Lecture 16: Parallel Computing

Simultaneous Resource Classes

Def: ATIME-ALT$[t(n), a(n)]$ is the set of problems solved by alternating Turing machines in time $O(t(n))$, while making at most $a(n)$ alternations between existential and universal states, starting with existential.

Example:
$$\text{ATIME-ALT}[t(n), 0] \;=\; \text{NTIME}[t(n)]$$

Similarly, define the simultaneous classes:

$$\text{ASPACE-TIME}[s(n), t(n)], \qquad \text{ASPACE-ALT}[s(n), a(n)]$$

LH and PH

Define the LOGTIME hierarchy (LH) and the PTIME hierarchy (PH) as follows:

$$\begin{aligned} \text{LH} &= \text{ATIME-ALT}[\log n, O(1)] \\ \text{PH} &= \text{ATIME-ALT}[n^{O(1)}, O(1)] \end{aligned}$$

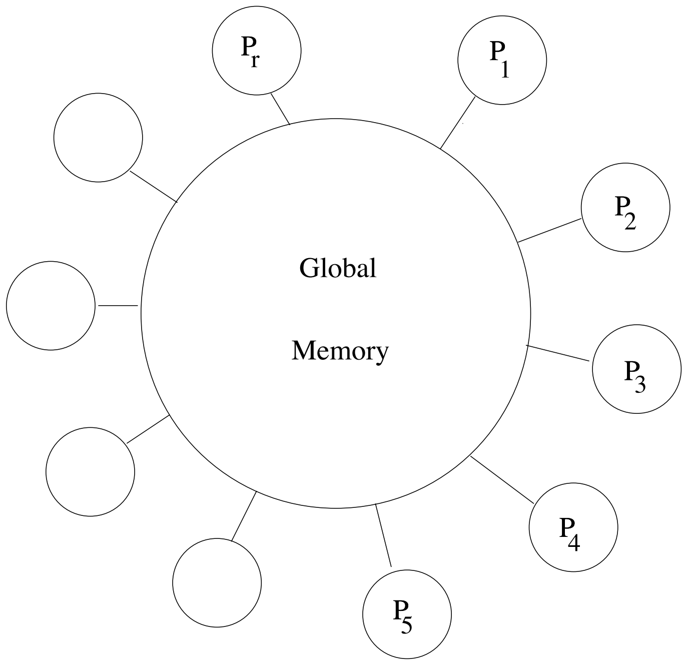
Thm: PH = SO

Proof: [idea] Follows from Fagin's Thm: NP $=$ SO$\exists$. □

Fact: LH = FO

CRAM$[t(n)] =$ CRCW-PRAM-TIME$[t(n)]$-HARD$[n^{O(1)}]$

synchronous, concurrent, $n^{O(1)}$ processors and memory



priority write: lowest number processor wins conflict

common write: no conflicts allowed

Thm: For all $t(n)$, $\quad \text{CRAM}[t(n)] \;=\; \text{AC}[t(n)]$.

Proof: (Sketch)

$\text{CRAM}[1] \subseteq \text{AC}[1] \;=\; \text{AC}^0 \;=\; \text{FO}$:

Only the global memory is tricky: all else is sequential.

$B(i,w,b)$ means that bit i of global memory word w is b.

$$\begin{aligned} B'(i,w,b) \quad \equiv \quad & B(i,w,b) \wedge \forall p\,(\text{“}p \text{ didn't just write into word } w\text{”}) \\ \vee \quad & \exists p\,(\text{“}p \text{ just wrote into word } w\text{ ”} \wedge R(p,i,b)) \end{aligned}$$

CRAM$[1] \supseteq$ FO:

$\forall x(\alpha(x))$

```
1. $w := 1$
2. Each processor $P_i$ in parallel, **do** {
3.     **if** $(\neg\alpha(i))$ **then Write**(0,w)
4. }
```

$\exists x(\alpha(x))$

```
1. $w := 0$
2. Each processor $P_i$ in parallel, **do** {
3.     **if** $(\alpha(i))$ **then Write**(1,w)
4. }
```

$\square$

Def: sAC^1 (semi-unbounded sAC^1) is the subset of AC^1 where the and-gates are binary and only the or-gates are unbounded.

Fact: [Ruzzo] $\quad \text{sAC}^1 \;=\; \log(\text{CFL}) \;=\; \text{FO}(\text{CFL}) \;=\; \big\{S \;\big|\; \exists\, \text{CFL}\, C\, (S \le C)\big\}$

Thm: $\quad \text{NC}^1 \;\subseteq\; \text{L} \;\subseteq\; \text{NL} \;\subseteq\; \text{sAC}^1 \;\subseteq\; \text{AC}^1$

Proof:

Alternation as Parallelism

Fact: [Ruzzo and Tompa] For $t(n) \geq \log n$,

$$\begin{aligned} \text{ASPACE-TIME}[\log n, t(n)] &= \text{NC}[t(n)] \\ \text{ASPACE-ALT}[\log n, t(n)] &= \text{AC}[t(n)] \end{aligned}$$

Cor: $\text{ATIME}[\log n] = \text{NC}^1$

$$\begin{aligned} \text{ASPACE-TIME}[\log n, t(n)] &= \text{NC}[t(n)] \\ \text{ASPACE-ALT}[\log n, t(n)] &= \text{AC}[t(n)] \end{aligned}$$

