x)	=	P_x	
pgn(x,n)	=	${}_nP_x$	
pn(x,n)	=	$P_{x:\overline{n}\rceil}$	
pngn(x,n,m)	=	${}^mP_{x:\overline{n}\rceil}$	limited premium endowment
pgm(x,m)	=	$P_x^{(m)}$	true premium
phm(x,m)	=	$P_x^{[m]}$	instalment premium

Table 3

This table shows the relationship between the existing International Actuarial Notation and the corresponding representation of the symbols using the proposed notation. The order of the items follows that used in the published description of the Notation.

International Actuarial Notation		Proposed Notation	Remarks
i	Interest Rate	i	
$i^{(m)}$	Nominal Rate	im(m)	e.g. $i^{(12)}$=im(12)
v	Value of 1	vp	To distinguish from v, a reserve
d	Discount Rate	it	i.e. interest due
δ	Force of Interest	ic	i.e. interest payable continuously
$a_{\overline{n}\|}$	Annuity certain	apOn(n)	e.g. $a_{\overline{10}\|}$=apOn(10)
		or apOn	(see note below)
$\ddot{a}_{\overline{n}\|}$	Annuity certain due	atOn(n)	(see note below)
		or atOn	
$\ddot{s}_{\overline{n}\|}$	Accumulation due	stOn(n)	(see note below)
		or stOn	
$s_{\overline{n}\|}$	Accumulation in arrears	spOn(n)	(see note below)
		or spOn	
l_x	Lives	l(x)	Danger of confusion with 'one'
		or l	
d_x	Deaths	dp(x)	To distinguish from $D_x = d(x)$
		or dp	
p_x	Probability of survival	pp(x)	To distinguish from $P_x = p(x)$
		or pp	
q_x	Probability of death	q(x)	
		or q	

International Actuarial Notation		Proposed Notation	Remarks
μ_x	Force of Mortality	qc(x) or qc	q.v. ic
m_x	Central death rate	mp(x) or mp	To distinguish from $M_x = m(x)$
e_x	Expectation of life	api(x,0)	i.e. an annuity at zero interest (see note below)
a_x	Life annuity	ap(x) or ap	To distinguish from $A_x = a(x)$
$\ddot{a}_x$	Life annuity due	at(x) or at	
A_x	Whole life assurance	a(x) or a	
${}_nP_x$	Probability of survival for n years	ppn(x,n) or ppn	
${}_nq_x$	Probability of death within n years	qn(x,n) or qn	
${}_nE_x$	Pure endowment	abOn(x,n) or abOn	Assurance payable at n before x (see note below)
$n\|q_x$	Probability of death in year n	qna(x,n) or qna	
$n\|a_z$	Deferred annuity	apna(x,n) or apna	
$n\|_t a_x$	Deferred temporary annuity	apnan(x,n,t)	
$a_x^{(m)}$	Annuity by instalments	apm(x,m)	
$\ddot{a}_x^{(m)}$	Annuity due by instalments	atm(x,m)	
$A_x^{(m)}$	Assurance at end of $1/m$ of a year	am(x,m)	

International Actuarial Notation		Proposed Notation	Remarks
$\bar{a}_x$	Continuous annuity	apc(x) or apc	atc(x) is equally possible!
$\bar{A}_x$	Assurance at moment of death	ac(x) or ac	
$\overset{o}{a}_x$	Complete Annuity	-	Function omitted
$\overset{o}{e}_x$	Complete expectation	apic(x,0)	Equivalent to $\bar{a}_x$ at zero interest
l_{xy}	$l_x \times l_y$	l2(x,y)	
d_{xy}	$l_{xy} - l_{x+1:y+1}$	d2(x,y)	
a_{xyz}	Joint Annuity	ap3(x,y,z)	
A_{xyz}	Joint Assurance	a3(x,y,z)	
$a_{x:\overline{n}\rceil}$	Temporary annuity	apn(x,n) or apn	
$A_{x:\overline{n}\rceil}$	Endowment Assurance	an(x,n) or an	
$a_{y\|x}$	Reversionary annuity	apa(y,x)	x after y
$A_{z\|xy}$	Contingent assurance	aa2(z,x,y)	
$a_{\overline{xyz}}$	Survivorship annuity	ap3k(x,y,z)	
$a^{2}_{\overline{xyz}}$	p^{2}_{xyz}	-	Functions omitted
$n^{q}_{\overline{xy}}$	$n^{q}_x \times n^{q}_y$	q2kn(x,y,n)	
$n^{A}_{\overline{xy}}$	Temporary assurance on last survivor	a2knb(x,y,n)	Note that a2kn(x,y,n) = $A_{\overline{x:y}:\overline{n}\rceil}$

