Computers, Programs and Logic

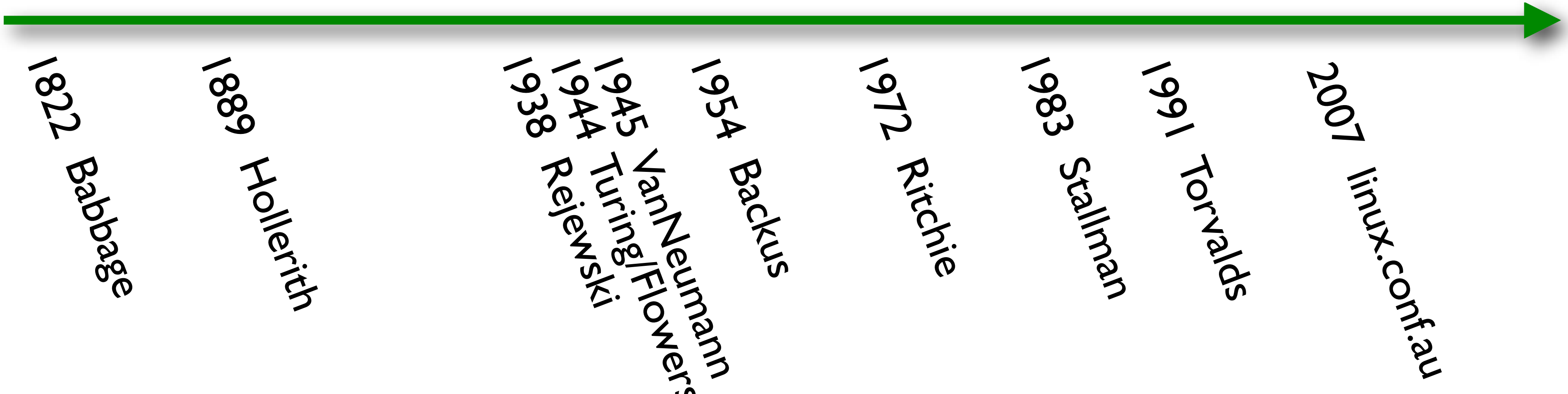
WHAT DOES LINUX PROVE?

Patryk Zadarnowski
pat@jantar.org
National ICT Australia

History of Computer Science

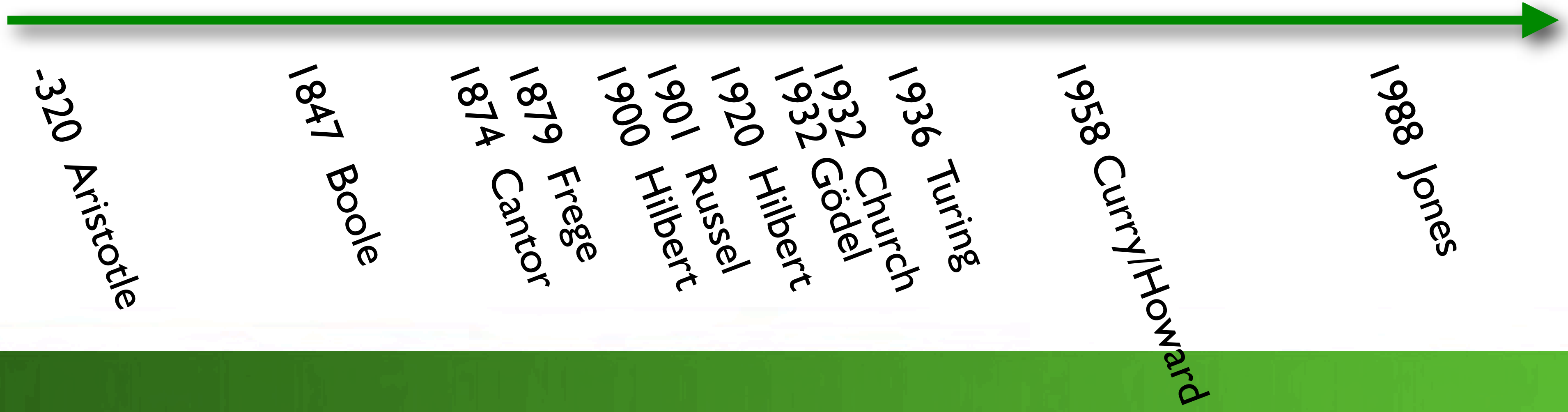
IN 3 MINUTES OR LESS

MECHANICAL CALCULATORS



1822 Babbage
1889 Hollerith
1938 Rejewski
1944 Turing/Flowers
1945 VanNeumann
1954 Backus
1972 Ritchie
1983 Stallman
1991 Torvalds
2007 linux.conf.au

FOUNDATIONS OF MATHEMATICS



-320 Aristotle
1847 Boole
1874 Cantor
1879 Frege
1900 Hilbert
1901 Russel
1920 Hilbert
1932 Gödel
1932 Church
1936 Turing
1958 Curry/Howard
1988 Jones

NATURAL DEDUCTION

WHAT DOES FORMAL LOGIC LOOK LIKE?

