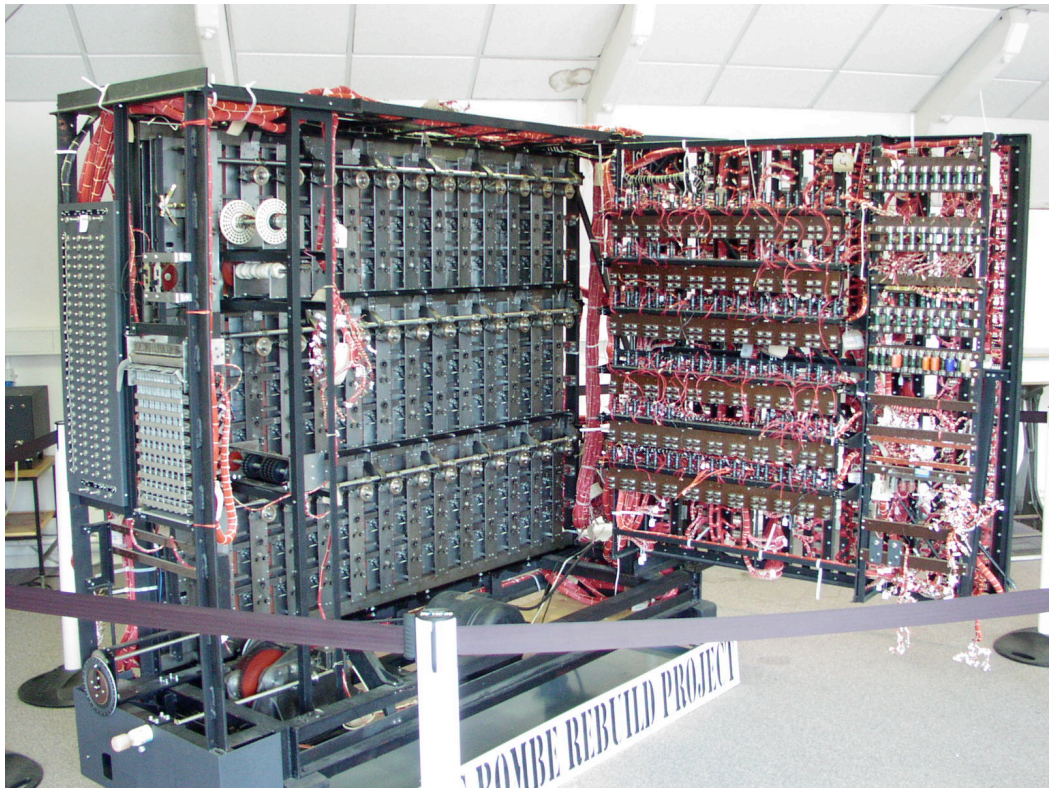
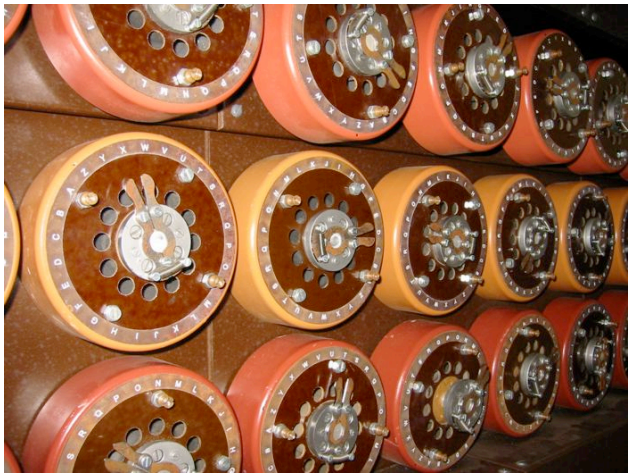
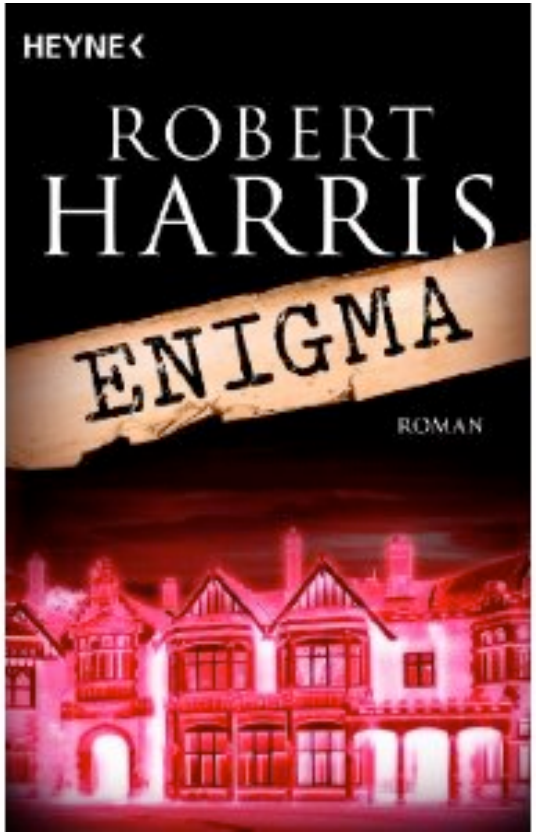
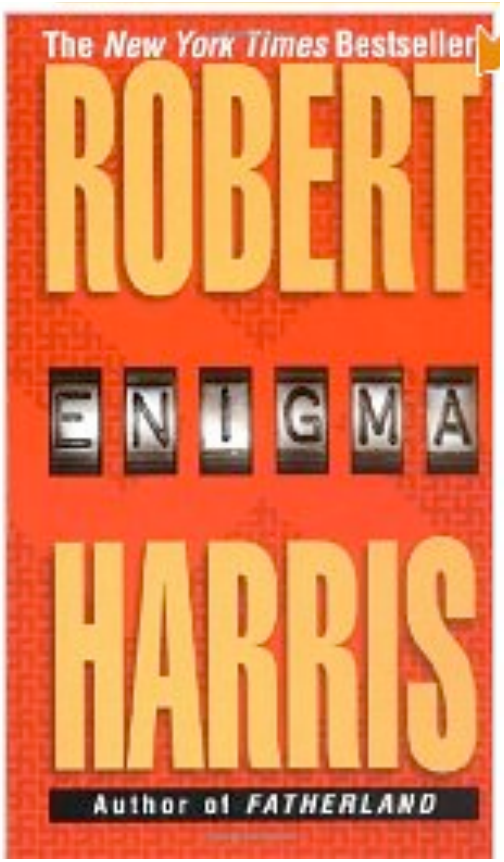


