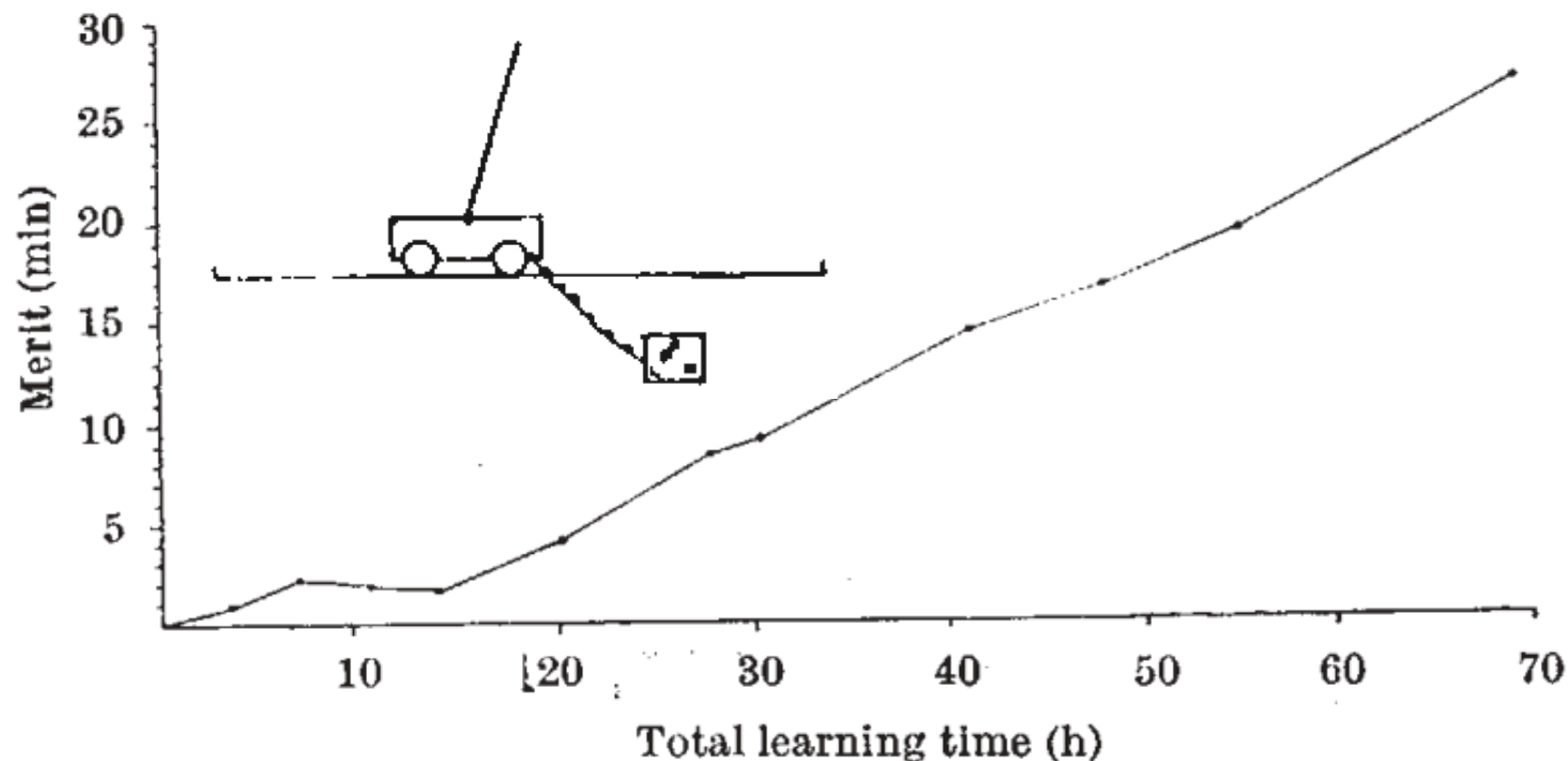


Algorithms

History



“It would be useful if computers could learn from experience and thus automatically improve the efficiency of their own programs during execution.”
Donald Michie, Nature, 1968.