Gentzen’s Natural Deduction

$$\dfrac{}{A \vdash A}\;\mathit{ID}$$

$$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\;\mathit{II} \qquad \dfrac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}\;\mathit{IE}$$

$$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \,||\, B}\;\mathit{DI} \qquad \dfrac{\Gamma \vdash A \,||\, B \qquad \Gamma \vdash\ !A}{\Gamma \vdash B}\;\mathit{DE}$$

WHAT DOES A PROOF LOOK LIKE?

$$\dfrac{\dfrac{\dfrac{}{B\to C \vdash B\to C}\,ID \qquad \dfrac{\dfrac{}{A\to B \vdash A\to B}\,ID \qquad \dfrac{}{A \vdash A}\,ID}{A\to B,\ A \vdash B}\,MP}{A\to B, B\to C, A \vdash C}\,MP}{A\to B, B\to C \vdash A\to C}\,II$$

Consistency

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash A}}{\vdash A \,||\, \text{Microsoft Good}}\,\mathit{DI} \qquad \dfrac{\vdots}{\vdash !A}}{\vdash \text{Microsoft Good}}\,\mathit{DE}$$

Russel’s Paradox

Microsoft writes software
for all people (and only those)
who cannot write it for themselves.

So, does Microsoft use its own software?

$$\underbrace{\text{If this sentence is true, then } \underbrace{\text{Windows don't crash}}_{B}}_{A}$$

$$A \equiv A \to B$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}}{\vdash A \to A}}{\vdash A \to (A \to B)}}{\vdash A \to B} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}}{\vdash A \to A}}{\vdash A \to (A \to B)}}{\vdash A \to B}}{\vdash A}$$
$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$
$$\vdash B$$

ID

II

substitute A→B for A

proven earlier

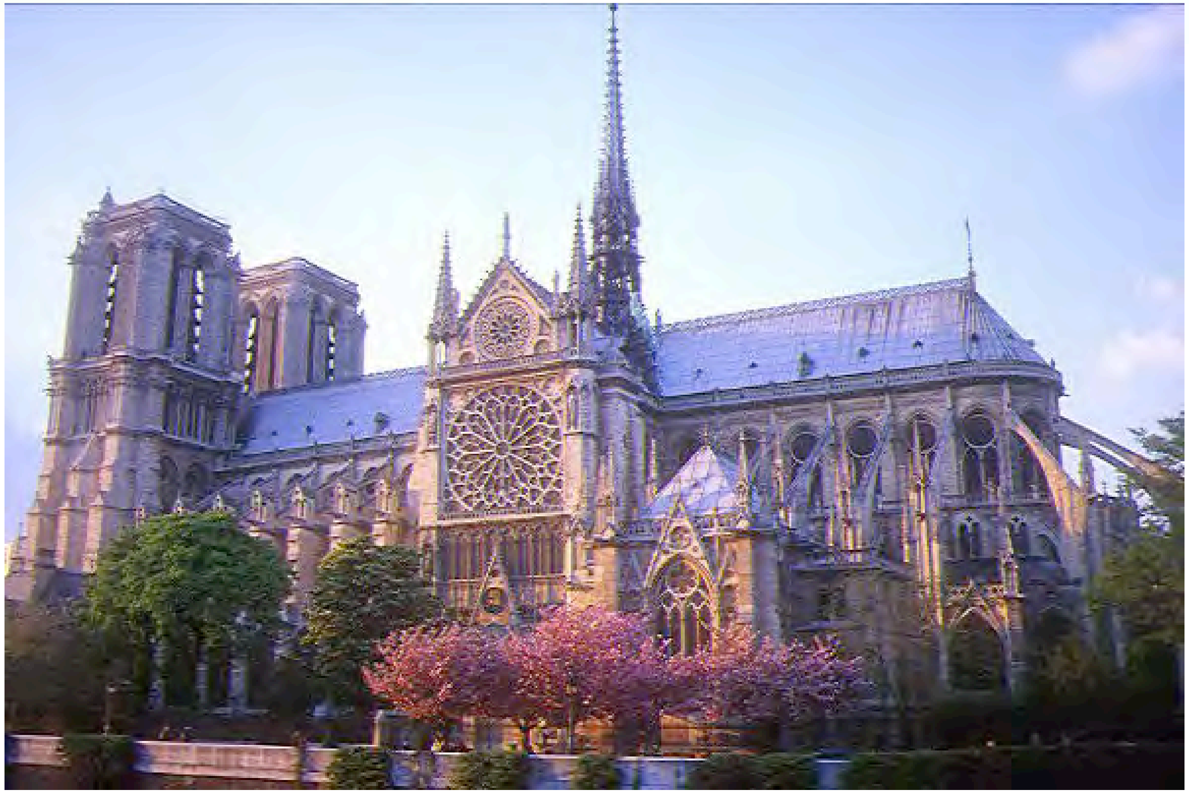
substitute A for A→B

IE

Fixing the Logic

- Logic == Foundations of Mathematics
- Bad Logic == Bad Mathematics
- Frege proposed a system of types
- We need to investigate the process of proving == formalize notion of an algorithm

WHAT DOES THAT HAVE TO DO WITH THE PRICE OF FISH?



