Collaborative Parallelization Framework

Sotiris Apostolakis, Greg Chan,
Ziyang Xu, Benjamin Huang, and
David I. August

Liberty Research Group
Princeton University

Automatic parallelization is great …

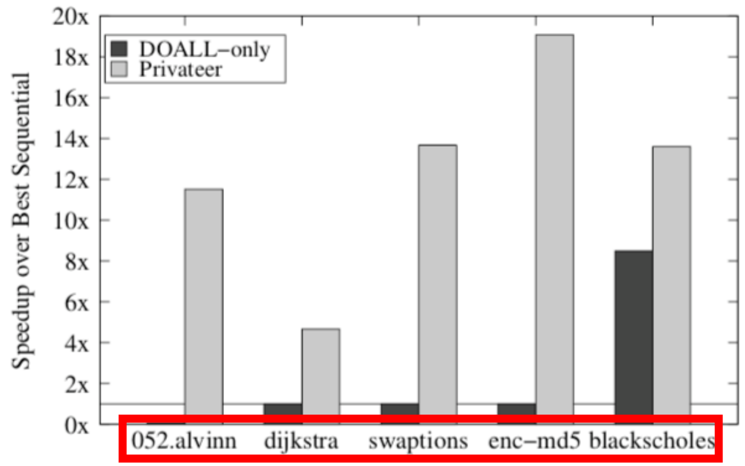


Figure 7: Enabling effect of Privateer at 24 worker processes. [1]

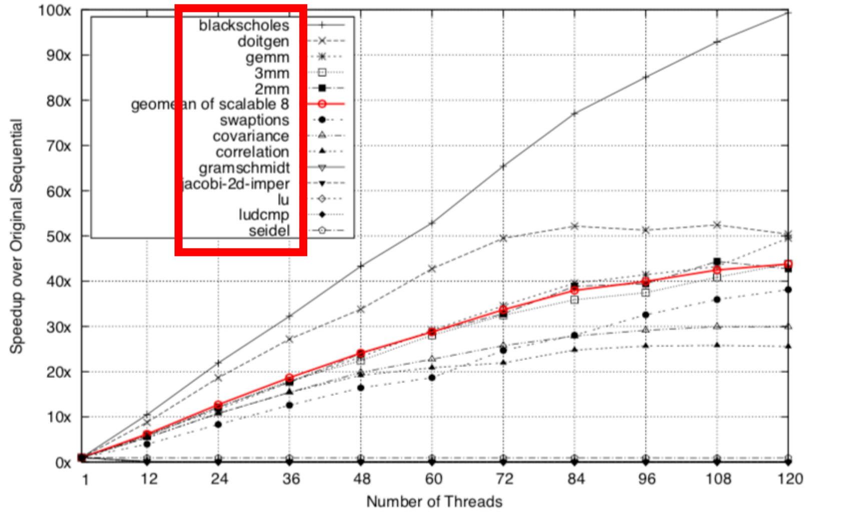


Figure 10: Overall speedup (Benchmarks in the legend are ordered from highest to lowest speedup) [2]

