Alternation

The concept of a nondeterministic acceptor of a boolean query has a long and rich history, going back to various kinds of nondeterministic automata.

It is important to remember that these are fictitious machines: we suspect that they cannot be built.

Open question: \( \text{NP} \ ? = \ \text{co-NP} = \{ \overline{A} \mid A \in \text{NP} \} \)

If one could really build an NP machine, then one could, with a single gate to invert its answer, also build a co-NP machine.

From a practical point of view, the complexity of a problem \( A \) and its complement, \( \overline{A} \) are identical.
Nondeterminism

Value(ID) := Value(LeftChild(ID)) \lor Value(RightChild(ID))
The states of an alternating Turing machine are split into: Existential states (∃) and Universal states (∀).

**Definition 13.1** An alternating TM in ID₀ accepts iff

1. ID₀ is in a final accepting state, or
2. ID₀ is in an ∃ state and some next ID’ accepts, or
3. ID₀ is in a ∀ state, has at least one next ID, and all next ID’s accept.
From now on assume that our Turing machines have a **random access** read-only input. There is an **index tape** which can be written on and read like other tapes. Whenever the value \( h \), written in binary, appears on the index tape, the read head will automatically scan bit \( h \) of the input.
**Definition 13.2** Let $\text{ASPACE}[s(n)]$ and $\text{ATIME}[t(n)]$ be the set of problems accepted by alternating TM’s using $O(s(n))$ tape cells, $O(t(n))$ time, respectively, in any computation path on any input of length $n$. □

**Theorem 13.3** [Alternation Thm.] For $s(n) \geq \log n$, and for $t(n) \geq n$,

$$\bigcup_{k=1}^{\infty} \text{ATIME}[(t(n))^k] = \bigcup_{k=1}^{\infty} \text{SPACE}[(t(n))^k]$$

$$\text{ASPACE}[s(n)] = \bigcup_{k=1}^{\infty} \text{DTIME}[k^{s(n)}]$$

**Corollary 13.4** $\text{ASPACE}[\log n] = \text{P}$ and $\text{ATIME}[n^{O(1)}] = \text{PSPACE}$. 
Definition 13.5 The **monotone, circuit value problem** (MCVP) is the subset of CVP in which no negation gates occur.

Proposition 13.6 \[\text{MCVP} \in \text{ASPACE}[\log n].\]

**Proof:** Let \( G \) be a monotone boolean circuit. For \( a \in V^G \), define “EVAL(a),”

1. if (InputOn(a)) then accept
2. if (InputOff(a)) then reject
3. if \((G \land (a))\) then universally choose child \( b \) of \( a \)
4. if \((G \lor (a))\) then existentially choose child \( b \) of \( a \)
5. Return(EVAL(b))

\( M \) simply calls EVAL(\( r \)). EVAL(a) returns “accept” iff gate \( a \) evaluates to one.

Space used for naming vertices \( a, b \): \( O(\log n) \).
The above circuit is a member of MCVP because it just has $\land$ and $\lor$ gates and it evaluates to 1.
Def: The quantified satisfiability problem (QSAT) is the set of true formulas of the following form:

$$\Psi = Q_1x_1 \land Q_2x_2 \land \cdots \land Q_rx_r(\varphi)$$

For any boolean formula $\varphi$ on variables $x$,

$$\varphi \in \text{SAT} \iff \exists x (\varphi) \in \text{QSAT}$$
$$\varphi \notin \text{SAT} \iff \forall x (\neg \varphi) \in \text{QSAT}$$

Thus QSAT logically contains SAT and SAT.
Proposition 13.7  \( \text{QSAT} \in \text{ATIME}[n] \).

Proof: Construct ATM, \( A \), on input, \( \Phi \equiv \)

\[
\exists x_1 \ \forall x_2 \ \cdots \ \exists x_{2k-1} \ \forall x_{2k} \ \bigwedge_{i=1}^{r} \bigvee_{j=1}^{s} \ell_{ij} \\
b_1 \ b_2 \ \cdots \ b_{2k-1} \ b_{2k} \ i \ j \ \ell_{ij}(b_1, \ldots, b_{2k})
\]

Quantifiers:

- in \( \exists \) state, \( A \) writes a bit \( b_1 \) for \( x_1 \),
- in \( \forall \) state, \( A \) writes a bit \( b_2 \) for \( x_2 \), and so on.

Boolean operators:

- in \( \forall \) state, \( A \) chooses \( i \),
- in \( \exists \) state, \( A \) chooses \( j \)

Final state: accept iff \( \ell_{ij}(b_1, \ldots, b_{2k}) \) is true.

\[ A \text{ accepts } \Phi \iff \Phi \text{ is true.} \]
**Theorem 13.8** For any $s(n) \geq \log n$, \( \text{NSPACE}[s(n)] \subseteq \text{ATIME}[s(n)^2] \subseteq \text{DSPACE}[s(n)^2] \).

**Proof:** \( \text{NSPACE}[s(n)] \subseteq \text{ATIME}[s(n)^2] \):

Let $N$ be an \( \text{NSPACE}[s(n)] \) Turing machine.

Let $w$ be an input to $N$, $n = |w|$.

\[
w \in L(N) \iff \text{CompGraph}(N, w) \in \text{REACH}
\]
\( w \in \mathcal{L}(N) \iff \text{CompGraph}(N, w) \in \text{REACH} \)

\[
P(d, x, y) \equiv \text{“In CompGraph}(N, w), \text{dist}(x, y) \leq 2^d”
\]

\[
P(d, x, y) \equiv \exists z \ (P(d - 1, x, z) \land P(d - 1, z, y))
\]

1. **Existentially:** choose middle ID \( z \).
2. **Universally:** \((x, y) := (x, z) \land (z, y)\)
3. Return\((P(d - 1, x, y))\)

\[
T(d) = O(s(n)) + T(d - 1) = O(d \cdot s(n))
\]
\[
d = O(s(n))
\]
\[
T(d) = O((s(n))^2)
\]
ATIME[\(t(n)\)] \subseteq DSPACE[\(t(n)\)]

Let \(A\) be an ATIME[\(t(n)\)] machine, input \(w, \ n = |w|\).

