Non-deterministic Ada Programs

A program can alter one 'bit' of memory at a time.

The current bit is indexed by an integer value: *at_cell*

The next bit is always adjacent to the previous one.

The program is in one of a *finite* number, s say, of states.

The next state entered is chosen *non-deterministically* from a choice of two.

A complete program is specified when the action for each state and bit value is defined.

State s is the unique *accept* state.

```
procedure nd_ada_f ( T : integer ) is
 type workspace is array ( integer range <>) of 01B;
 n : integer;
 begin
  get(n);
  declare
   w : workspace( -T..T ):=(others=>B);
   procedure do_it_on ( w : in out workspace;
                        position : in out integer;
                        state : in out integer);
    begin
     case state is
      when k =>
       if w(position)=B then
        w(position):=(0 or 1)_k; position:=position(±1)_k;
        state:=choose ( next_1 ,next_2 )_k;
       elsif w(position)=0 then
        w(position):=(0 or 1)_k; position:=position(±1)_k;
        state:=choose ( next_1 ,next_2 )_k;
       else
        w(position):=(0 or 1)_k; position:=position(±1)_k;
        state:=choose ( next_1 ,next_2 )_k;
       end if;
     end case;
    end do_it_on;
   begin
    at_cell:=1; in_state:=1; get (w (1..n));
    for i in 1..T loop
     do_it_on (w,at_cell,in_state);
    end loop;
    if in_state=accept then return true; end if;
   end;
 end nd_ada_f;
```

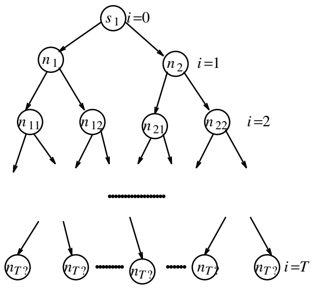
The decision problem NCOMP is:

Input: A non-deterministic program, *AP*;

An input $x \in \langle 0,1\rangle^n$ for *AP*;

A time bound T.

Question: Is there a 'computation path' of *AP* on x that accepts in T iterations?



Computation Tree

An Alternative to NP

ADA_NP is the class of decision problems, f, for which there are:

a) A non-deterministic Ada program, *NA*, **and**

b) A polynomial $p(n)$

that satisfy:

$$\forall\, x \in \{0,1\}^n \text{ s.t. } f(x)=1,$$

$$NCOMP\,(NA, x, p(n)) = 1$$

Theorem: $ADA\_NP = NP$

Proof:

a) $ADA\_NP \subseteq NP$.

Suppose that $f \in ADA\_NP$.

Let NA_f and $p(n)$ be the program and polynomial such that $\forall\, x \in \{0,1\}^n$:

$$f(x)=1 \Leftrightarrow NCOMP\,(NA_f, x, p(n)) = 1$$

A set of witnesses to $f(x)$=1 is the set of computation paths *NACP*, for NA_f.

$Check_f(\,CP,\, x\,)$=1 if *CP* is a valid *accept* computation for NA_f on input x.

Given NA_f, *CP* and x this can be checked by simulating NA_f on x using the computation path *CP*.

This test can be done in polynomial-time, so $Check_f \in P$, i.e. $f \in NP$.

b) $NP \subseteq ADA\_NP$.

Let $f \in NP$, so that $Check_f \in P$.

Let W_n be the witness set for $x \in \{0,1\}^n$.

For any $w \in W_n$ it must hold that $|w| \le p(n)$ for some polynomial $p(n)$.

(otherwise $Check_f(\,x,\, w\,)$ could not be carried out in P).

Let W_n be encoded in binary. Then $\forall\, x \in \{0,1\}^n$ there are at most $2^{p(n)}$ 'witnesses to $f(x)$=1'.

NA_f is the program which:

a) In its first $p(n)$ iterations generates each $w \in W_n$ on *some* computation path.

b) In the next $q(n)$ iterations runs $Check_f(\,x,w\,)$ algorithm (reaching *accept* $\Leftrightarrow Check_f(x,w)$=1).