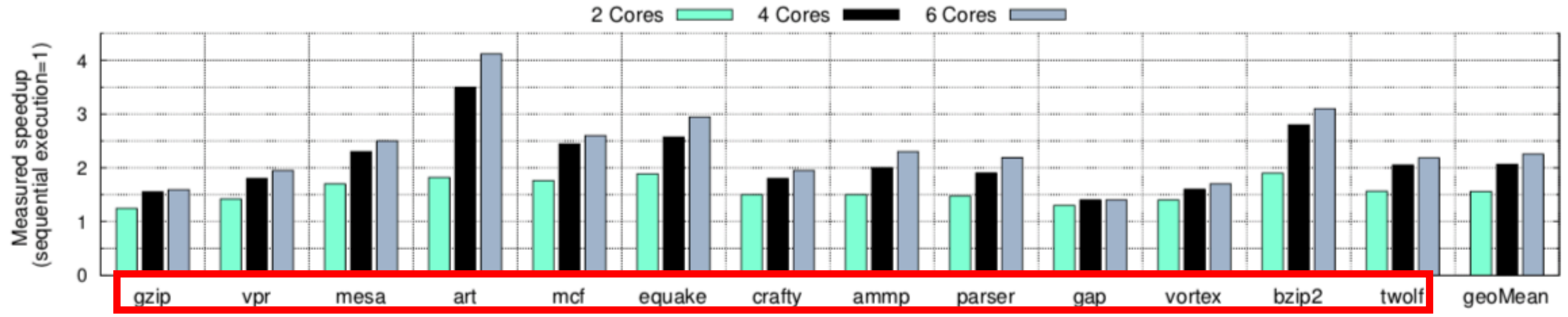
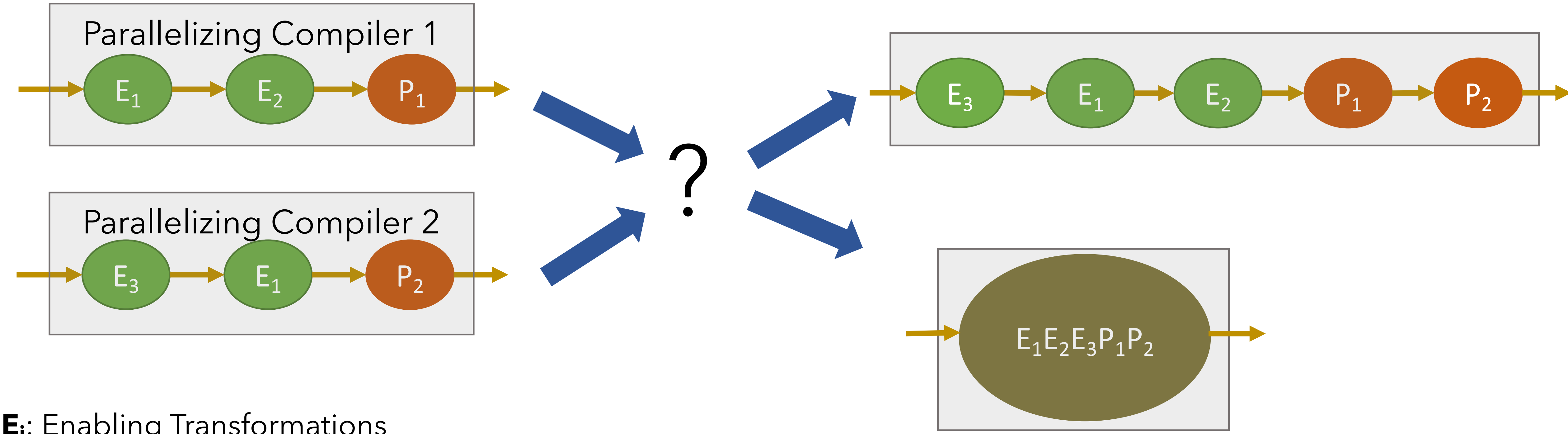


Figure 9: Speedups achieved by HELIX on a real system [3]

… when it works

[1] N. P. Johnson, H. Kim, P. Prabhu, A. Zaks, and D. I. August. Speculative separation for privatization and reductions. PLDI 2012.
[2] H. Kim, N. P. Johnson, J. W. Lee, S. A. Mahlke, and D. I. August. Automatic speculative doall for clusters. CGO 2012
[3] S. Campanoni, T. Jones, G. Holloway, V. J. Reddi, G.-Y. Wei, and D. Brooks. Helix: automatic parallelization of irregular programs for chip multiprocessing. CGO 2012