Church

Alonzo Church

λ-calculus

~~$e ::= x \mid e_1\ e_2 \mid \lambda x.e$~~

$e ::= x \mid e_1(e_2) \mid$

FUN(x){return e;}

Booleans

```
int TRUE(x,y)  { return x; }
int FALSE(x,y) { return y; }

int NOT(P) {
  int Q(x,y) { return P(y,x); }
  return (int)Q;
}

int AND(P,Q)  { return P(Q, FALSE); }
int OR(P,Q)   { return P(TRUE, Q); }

int IF(P,A,B) { return P(A,B); }
```

Types to the Rescue!

$e ::= x \mid e_1(e_2) \mid$

t_1 FOO(t_2 x){return e;}

$t ::=$ int | ~~$t_2(*)(t_1)$~~

~~typedef t_2 (*foo_t)(t_1);~~

$t_1 \Longrightarrow t_2$

Making Types

$$\frac{}{(A\ x)\quad x\ ::\ A}\ \mathit{ID}$$

$$\frac{(A\ x)\qquad \text{return foo;}\ ::\ B}{()\quad \text{B foo(A x) \{ return foo; \}}\ ::\ A \Rightarrow B}\ \mathit{II}$$

$$\frac{()\quad \text{foo} :: A \Rightarrow B \qquad x :: A}{()\quad \text{foo(x)} :: B}\ \mathit{IE}$$

VIVA CURRY-HOWARD ISOMORPHISM!!!

THEOREM PROVING IN GCC

SHOW ME THE ~~CODE!~~

PROOF!

Theorem Proving with GCC

$$\dfrac{\dfrac{\dfrac{}{B\to C \;\vdash\; B\to C}\,\mathit{ID} \qquad \dfrac{\dfrac{}{A\to B \;\vdash\; A\to B}\,\mathit{ID} \qquad \dfrac{}{A \vdash A}\,\mathit{ID}}{A\to B,\ A \;\vdash\; B}\,\mathit{MP}}{A\to B, B\to C, A \;\vdash\; C}\,\mathit{MP}}{A\to B, B\to C \;\vdash\; A\to C}\,\mathit{II}$$

$\{\ x::A \Longrightarrow B;\ \ y::B \Longrightarrow C;\ \}$ proof $::$ $A \Longrightarrow C$

$\{\}$ foo $::$ $(A \Longrightarrow B,\ B \Longrightarrow C) \Longrightarrow (A \Longrightarrow C)$

```
typedef B (*AB)(A);
typedef C (*BC)(B);
typedef C (*AC)(A);

AC foo(AB x, BC y) {
  C proof(A a) {
    B b = x(a);
    C c = y(b);
    return c;
  }
  return proof;
}
```

Russel's Paradox Revisited

$$A \equiv A \rightarrow B$$

```
~~typedef B (*A)(A);~~

B paradox(A a) {
   return paradox(a);
}
                           int proof(A a) {
                              X x = paradox(a);
                              return 1/0;
                           }
```

Type systems save us from inconsistent theorems,

but not inconsistent proofs!

Gödel’s Incompleteness Theorem

- P: “this statement is unprovable”.
- If P is unprovable, then P is true
  ⇒ *incompleteness*
- If P is provable, then P is false
  ⇒ *inconsistency*
- Requires all statements to be recursively enumerable.

So, what *does Linux prove?*

LINUX =

user ⟹ user ⟹ user ⟹

user ⟹ user ⟹ user ⟹

user ⟹ user ⟹ user ⟹

user ⟹ user ⟹ use

user ⟹ user ⟹

What does Linux prove?

- Takes user input forever
- Never stops, never crashes
- Paradox
- Does useful stuff
- We need paradoxical proofs to prove *some* problems
- Gödel's Incompleteness Theorem!

Further Reading

- Twelf:
  http://www.cs.cmu.edu/~twelf/
- Philip Wadler:
  http://homepages.inf.ed.ac.uk/wadler/topics/history.html
- Wikipedia:
  http://www.wikipedia.org/

Booleans

```
int TRUE(x,y)  { return x; }
int FALSE(x,y) { return y; }

int NOT(P) {
  int Q(x,y) { return P(y,x); }
  return (int)Q;
}

int AND(P,Q)  { return P(Q, FALSE); }
int OR(P,Q)   { return P(TRUE, Q); }

int IF(P,A,B) { return P(A,B); }
```

```
typedef struct F *F;
struct F{
  F (*call)(F this, F x, F y);
  int args[2];
};

F NOT(F p) {
  F q = malloc(sizeof(struct F));
  q->call = mk_not;
  q->args[0] = p;
  return q;
}

F mk_not(F q, F x, F y) {
  F p = q->args[0];
  return p->call(p, y, x);
}
```