$NCOMP(\,NA_f, x, p(n)+q(n)\,)$=1 $\Leftrightarrow NA_f$ has a path of length $p(n)+q(n)$ ending in *accept* on input x.

$\Leftrightarrow NA_f$ constructs a witness w for x after $p(n)$ moves which checks correctly in $q(n)$ steps

$\Leftrightarrow Check_f \in P$.

In summary, $Check_f \in P \Rightarrow f \in ADA\_NP$.

Hence (from (a) and (b)), $ADA\_NP = NP$.

The *Satisfiability Problem* (*SAT*) is:

Input: A set of n Boolean variables $X_n = \langle x_1, x_2, \ldots, x_n \rangle$; A Boolean formula F over Xn, (whose length is at most $r(n)$ for some poly. r)

Question: Is there an assignment, α of true/false to the variables X_n with the property that $F(\,\alpha\,) = true$?

Cook's Theorem

SAT is *NP*-complete.

Proof: We have to do 2 things:

a) Prove that $SAT \in NP$;

b) Show that *for any* decision problem $f \in NP$: $f \leq_p SAT$:

i.e. Any instance x of f can be transformed into an instance $F(x)$ of SAT such that $f(x)=1 \Leftrightarrow SAT(F(x))=1$.

That (a) holds has already been shown.

Now let f be *any decision problem* in NP.

We know that f also belongs to ADA_NP (since $NP = ADA\_NP$).

So there is a non-deterministic Ada program NA_f for f and a polynomial $p(n)$ such that, $\forall\, x \in \{0,1\}^n$,

$$f(x)=1 \Leftrightarrow NCOMP(NA_f, x, p(n)) = 1$$

To show that SAT is NP-complete, we build an instance of SAT (i.e. a Boolean formula), $H(NA_f)(X_{m(n)})$ with the property

$$H(NA_f)(X_{m(n)}) \text{ is satisfiable}$$

$$\Leftrightarrow$$

$$NCOMP(NA_f, x, p(n))=1.$$

What do we know about NA_f on input $x \in \{0,1\}^n$?

At any value i of its main **for** loop its status is *fully* described by

A1) The value of *in_state*.

A2) The value of *at_cell*.

A3) The content (0, 1, or B) of each 'bit' $w(k)$.

These can change **only** in accordance with what the procedure *do_it_on* allows, given the initial n-bit input x.

So, suppose we wish to know if a given sequence of $p(n)$ configurations (i.e. state, position, memory) is a *valid accepting computation path* on NA_f.

Then such a sequence **must** embody **all** of the following characteristics at **all** times i:

C1) At time $i=0$: *in_state*$=1$, $w(1..n)=x$, all other bits equal B; *at_cell*$=1$.

C2) *in_state* has exactly one value at time i.

C3) *at_cell* has exactly one value at time i.

C4) $w(k)$ has exactly one value at time i, $(\forall k)$

C5) *in_state*$=s$ (*accept* state) at $i=p(n)$.

C6) $w(-p(n)...p(n))$, *at_cell*, and *in_state* must change from time i to time $i+1$ according the **case** statement in *do_it_on* of NA_f.

We construct $H(NA_f)$ with the Boolean variables: (below $0 \leq i \leq p(n)$)

V1) $3(2p(n)+1)(p(n)+1)$ variables

$$W(v,k,i)\ v \in \{0,1,B\},\ -p(n) \leq k \leq p(n)$$

V2) $s(p(n)+1)$ variables

$$S(st,i)\ 1 \leq st \leq s,$$

V3) $(2p(n)+1)(p(n)+1)$ variables

$$A(k,i)\ -p(n) \leq k \leq p(n)$$

$H(NA_f(x))$ is built so that: in a satisfying assignment those variables which which are assigned the value **true** define a valid computation, CP of NA_f on x:

V1T) $W(v,k,i)=1 \Leftrightarrow$ in CP: at time i, 'bit' $w(k)$ contains v.