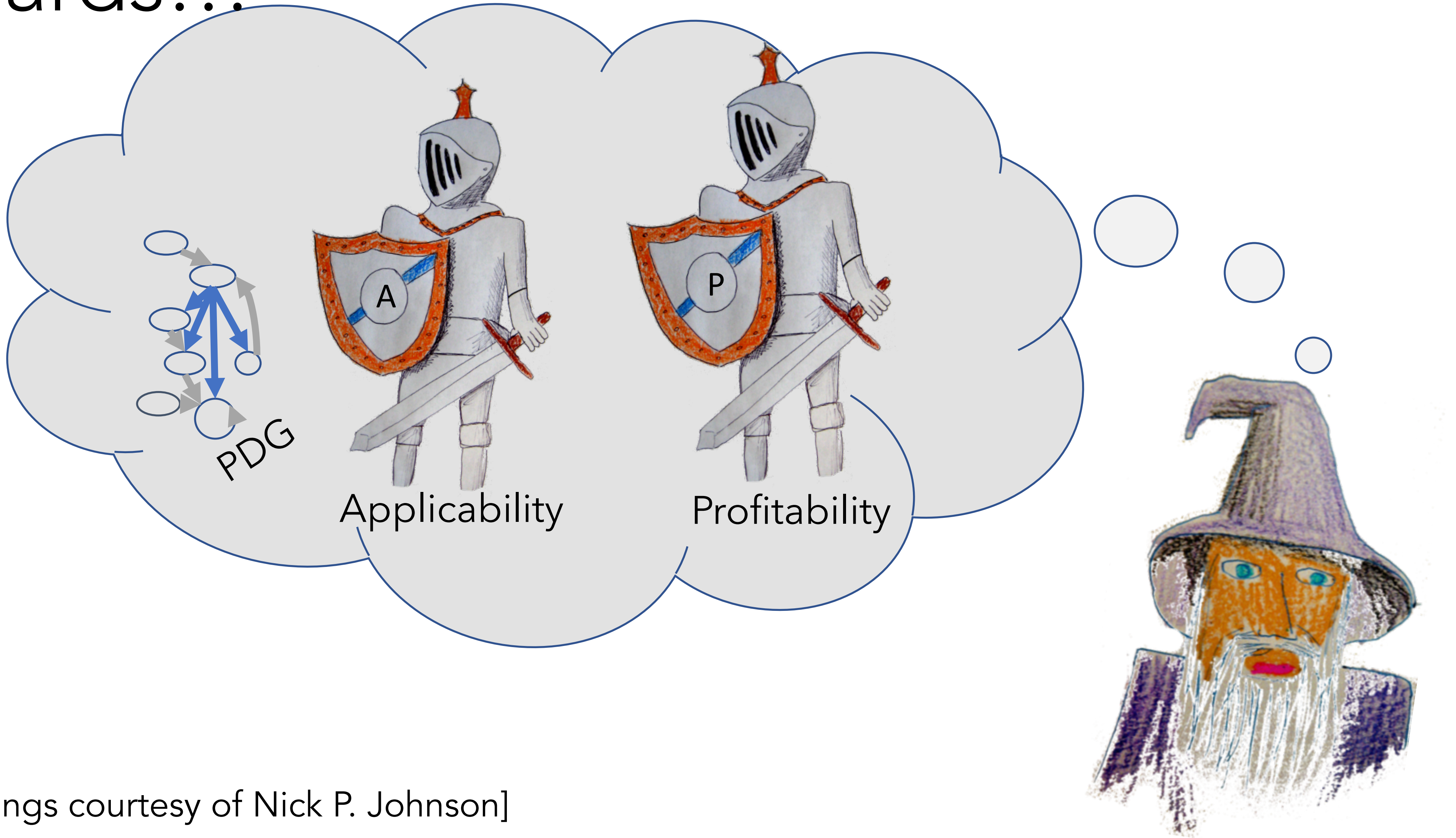
How to compose?



E_i: Enabling Transformations
(e.g., Memory Speculation)