Memo Functions

Factorial function, recursive “for clarity”:

```
if n < 0 or if not (n. isinteger) then undef
else
if n = 0 then 1 else n * fact (n - 1) close
end
```

Add “memo”:newmemo (fact, 100, nonop =)
→ fact;...rote has an upper fixed limit of 100
entries...the symbol nonop warns the
machine not to try to operate the “=”
function at this stage...



Elliot 4100

2-6 microseconds (MHz)

24 bits

4 x 65,536 words (96kB)

1,000 pounds (~450kg)

Algol, H, Fortran, Assembler

A red advertisement for the NCR Elliott 4100 computer. The top half of the ad features the large number '4100' in white, with the text 'Forty-one hundred' underneath it. Below this, there is a paragraph of text describing the computer's capabilities. The bottom half of the ad features a black and white photograph of the computer system, with the NCR and ELLIOTT logos and contact information below it.

4100
Forty-one hundred

The NCR Elliott 4100 is an up-to-the-minute computer in every way – design, construction, speed, simplicity and power. British designed and built for business, science and real-time computing. No multi-purpose computer is more useful or more efficient.

Specification
Silicon constructed. Over 400 program instructions. Comprehensive software – Algol, Language H, Fortran IV and NEAT 4100 assembler. Standard interface. Large range of peripherals. Processors and peripherals expandable and interchangeable.

The manufacturers reserve the right to alter, without notice, the specification for any equipment described in this bulletin.

NCR
206/216 Marylebone Road, London NW1. Tel: PADdington 7070

ELLIOTT
ELLIOTT-AUTOMATION COMPUTERS LTD.
Elmore Way, Borehamwood, Herts. Tel: ELStree 2040

Catalogue 4100 (11/60)

Evaluation

Strict

Applicative Order

Call by Value

Call by Reference

Non-Strict

Normal Order

Call by Name

Call by Need

Call by Need or Lazy

GHC Objects

S# 5050

CONSTR

FUN_STATI

C



CLOSURE

Normal Forms

Example

```
> let x = sum [1..100]
```

```
> let y = x * x
```


```
> x
```

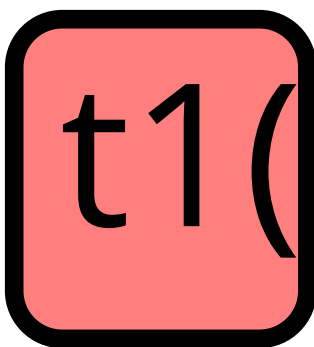
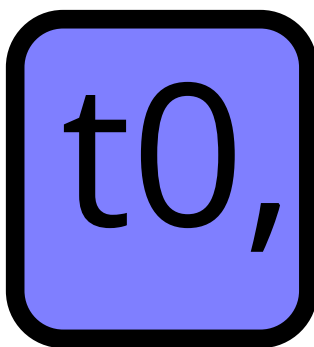
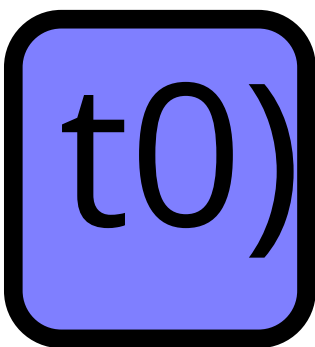
```
5050
```

```
> y
```

```
25502500
```

:view a, :view b

x: 

y:   

:eval t0

x:

S# 5050

b0

y:

t0(

b0,

b0)

:eval t0

x: S# 5050

y: S# 25502500

IntMap

Letters and Numbers

Trie

Skeleton Tree

Space Leak

Nexus