V2T) $S(st,i)=1 \Leftrightarrow$ in CP: at time i, *in_state*$=st$.

V3T) $A(k,i)=1 \Leftrightarrow$ in CP: at time i, *at_cell*$=k$.

Thus the whose only satisfying assignments of $H(NA_f)(V1,V2,V3)$ will be as given by (V1T, V2T, V3T) **and** define legal computations of NA_f on x, i.e. capture the conditions (C1) through (C6)

(For brevity we assume $\wedge$ is implied).

C1)

$$S(1,0)A(1,0)\bigwedge_{k=1}^{n} W(x_i,k,0)\bigwedge_{k=-p(n)}^{0} W(B,k,0)\bigwedge_{k=n+1}^{p(n)} W(B,k,0)$$

C2) 'exactly one value' $\equiv$ '*at least* one value' and *at most* one value.

$$\bigwedge_{i=0}^{p(n)}\Big(\bigvee_{st=1}^{s} S(st,i)\Big)\bigwedge_{1\leq c<d\leq s}(\neg S(c,i)\vee\neg S(d,i))$$

C3) Similarly,

$$\bigwedge_{i=0}^{p(n)}\Big(\bigvee_{k=-p(n)}^{p(n)} A(k,i)\Big)\bigwedge_{-p(n)\leq c<d\leq p(n)}(\neg A(c,i)\vee\neg A(d,i))$$

C4) Again, similarly,

$$\bigwedge_{i=0}^{p(n)}\bigwedge_{k=-p(n)}^{p(n)}\Big(\bigvee_{v\in\{0,1,B\}} W(v,k,i)\Big)\bigwedge_{v_1\neq v_2}(\neg W(v_1,k,i)\vee\neg W(v_2,k,i))$$

C5) $S(s, p(n))$

C6) is, superficially, more complicated, however at time i: from

in_state$=st$, *at_cell*$=k$, $w(k) \in \{0,1,B\}$

the **case** statement determines the values at time $i+1$ of

the new value of *at_cell*

the content of $w(k)$,

the two possible next states to jump to.

Thus each of the s states is described by a 'move' function:

$$\delta(st, v) = ((st_1, st_2), nv, ch)$$

where $v \in \{0,1,B\}$, $nv \in \{0,1\}$, $ch = \pm 1$

We thus have (C6) as

$$\bigwedge_{i=1}^{p(n)-1}\bigwedge_{j=-p(n)}^{p(n)}\bigwedge_{\delta(st,v)} MOVE(i,j,st,v,st_1,st_2,nv,ch)$$

where $MOVE(i,j,st,v,st_1,st_2,nv,ch)$ is,

$$S(st,i)\wedge A(j,i)\wedge W(v,j,i)\Rightarrow$$
$$(S(st_1,i+1)A(j+ch,i+1)W(nv,j,i+1)\vee$$
$$S(st_2,i+1)A(j+ch,i+1)W(nv,j,i+1))$$

Thus,

If at time i

NA_f is *in_state st*

at_cell j which contains v then

at time $i+1$,

NA_f is *in_state* st_1 or st_2

at_cell $j+ch$ with cell j containing nv.

So the final formula $H(NA_f)(V1,V2,V3)$ is

$$C1 \wedge C2 \wedge C3 \wedge C4 \wedge C5 \wedge C6(V1,V2,V3)$$

$H(NA_f)(V1,V2,V3)$ is satisfiable

$\Leftrightarrow$

$NCOMP(NA_f, x, p(n))=1$.

If $NCOMP(NA_f,x,p(n))=1$ setting the variables of V1, V2, V3 that encode the accepting computation path of NA_f gives a satisfying assignment of H;

H guarantees that the **true** variables of any satisfying assignment must represent a valid computation of NA_f on input x that ends in *accept* after $p(n)$ steps. i.e. a witness that

$$NCOMP(NA_f,x,p(n))=1 \qquad \square$$