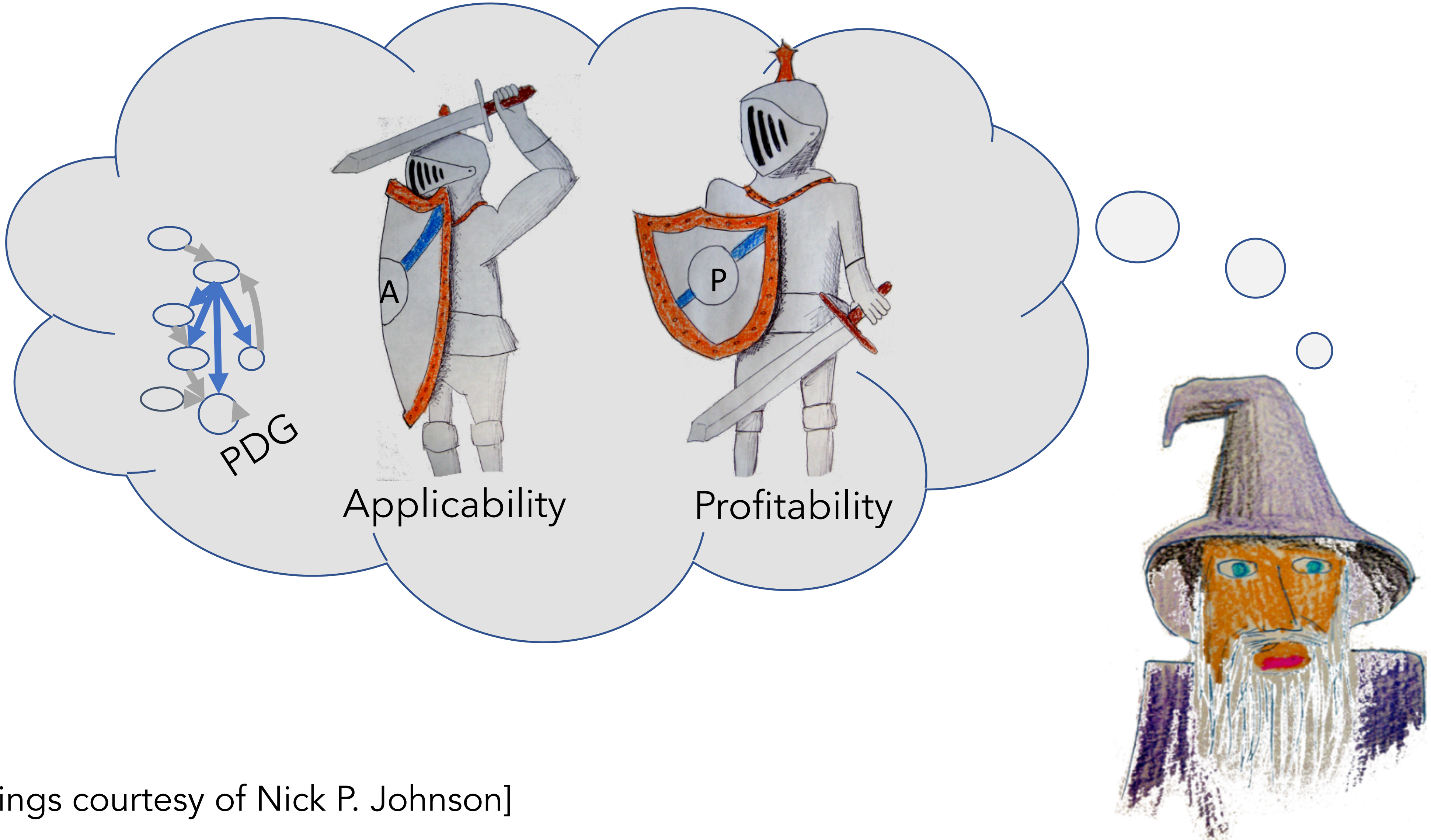
P_i: Parallelization Techniques
(e.g., DOALL, PS-DSWP)