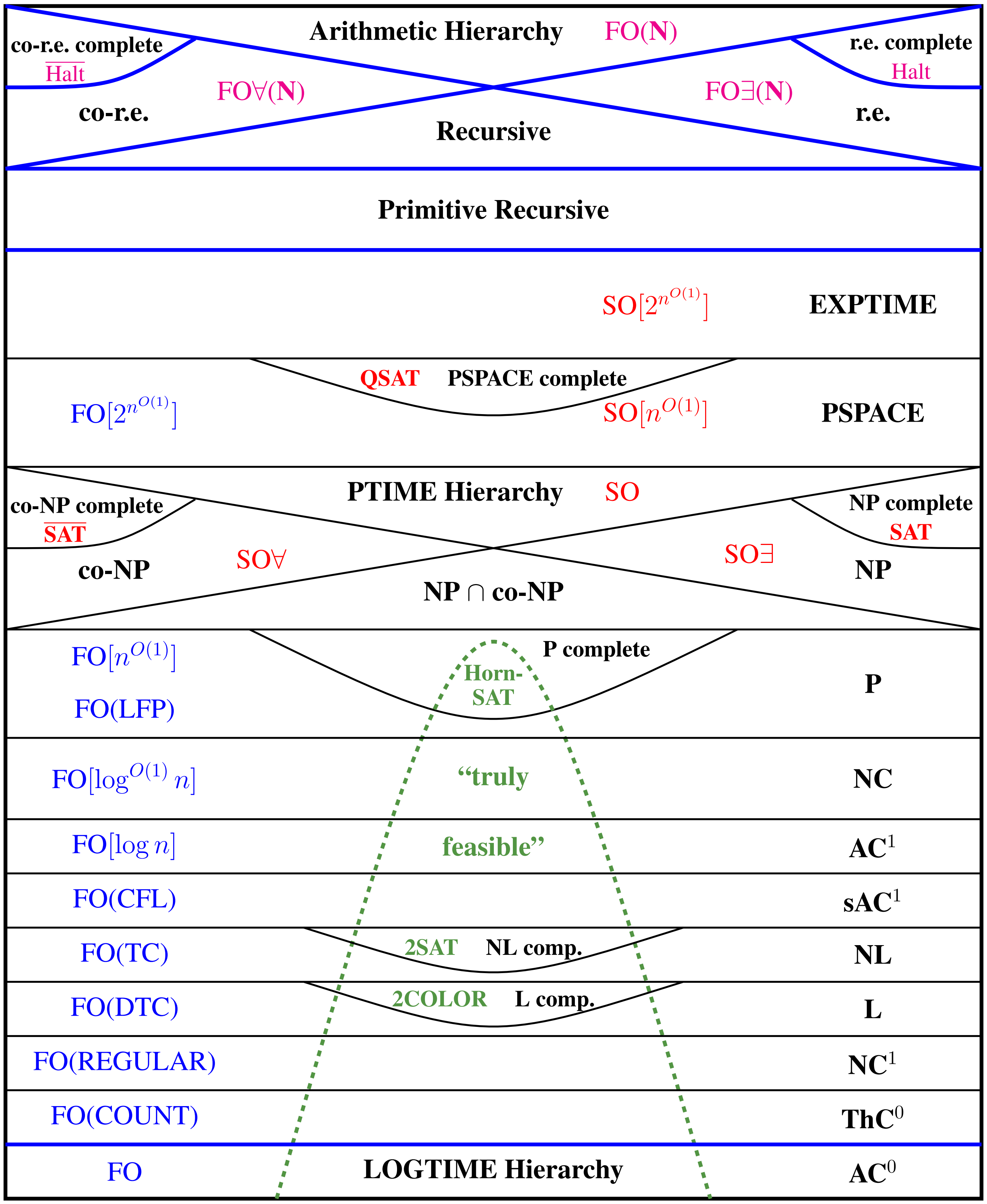
Proof: (sketches)

The computation graph of an ASPACE-TIME$[\log n, t(n)]$ machine on an input of size n is an NC$[t(n)]$ circuit.

The computation graph of an ASPACE-ALT$[\log n, t(n)]$ machine is a series of $t(n)$ computation graphs of NL or co-NL machines.

We can modify the ASPACE-ALT$[\log n, t(n)]$ machine to make a series of alternating guesses: ID$_1$, ID$_2$, ... ID$_{t(n)}$ of the endpoints of each of these NL or co-NL computations, and **check only once** that ID$_{i+1}$ follows correctly from ID$_i$.

Since NL $\subseteq$ AC^1, the whole thing is an AC$[t(n) + \log n]$ = AC$[t(n)]$ circuit. $\square$



Arithmetic Hierarchy
$\mathrm{FO}(\mathbf{N})$
co-r.e. complete
$\overline{\mathrm{Halt}}$
r.e. complete
Halt
$\mathrm{FO}\forall(\mathbf{N})$
$\mathrm{FO}\exists(\mathbf{N})$
co-r.e.
r.e.
Recursive
Primitive Recursive
$\mathrm{SO}[2^{n^{O(1)}}]$
EXPTIME
QSAT
PSPACE complete
$\mathrm{FO}[2^{n^{O(1)}}]$
$\mathrm{SO}[n^{O(1)}]$
PSPACE
PTIME Hierarchy
SO
co-NP complete
$\overline{\mathrm{SAT}}$
NP complete
SAT
$\mathrm{SO}\forall$
$\mathrm{SO}\exists$
co-NP
NP
NP ∩ co-NP
$\mathrm{FO}[n^{O(1)}]$
P complete
Horn-SAT
FO(LFP)
P
$\mathrm{FO}[\log^{O(1)} n]$
“truly
NC
$\mathrm{FO}[\log n]$
feasible”
AC^1
FO(CFL)
sAC^1
FO(TC)
2SAT
NL comp.
NL
FO(DTC)
2COLOR
L comp.
L
FO(REGULAR)
NC^1
FO(COUNT)
ThC^0
FO
LOGTIME Hierarchy
AC^0