Nachbau der Turing-Bombe (für die Entschlüsselung der Enigma-Funkspüche) (Wikipedia)



Walzensatz der Turing-Bombe (Wikipedia)


HEYNE
ROBERT
HARRIS
ENIGMA
ROMAN


The New York Times Bestseller
ROBERT
ENIGMA
HARRIS
Author of FATHERLAND

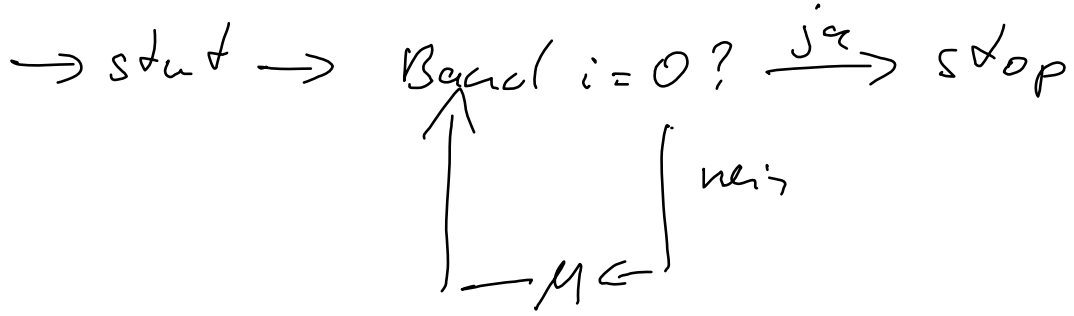
$\forall n_1, \dots, n_k, m$:

$f(n_1, \dots, n_k) = m$

ggdw

$z_0 \, bin(n_1) \# \dots \# bin(n_k) \longmapsto^* z_0 \, bin(m)$

Sei M eine TM, dann realisiert



eine WHILE-Schleife: "WHILE Band i ≠ 0 DO M"

3.3 LOOP-, WHILE-, GOTO-Berechenbarkeit

→ Berechenbarkeit mit Prog-Spr.

3.3.1 LOOP-Prog.

Bausteine

- Variable: $x_0, x_1, \ldots$
- Konstanten: $0, 1, 2, \ldots$
- Trennsymbole: ; :=
- Operatoren: $+, -$
- Schlüsselworte: LOOP DO END

Loop-Prog. sind

- Wertzuweisungen: $x_i := x_j + c$ $x_i := x_j - c$

  x_i, x_j Var., c Konst.

- Konkatenation

  $P_1 ; P_2$

  P_1, P_2 LOOP-Prog.

- Schleife:

  LOOP x_i DO P END

  x_i Var, P LOOP-Prog.

Semantik:

Zuweisung: wie üblich, wobei für "$x_j - c$" die modifizierte Subtraktion benutzt wird: $= \max(0, x_j - c)$.

Schleife: P wird x_i-mal ausgeführt (egal wie x_i im Schleifenkörper modifiziert wird)

Resultat eines Programmes:

Programm wird mit Eingabe $n_1, \ldots, n_k$ in $x_1, \ldots, x_k$ gestartet. Resultat steht hinterher in x_0.

Def Eine Fkt. $f: \mathbb{N}^k \to \mathbb{N}$ heißt LOOP-berechenbar (LB), falls es ein LOOP-Programm gibt, das mit $n_1, \ldots, n_k$ in $x_1, \ldots, x_k$ gestartet (und alle anderen Var. sind 0) mit $f(n_1, \ldots, n_k)$ in x_0 stoppt.

Bem: Alle LB Fkt sind total.

Bsp: $x_0 := x_1;$
LOOP x_2 DO $x_0 := x_0 + 1$ END | Summenprog.

Bsp:
```
LOOP x2 DO
   LOOP x1 DO
      x0 := x0 + 1
   END
END
```
| Figur für $x_1 \cdot x_2$ ↪ Produkt

Notation: In Zukunft werden wir +, *, DIV, MOD als Operatoren zwischen Var zulassen.

Bsp

```
y := 1;
LOOP x DO y := 0 END;
LOOP y DO A END
```

⇒ IF x = 0 THEN A END

Notation: IF-Statements sind auch zugelassen

3.3.2 WHILE-Progs.

Zusätzlich: WHILE $x_i \neq 0$ DO P END
P WHILE-Prog

→ LOOP wird überflüssig

```
y := x;
WHILE y ≠ 0 DO y := y - 1; P END      mit y als neuer Var.
```

(!!)

```
LOOP x DO P END
```

Def Eine Fkt $f: \mathbb{N}^k \to \mathbb{N}$ ist WHILE-berechenbar (Wb), falls es ein WHILE-Prog P gibt, das gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ (und 0 in den restlichen Var.) mit $f(n_1, \ldots, n_k)$ in x_0 stoppt, sofern $f(n_1, \ldots, n_k)$ definiert ist, sonst stoppt P nicht.

Bem Wb $\supsetneq$ Lb

Satz Alle Wb Fkt. sind Tb.

Bew: Offensichtlich.