International Actuarial Notation		Proposed Notation	Remarks
A^{1}_{xy}	Contingent assurance	ab(x,y)	"assurance before"
A^{2}_{xyz} $A^{2}_{xyz\,1}$	Contingent assurance		Functions omitted
$A_{\overline{xy}:\,z\,3}$	Contingent assurance	a2kb(x,y,z)	On survivor of x and y before z.
$A^{1}_{x:\,\overline{n}\rceil}$	Temporary assurance	anb(x,n) or anb	"before n". We could also write abOn(x,n) i.e. on death of x before (zero lives) and (n years)
$a_{\overline{yz}\rceil x}$, $a^{2}_{\overline{yz}\rceil x}$	Contingent annuities		Functions omitted
$A^{1}_{\overline{xy}\,:\,\overline{n}\rceil}$	Joint temporary assurance	a2nb(x,y,n)	Compare $A^{1}_{\overline{xy}\,:\,\overline{n}\rceil} = {}_{n}A_{\overline{xy}}$ = a2knb(x,y,n)
$\hat{a}_{y\rceil x}$, $\overset{o}{a}_{y\rceil x}$	Contingent annuities		Functions omitted
P_x	Premium - whole life	p(x) or p	
$P_{x\,:\,\overline{n}\rceil}$	Premium - endowment	pn(x,n) or pn	Note that ${}^{n}P_x$ = pgr
P^{1}_{xy}	Premium - contingent assurance	pb(x,y)	
${}_{n}P(\bar{A}_x)$	Limited Premium	pcgn(x,n) or pcgn	'premium for continuous payable n years'
${}_{t}P^{(m)}(A_{x:\overline{n}\rceil})$	Limited Premium Endowment	pngnm(x,n,t,m)	
${}_{t}V_x$	Reserve	v(x,t)	
${}_{t}W_x$	Paid up policy	w(x,t)	

International Actuarial Notation		Proposed Notation	Remarks
${}_tV^{(m)}(\bar{A}_{x:n})$	Reserve	vncgm(x,n,t,m)	

In general the expression for a reserve or paid-up policy value is similar to the net premium expression, with 'p' replaced by 'v' or 'w' and a duration parameter added.

$P'\ P''\ V'$	Commercial premiums etc.	-	Omitted
$(Ia)_x$	Increasing annuity	jp(x) or jp	
$(IA)_x$	Increasing assurance	j(x) or j	
$(Ia)_{x:\overline{n}\rceil}$	Increasing Temp. annuity	jpn(x,n) or jpn	
$(IA)^1_{x:\overline{n}\rceil}$	Increasing Temp. assurance	jnb(x,n) or jnb	q.v. $A^1_{x:\overline{n}\rceil}$ = anb(x,n)
$(I_{\overline{n}\rceil}a)_x,\ (I_{\overline{n}\rceil}A)_x$			Functions omitted
$(D_{\overline{n}\rceil}A)^1_{x:\overline{n}\rceil}(va)$			Functions omitted
D_x	Commutation Column	d(x) or d	
N_x	Commutation column	n(x) or n	
S_x	Commutation column	s(x) or s	
C_x	Commutation column	c(x) or c	
M_x	Commutation column	m(x) or m	
R_x	Commutation column	r(x) or r	
$\bar{C}_x$	Commutation column	cc(x) or cc	
$\bar{M}_x$	Commutation column	mc(x) or mc	

International Actuarial Notation		Proposed Notation	Remarks
$\bar{R}_x$	Commutation column	rc(x) or rc	
D_{xy}	Commutation column	d2(x,y)	
N_{xy}	Commutation column	n2(x,y)	
C_{xy}	Commutation column	c2(x,y)	
M_{xy}	Commutation column	m2(x,y)	
$\overset{\circ}{C}{}^{1}_{xy}$	Commutation column	cb(x,y)	x before y
M^{1}_{xy}	Commutation column	mb(x,y)	
$l_{[x]+t}$	Commutation column	ld(x,t)	
$d_{[x]+t}$	Commutation column	dd(x,t)	
$a_{[x]}$	Commutation column	aps(x) or aps	
$a_{[x-n]+n}$	Commutation column	apd(x-n,n)	
$N_{[x]}$	Commutation column	ns(x) or ns	
$\ddot{a}_{[x]}$	Commutation column	ats(x) or ats	

Note

This list contains a number of cases where the proposed notation is rather clumsy - in these cases it may be desirable to add further codes to Table 1 to simplify the writing of these functions. In the present proposals, the number of entries in Table 1 has been deliberately kept to a minimum, but the subject is open to discussion. For example, the following simplifications could be achieved (at the loss of some symmetry):-

Function	Proposed Representation	Possible Simplification
$a_{\overline{n}\|}$	apOn(n)	bp(n) or bp
$''a_{\overline{n}\|}$	atOn(n)	bt(n) or bt
e_x	api(x,o)	ep(x) or ep
$_nE_x$	abOn(x,n)	e(x,n) or e
$s_{\overline{n}\|}$	spOn(n)	t(n) or t
$''s_{\overline{n}\|}$	stOn(n)	tt(n) or tt

Table 4

This table illustrates various notations by comparing the representations of the functions listed at the beginning of Section 5.4.

Existing International Actuarial Notation	Boehm	Australia	Proposed
$a^{(12)}_{20}$	a(20,12k)	a12(20)	apm(20,12)
$a_{20:\overline{12}\vert}$	a(20,12)	b(20,12)	apn(20,12)
$a_{20:12}$	a(20,12)	aa(20,12)	ap2(20,12)
$a_{\overline{20:12}}$	a(20:12;1:)	*	ap2k(20,12
$A^{(12)}_{20}$	A(20,12k)	A12(20)	am(20,12)
$A^{1}_{20:\overline{12}\vert}$	A(20,12)	T(20,12)	anb(20,12)
$A_{20:\overline{12}\vert}^{1}$	E(20,12)	E(20,12)	abOn(20,12 or e(20,12)
$A_{20:\overline{12}\vert}$	AE(20,12) or A(20:12n)	B(20,12)	an(20,12)
$A^{(12)}_{\lceil 20 \rceil}$	A((20),12k)	A12(20:)	asm(20,12)

* joint and survivor is apparently not provided for.

References

1) Boehm and Reichel, Transaction of the 18th International Actuarial Congress, 1968.

2) Report by a Sub-committee of the Institute of Actuaries of Australia and New Zealand, 1971.