Turing Machine ASM

Signature

Q with q, Σ with Blank

Int with +1 and –1,0,1,h

T = Int → Σ (a.e. Blank)

P : Q × Σ → Q × Σ × {-1,0,1}

gives three funs F_1, F_2, F_3

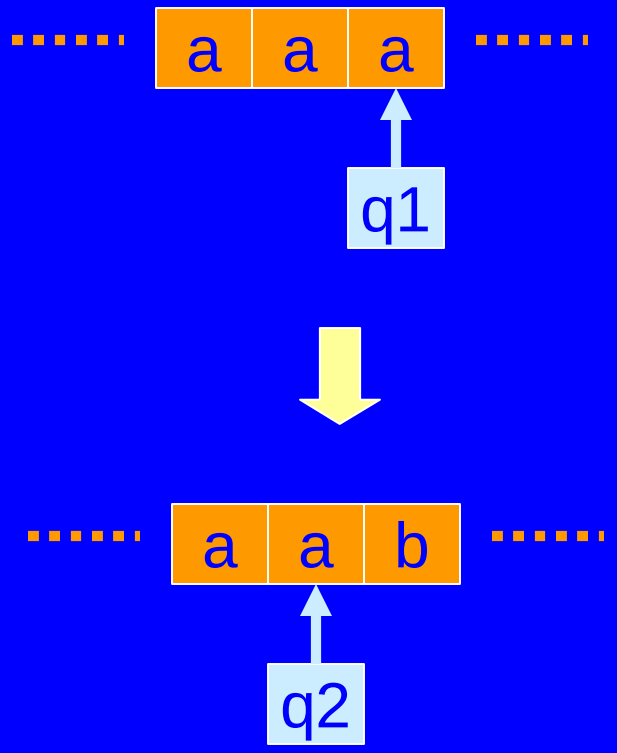
The program

q := F_1 (q,T(h))

T(h):= F_2 (q,T(h))

h := h + F_3 (q,T(h))

Instruction (q1,a)→(q2,b,-1)



© Egon Börger: Turing Machine

Refining Turing Machine ASM to FSM

Writing only as output

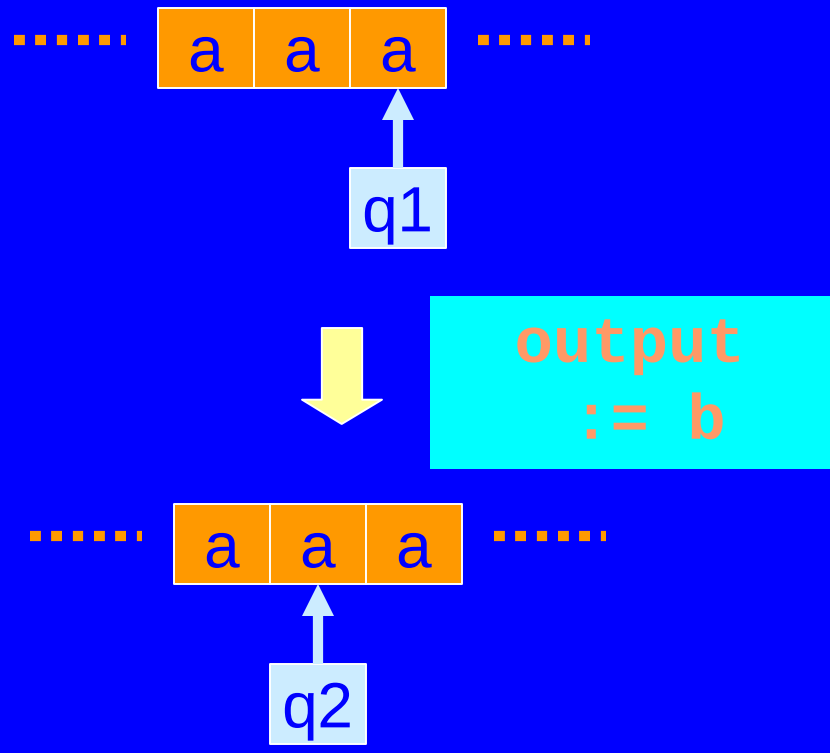
Instruction (q1,a)→(q2,b,-1)

- ASM program:

q := $F_1(q,T(h))$

output := $F_2(q,T(h))$

h := h + $F_3(q,T(h))$



one-way/two-way FSM
depending on whether besides +1 also –1 is allowed in instructions

Extending Turing Machine to Alternating TM

- **Alternating TMs are**
  - **focussed to either accept or reject the initial tape**
  - **permitted to**
    - **invoke TM-subcomputations**
    - **explore whether some or all subcomputations accept or reject their input**
- **normal execution states with program TM(Nxtctl,Write,Move):**

  control := Nxtctl(control, tape(head))
  tape(head) := Write(control, tape(head))
  head := head + Move(control, tape(head))

  enriched by
  - **control states which simply accept or reject**
  - **existential control states which accept/reject in case some/every subcomputation accepts/rejects**
  - **universal control states which accept/reject in case every/some subcomputation accepts/rejects**

Alternating Turing automata

When in an existential or universal control state subcomputations are created to be run (AltTmSpawn), the invoking computation becomes idle to observe whether the yield of the subcomputations switches from undef to either accept or reject and to define its own yield correspondingly

AlternatingTm(Nxtctl,Write,Move) =

IF type(SELF.ctlstate)=normal THEN
TM(Nxtctl,Write,Move)(SELF)
IF type(SELF.ctlstate) Î {existential,universal} THEN
AltTmSpawn(SELF)
TmYieldExistential(SELF)
TmYieldUniversal(SELF)
IF type(SELF.ctlstate) Î {accept,reject} THEN
yield(SELF):= type(SELF.ctlstate)

Alternating Turing automata

AltTmSpawn(a) = IF a.mode = running THEN
LET j_1,…,j_k = Nxtctl(a.ctlstate, a.tape(a.head))
LET a_1,…,a_k = new(Agent)
FORALL i Î {1,…, k}
Activate(a_i,a,j_i)
parent(a_i) := a
a.mode := idle

Activate(b,a,j) =
StartSubComp(b,a,j), b.mode := running, b.yield:=undef
StartSubComp(b,a,j) =
b.ctlstate:=j
b.head:=a.head
FORALL pos Î domain(a.tape) DO
b.tape(pos):=a.tape(pos)

Alternating Turing automata

TmYieldExistential(a) =

IF a.mode = idle AND type(a.ctlstate) = existential THEN
IF forall c Î children(a) yield(c)=reject
THEN yield(a) := reject
IF exists c Î children(a) yield(c)=accept
THEN yield(a) := accept

TmYieldUniversal(a) =

IF a.mode = idle AND type(a.ctlstate) = universal THEN
IF forall c Î children(a) yield(c)=accept
THEN yield(a) := accept
IF exists c Î children(a) yield(c)=reject
THEN yield(a) := reject

References

- E. Börger, R. Stärk: Abstract State Machines. A Method for High-Level System Design and Analysis Springer-Verlag 2003, Chapter 7
  - see http://www.di.unipi.it/AsmBook