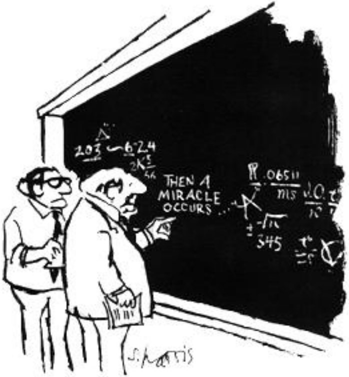
Theorem Proving



"I think you should be more explicit here in step two."

Why Theorem Prooving

- We want to make sure algorithms (and their implementations) are correct.

Why Theorem Prooving

- We want to make sure algorithms (and their implementations) are correct.
- Ideally we develop the algorithm and the proof of its correctness concurrently.

- Nice example about regular expression matching:

  'Proof-directed debugging' revisited for a first-order version by Kwangkeun Yi.

  This is based on an earlier paper by Robert Harper.

Languages

```
definition
  lang_seq :: "string set ⇒ string set ⇒ string set" ("_ ; _")
where
  "L1 ; L2 = {s1@s2 | s1 s2. s1 ∈ L1 ∧ s2 ∈ L2}"

fun
 lang_pow :: "string set ⇒ nat ⇒ string set" ("_ ↑ _")
where
  "L ↑ 0 = {[]}"
| "L ↑ (Suc i) = L ; (L ↑ i)"

definition
  lang_star :: "string set ⇒ string set" ("_⋆")
where
  "L⋆ ≡ ⋃i. (L ↑ i)"
```

Regular Expressions

```
datatype rexp =
   EMPTY
  | CHAR char
  | SEQ rexp rexp
  | ALT rexp rexp
  | STAR rexp

fun
  L :: "rexp ⇒ string set"
where
  "L(EMPTY) = {[]}"
| "L(CHAR c) = {[c]}"
| "L(SEQ r1 r2) = (L r1) ; (L r2)"
| "L(ALT r1 r2) = (L r1) ∪ (L r2)"
| "L(STAR r) = (L r)⋆"
```

Dagger

```
function
  dagger :: "rexp ⇒ char ⇒ rexp set" ("_ † _")
where
  r1: "(EMPTY) † c = {}"
| r2: "(CHAR c') † c = (if c = c' then {EMPTY} else {})"
| r3: "(ALT r1 r2) † c = r1 † c ∪ r2 † c"
| r4: "(SEQ EMPTY r2) † c = r2 † c"
| r5: "(SEQ (CHAR c') r2) † c = (if c= c' then {r2} else {})"
| r6: "(SEQ (SEQ r11 r12) r2) † c = (SEQ r11 (SEQ r12 r2)) † c"
| r7: "(SEQ (ALT r11 r12) r2) † c =
        (SEQ r11 r2) † c ∪ (SEQ r12 r2) † c"
| r8: "(SEQ (STAR r1) r2) † c =
        r2 † c ∪ {SEQ (SEQ r' (STAR r1)) r2 | r'. r' ∈ r1 † c}"
| r9: "(STAR r) † c = {SEQ r' (STAR r) | r'. r' ∈ r † c}"
```

Matcher

```
function matcher :: "rexp ⇒ string ⇒ bool" ("_ ! _")
where
  s01: "EMPTY ! s = (s =[])"
| s02: "CHAR c ! s = (s = [c])"
| s03: "ALT r1 r2 ! s = (r1 ! s ∨ r2 ! s)"
| s04: "STAR r ! [] = True"
| s05: "STAR r ! c#s =
          (False ∨ OR {SEQ (r') (STAR r)!s | r'. r' ∈ r † c})"
| s06: "SEQ r1 r2 ! [] = (r1 ! [] ∧ r2 ! [])"
| s07: "SEQ EMPTY r2 ! (c#s) = (r2 ! c#s)"
| s08: "SEQ (CHAR c') r2 ! (c#s) = (if c'=c then r2 ! s else False)"
| s09: "SEQ (SEQ r11 r12) r2 ! (c#s) = (SEQ r11 (SEQ r12 r2) ! c#s)"
| s10: "SEQ (ALT r11 r12) r2 ! (c#s) =
          ((SEQ r11 r2) ! (c#s) ∨ (SEQ r12 r2) ! (c#s))"
| s11: "SEQ (STAR r1) r2 ! (c#s) =
     (r2 ! (c#s) ∨ OR {SEQ r' (SEQ (STAR r1) r2) ! s | r'. r' ∈ r1 † c})"
```

Correctness

- Correctness of the matcher:

$$r \,!\, s \text{ implies } s \in L\, r$$
$$\neg\, r \,!\, s \text{ implies } s \notin L\, r$$