Every transformation is protected by two guards…



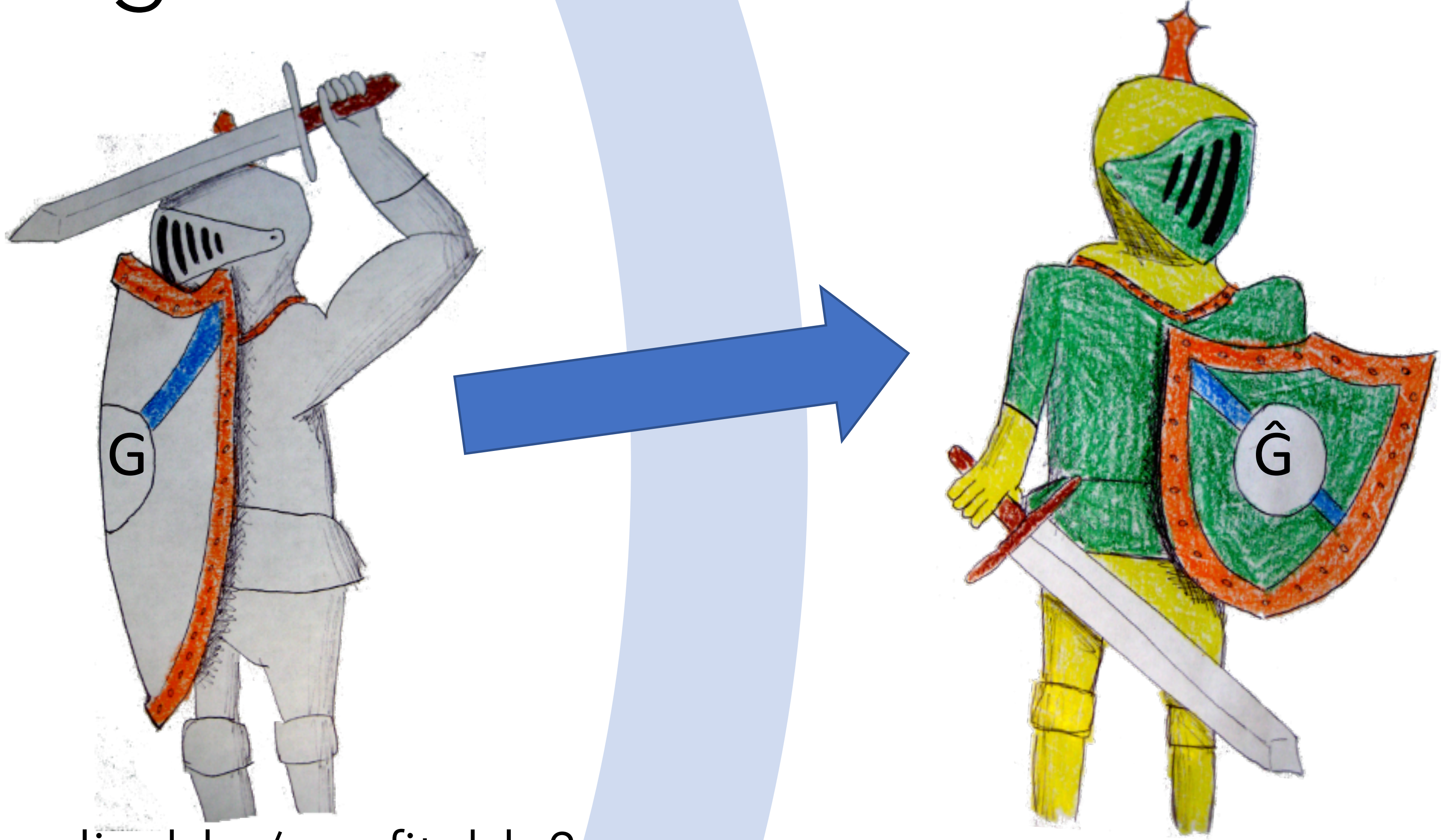
[Drawings courtesy of Nick P. Johnson]

Either may reject a program



[Drawings courtesy of Nick P. Johnson]

