

Systemes concurrents – Conclusion

Philippe Quéinnec

27 septembre 2019



Programmation parallèle

- souvent utile
- parfois indispensable
- fragile et complexe
- souvent difficile
- amusant

Deux aspects

- le parallélisme
- la synchronisation



Approches traditionnelles

- création explicite d'activités
- synchronisation explicite
- mécanismes classiques (verrou d'exclusion mutuelle, sémaphore, moniteur)
- raisonnement connues
- schémas classiques (producteurs/consommateurs, lecteurs/rédacteurs)

Exemples : Java Thread, C POSIX Threads



Approches modernes

- création implicite d'activités
- synchronisation implicite
- schémas classiques (fork-join)
- ne résolvent pas tous les problèmes

Exemple : OpenMP, interface Java Executor (pool de threads...), bibliothèques avancées



Approches d'avenir (?)

- Création implicite et explicite d'activités
- Synchronisation implicite
 - mémoire transactionnelle
 - bibliothèque de structures de données non bloquantes
- Absence d'effets de bord :
 - langages fonctionnels (Haskell)
 - parallélisation paresseuse
 - programmation événementielle ou dataflow

