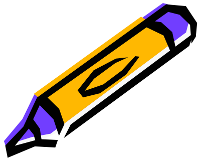
Project Proposal:

Parallel Nondeterminator

He Yuxiong



Objective

I propose Parallel Nondeterminator to:

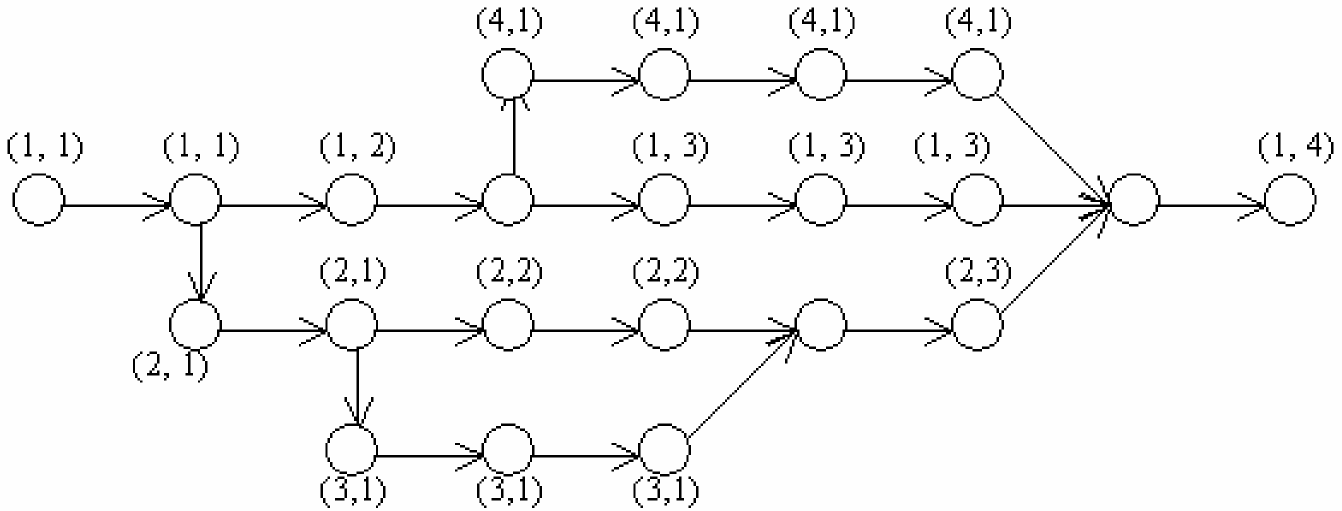
- Check the determinacy race in the parallel execution of the program written in the language like Cilk
- Develop efficient algorithm to decide the concurrency between threads
- Develop efficient algorithm to reduce the number of entries in access history

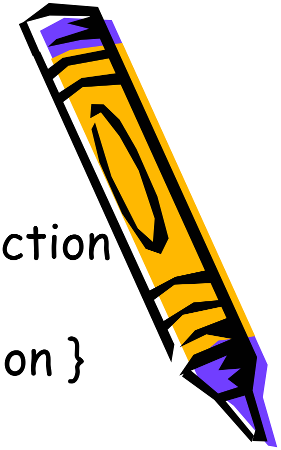
Primitive Idea in Concurrency Test

(1) Labeling Scheme

(2) Set operation

Thread representation (fid, tid)





Two sets:

Parallel set PS(f) ={(pf, ptid) | all threads with tid >= ptid in function pf is parallel with the running thread of f}

Children set CS(f)={fid | fid is the descendant of current function }

Operations:

Spawn: Thread Tx of function Fi spawn function Fj

Operations on child Fj

(1) $PS(Fj) = PS(Fi) \cup (Fi, Tx+1)$

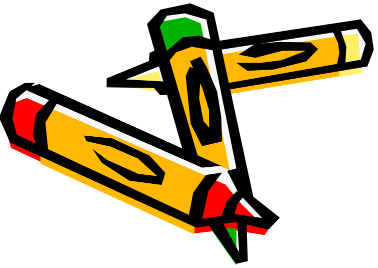
(2) $\mathrm{CS(Fj)} = \{\}$

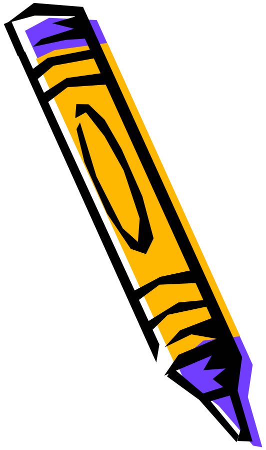
Operations on parent Fi

(1) $\forall Fp, \exists tid, (Fp, tid) \in PS(Fi) \rightarrow PS(Fp) = PS(Fp) \cup (Fj, 1)$

(2) $PS(Fi) = PS(Fi) \cup (Fj, 1)$

(3) $CS(Fi) = PS(Fi) \cup \{Fj\}$





Sync: Function Fi executes sync

PS(Fi)=PS(Fi)-CS(Fi)

Return: Function Fj returns to Function Fi

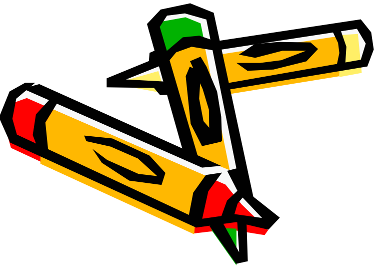
CS(Fi) = CS(Fi) + CS(Fj)

Release PS(Fj) and CS(Fj)

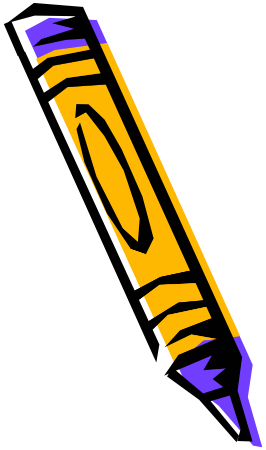
Concurrency Test:

Check if (fx, tx) is parallel with the current running thread (fc, tc):

$$\exists tid, (fx, tid) \in PS(fc) \wedge tid \geq tx \rightarrow (fx, tx) \text{ is parallel with (fc, tc)}$$

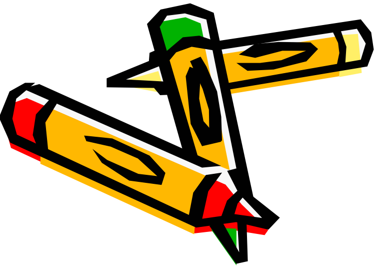


Primitive Idea for Access History



- Serial
  - read(l), write(l)
- Simplest parallel program without nested parallelism
  - Two parallel read records, one write record
- Language structure like Cilk
  - read: max level of parallelism, one write record

- Q: Is it possible to keep only two read records for each shared location in Cilk Parallel Nondeterminator?
  - Keep two parallel read records with highest level of LCA in parent child spawn tree.



Thank you very much!