3.3.3 GOTO-Berechenbarkeit

Anweisungen:

- Wertzuweisung $x_i := x_j \pm c$
- unbedingte Sprünge: GOTO M_i
- bedingte Sprünge: IF $x_i = c$ THEN GOTO M_i
- Stop: HALT

GOTO-Programme:

$M_1 : A_1 ; M_2 : A_2 ; \ldots , M_k : A_k$

M_i Marken (können auch weggelassen werden)

A_i Anweisungen

Semantik:

Def Goto-Berechenbarkeit (Gb) analog zu Wb.

Satz Alle Wb Fkt. sind Gb.

Bew: WHILE $x_i \neq 0$ DO P END

wird simuliert durch:

M_1: IF $x_i = 0$ THEN GOTO M_2;

P;

GOTO M_1;

M_2: ....

Satz Alle 6b Fkt. sind Wb.

Bew: Gegeben GOTO-Prog.

$M_1 : A_1;\ M_2 : A_2;\ \ldots;\ M_k : A_k$

Konstr. WHILE-Prog.:

```
PC := 1;
WHILE PC ≠ 0 DO
IF PC = 1 THEN A_1' END;
IF PC = 2 THEN A_2' END;
        :
IF PC = k THEN A_k' END;
END
```

wobei A_i' =

- $x_i := x_j \pm c;\ PC := PC + 1$ falls $A_i =$ „$x_i := x_j \pm c$"
- $PC := n$ falls $A_i =$ „GOTO M_n"
- $Y := PC;\ PC := n;$ IF $x_j \neq c$ THEN $PC := Y + 1$ END falls $A_i =$ „IF $x_j = c$ THEN GOTO M_n"
- $PC := 0$ falls $A_i =$ „HALT"

□

Satz (Kleenesches Normalform für WHILE-Prog)

Jede WB Fkt kann durch ein WHILE-Prog mit nur einer WHILE-Schleife berechnet werden.

Bew: Gegeben WHILE-Prog, transformiere in äquivalenten GOTO-Prog und dann zurück in ein WHILE-Prog. Dies hat nur eine WHILE-Schleife. □

Satz Jede TB Fkt. ist GB.

Bew: Gegeben $M=(Z, \Sigma, \Gamma, \delta, z_1, \sqcup, E)$ zur Berechnung von f. Wir sim. M durch GOTO-Prog P:

$M_1 ; P_1 ; M_2 ; P_2 ; M_3 ; P_3$

P_1: Transformiert die Eingabewerte in $x_1, \ldots, x_k$ in eine TM-Banddarstellung

P_3: Enkodiert Resultat und speichert das Resultat in x_0

Seien $Z = \{z_1, \dots, z_k\}$ und $\Gamma = \{a_1, \dots, a_m\}$

Sei $b > |\Gamma|$.

Repräsentiere Konfiguration $a_{i_1} a_{i_2} \dots a_{i_p} z_\ell \boxed{a_{j_1}} a_{j_2} \dots a_{j_q}$

durch Variablen:

$$\boxed{\begin{aligned} X &= (i_1 i_2 \dots i_p)_b \\ Y &= (j_q \dots j_2 j_1)_b \\ Z &= \ell \end{aligned}} \qquad \text{wobei } (i_1 \dots i_p)_b = \sum_{\mu=1}^{p} i_\mu \cdot b^{p-\mu}$$

$M_2 : P_2$ ist classic

M_1 : $a := Y \bmod b$;
IF $(z=1)$ AND $(a=1)$ THEN GOTO M_{11};
IF $(z=1)$ AND $(a=2)$ THEN GOTO M_{12};
⋮
IF $(z=k)$ AND $(a=m)$ THEN GOTO M_{km};

M_{11} : P_{11}; GOTO M_2
M_{12} : P_{12}; GOTO M_2
⋮

P_{ij} ist $\delta(z_i, q_j) = (z_{i'}, q_{j'}, L)$
wie folgt

```
z := i';
y := y DIV b;
y := b * y + j';
y := b *' y + (x MOD b)
x := x DIV b;
```

Andere Fälle analog

□