CompGraph(\(A, w\)) has depth \(c(t(n))\) and size \(2^{c(t(n))}\), for some constant \(c\).

Search this and/or graph systematically using \(c(t(n))\) extra bits of space.

\[
\text{ATIME}[t(n)] \subseteq \text{DSpace}[t(n)]
\]
Evaluate computation graph of ATIME[t(n)] machine using t(n) space to cycle through all possible computations of A on input w.
Example: $\text{ATIME}[t(n)] \subseteq \text{DSPACE}[t(n)]$
Theorem 13.9 \( \text{ASPACE}[s(n)] = \text{DTIME}2^{O(s(n))} \)

**Proof:** \( \text{ASPACE}[s(n)] \subseteq \text{DTIME}[2^{O(s(n))}] \):

Let \( A \) be an \( \text{ASPACE}[s(n)] \) machine, \( w \) an input, \( n = |w| \).

\( \text{CompGraph}(A(w)) \) has size \( \leq 2^{O(s(n))} \)

Marking algorithm evaluates this in \( \text{DTIME}2^{O(s(n))} \).
DTIME\([2^{O(s(n))}] \subseteq \text{ASPACE}[s(n)]\):

Let \(M\) be DTIME\([2^{k(s(n))}]\) TM, \(w\) an input, \(n = |w|\).

alternating procedure \(C(t, p, a)\) accepts iff contents of cell \(p\) at time \(t\) in \(M\)’s computation on input \(w\) is symbol \(a\).

\(C(t + 1, p, b)\) holds iff the three symbols \(a_{-1}, a_0, a_1\) in tape positions \(p - 1, p, p + 1\) lead to a “b” in position \(p\) in one step of \(M\)’s computation.

\[
C(t + 1, p, b) \equiv \bigvee_{(a_{-1}, a_0, a_1) \in \delta_t b} \bigwedge_{i \in \{-1, 0, 1\}} C(t, p + i, a_i)
\]

Space needed is \(O(\log 2^{k(s(n))}) = O(s(n))\).

Note that \(M\) accepts \(w\) iff \(C(2^{k(s(n))}, 1, \langle q_f, 1 \rangle)\)

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>1</th>
<th>(\bar{s})</th>
<th>(n - 1)</th>
<th>(n)</th>
<th>(2^{ks(n)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\langle q_0, w_0 \rangle)</td>
<td>(w_1)</td>
<td>\ldots</td>
<td>(w_{n-1})</td>
<td>(\square)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>1</td>
<td>(w_0)</td>
<td>(\langle q_1, w_1 \rangle)</td>
<td>\ldots</td>
<td>(w_{n-1})</td>
<td>(\square)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\bar{t})</td>
<td></td>
<td></td>
<td>(a_{-1})</td>
<td>(a_0)</td>
<td>(a_1)</td>
<td></td>
</tr>
<tr>
<td>(\bar{t} + 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2^{ks(n)})</td>
<td>(\langle q_f, 1 \rangle)</td>
<td>(\square)</td>
<td>\ldots</td>
<td>(\square)</td>
<td>(\square)</td>
<td>(\ldots)</td>
</tr>
</tbody>
</table>

\[
C(t + 1, p, b) \equiv \bigvee_{(a_{-1}, a_0, a_1) \in \delta_t b} \bigwedge_{i \in \{-1, 0, 1\}} C(t, p + i, a_i)
\]

This completes the proof of the Alternation Thm. \(\square\)
Alternation

Arithmetic Hierarchy

FO(\(\exists(N)\))

r.e. complete

Halt

r.e. complete

Halt

Recursive

Halt

co-r.e. complete

FO-SAT

Halt

FO-SAT

Halt

Arithmetic Hierarchy

FO(\(\forall(N)\))

\(r.e.\)

FO-VALID

Halt

co-r.e. complete

FO-SAT

Halt

FO-SAT

Halt

Primitive Recursive

SuccinctHornSAT

EXPTIME complete

EXPTIME

SO(LFP)

SO[\(2^{n^{O(1)}}\)]

PSPACE

QSAT

PSPACE complete

FO[\(2^{n^{O(1)}}\)]

FO(PFP)

SO(TC)

SO[n^{O(1)}]

PTIME Hierarchy

SO

NP complete

SAT

co-NP complete

SAT

co-NP

SO\(\forall\)

NP

SO\(\exists\)

NP \(\cap\) co-NP

\(\exists\) co-NP

\(\forall\) NP

NP \(\cap\) co-NP

P complete

P

P

Horn-SAT

“truly feasible”

AC\(^1\)

sAC\(^1\)

2SAT

NL comp.

NL

2COLOR

L comp.

L

NC\(^1\)

NC\(^1\)

FO(COUNT)

FO(REGULAR)

FO(DTC)

FO(TC)

FO(CFL)

FO(log n)

FO[log n]

FO[(log n)^{O(1)}]

FO[\(n^{O(1)}\)]

FO(LFP)

SO(Horn)

\(\forall\) SAT

\(\exists\) SAT

LOGTIME Hierarchy

AC\(^0\)

ThC\(^0\)

AC\(^0\)