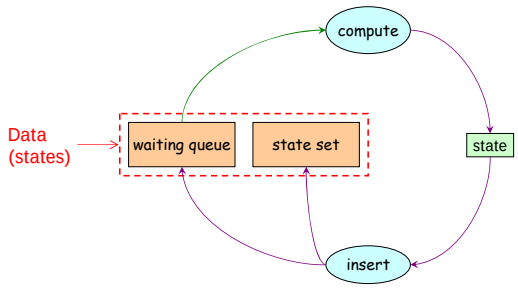
Parallelizing the Model-Checker

Alexandre David
1.2.05

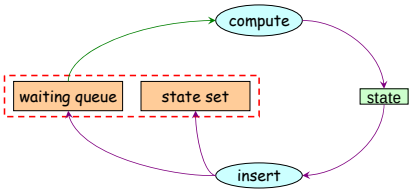
Basic Problems

- Where are the data?
  - Find the dataflow – data & functions.
- Which computations can be done in parallel?
- Identify critical sections.
- What data can be shared?
- How to solve load balancing?
- How to detect termination?

Simplified Dataflow

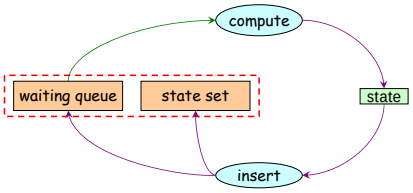


Which Computations in Parallel?



- All functions.
- Critical sections:
  - read & write to shared data.

Shared Data



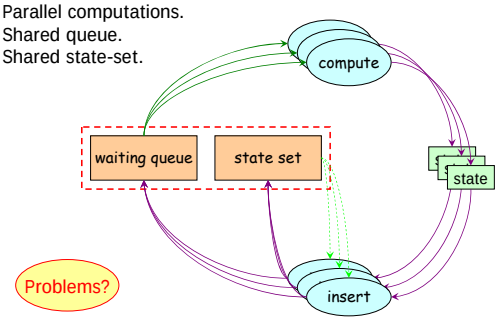
- Queue & state-set.
  - Evenly distributed among processes.
  - Load balancing through (universal) hash. Owner computes rule.

Termination Issue

- Detect that all processes are idle.
  - If process A is idle but B is working: no.
    - If B sends something to A and then becomes idle: no.
  - All processes idle and no data in transit: yes.
- Barrier protocol – principle:
  - Processes block on empty queues,
  - the last process detects termination.
  - Race condition issues
    - pthreads: condition synchronization.
    - MPI: distributed token based protocol.

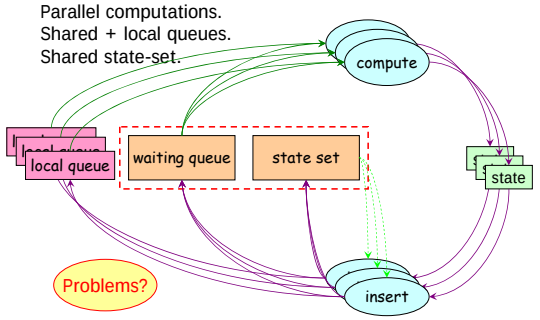
PThreads – 1

Parallel computations.
Shared queue.
Shared state-set.



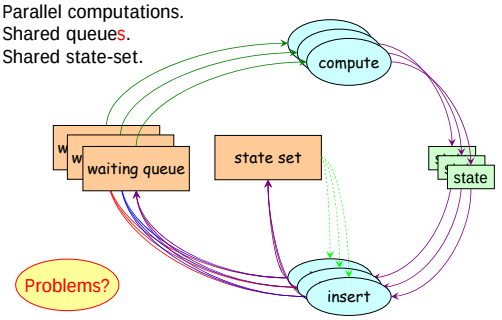
PThreads – 2

Parallel computations.
Shared + local queues.
Shared state-set.



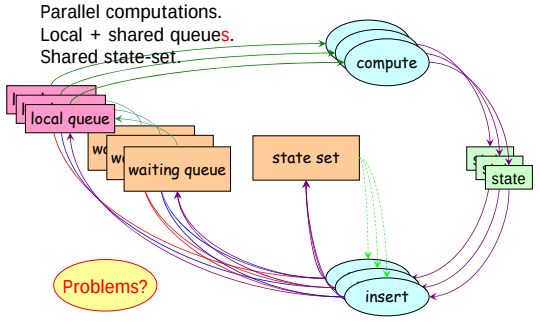
PThreads – 3

Parallel computations.
Shared queues.
Shared state-set.



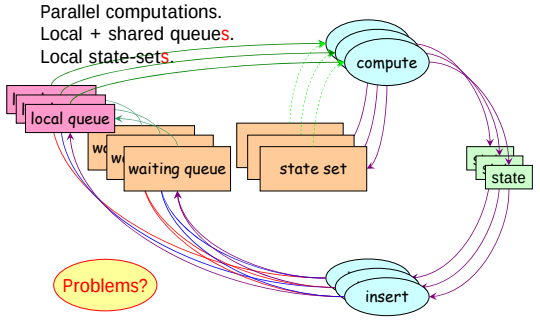
PThreads – 4

Parallel computations.
Local + shared queues.
Shared state-set.



PThreads – 5

Parallel computations.
Local + shared queues.
Local state-sets.



Issues

- Contention.
- False sharing.
  - Data.
  - Locks!!!
    All the threads will want to lock all the locks.
- Detect termination! (overhead)
- Solutions: tryLock, lock on hash entries.