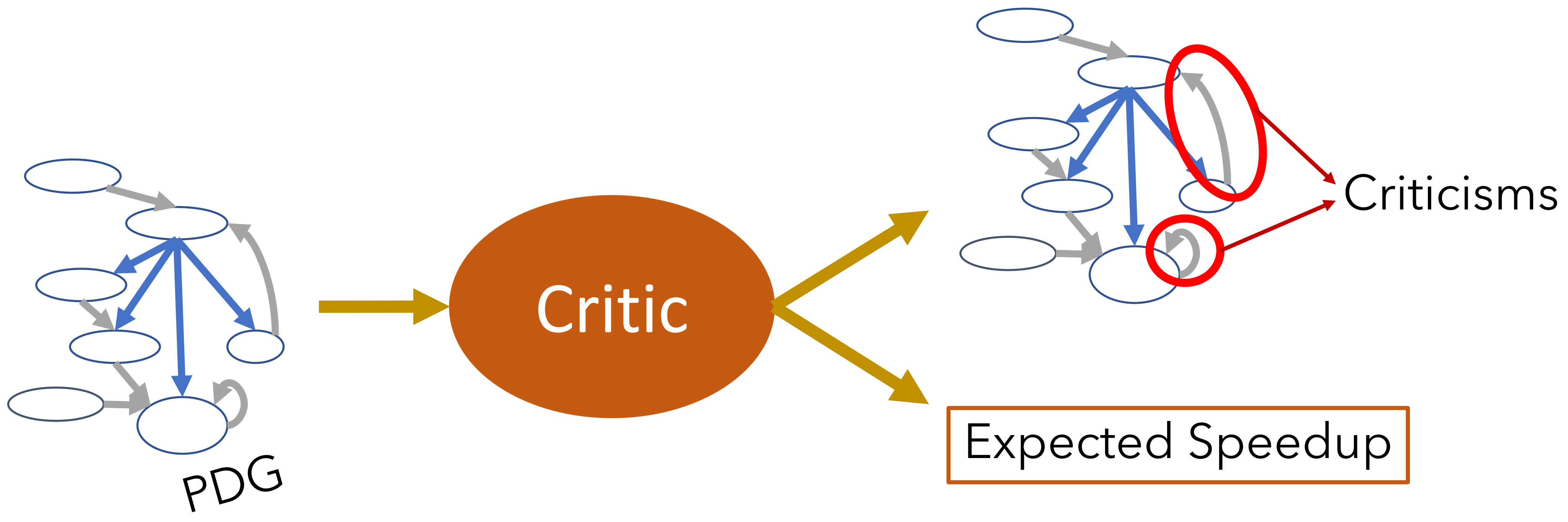
We can gather wisdom from them



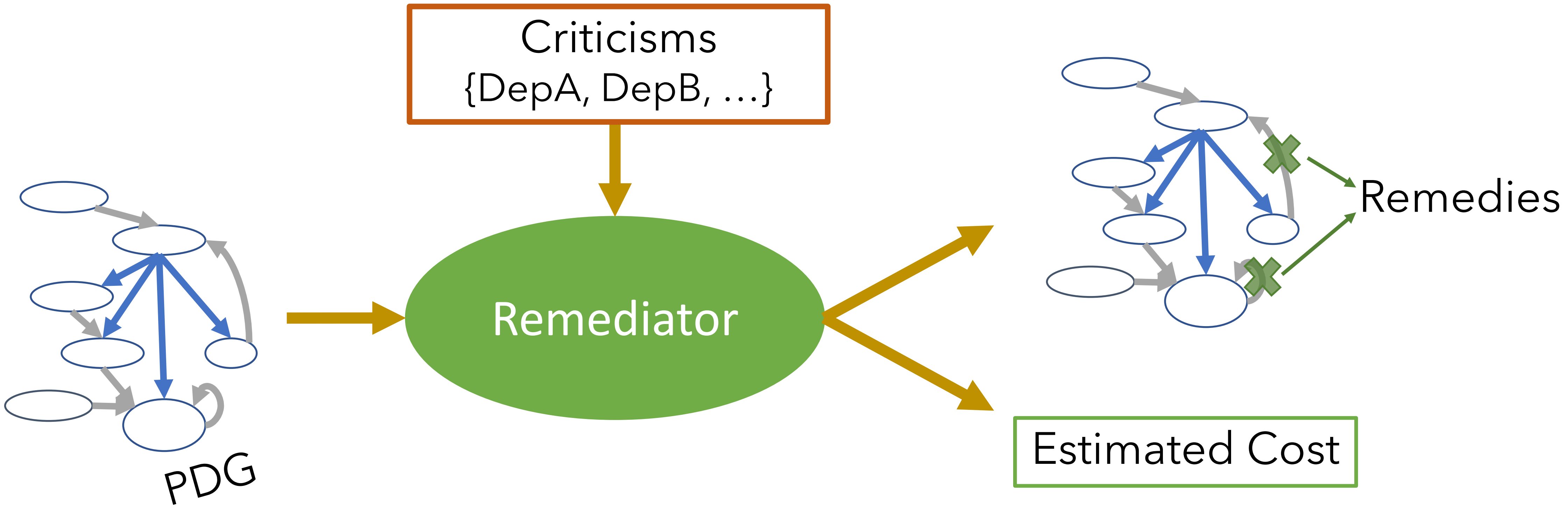
Is it applicable / profitable?

Why is it not applicable / profitable?

[Drawings courtesy of Nick P. Johnson]



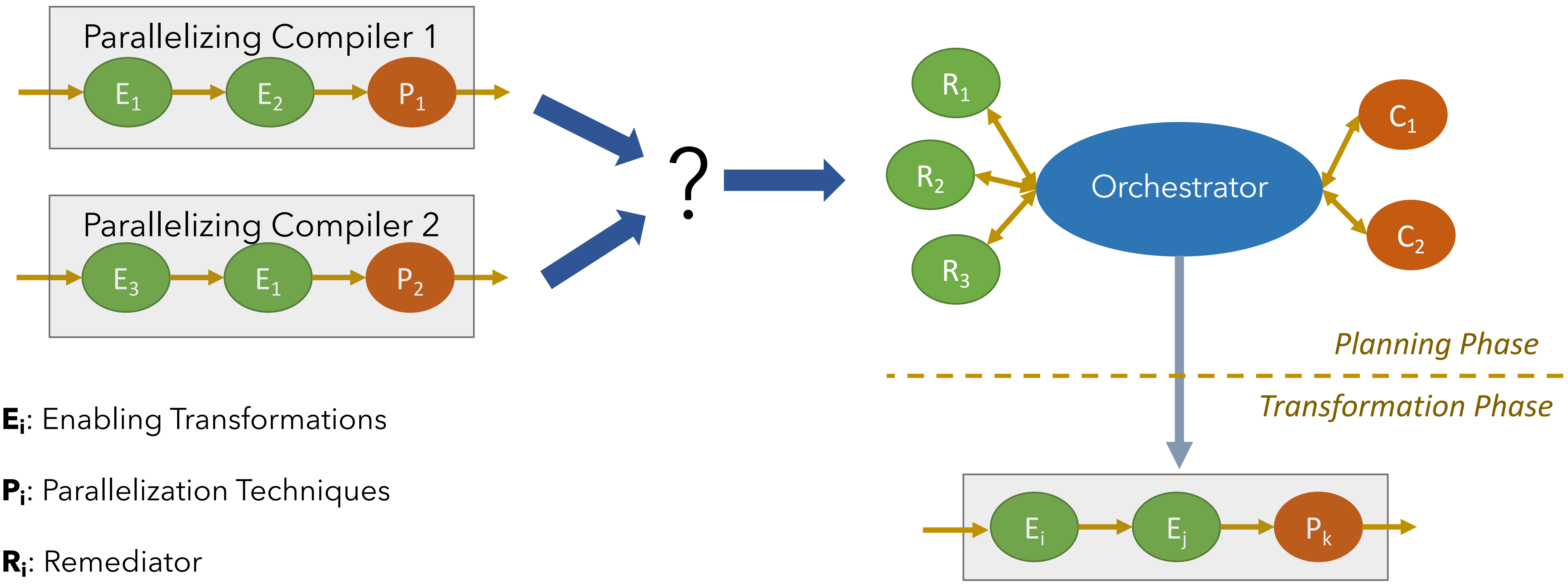
Critic: Answers why a parallelization technique
is not applicable/profitable?



Remediator:

- Uses applicability guard of enabling transformations.
- Ignores original profitability guard;
  the transformation is useful if a criticism is satisfied
- Do not apply the transformation but express its effect

Collaborative Parallelization Framework



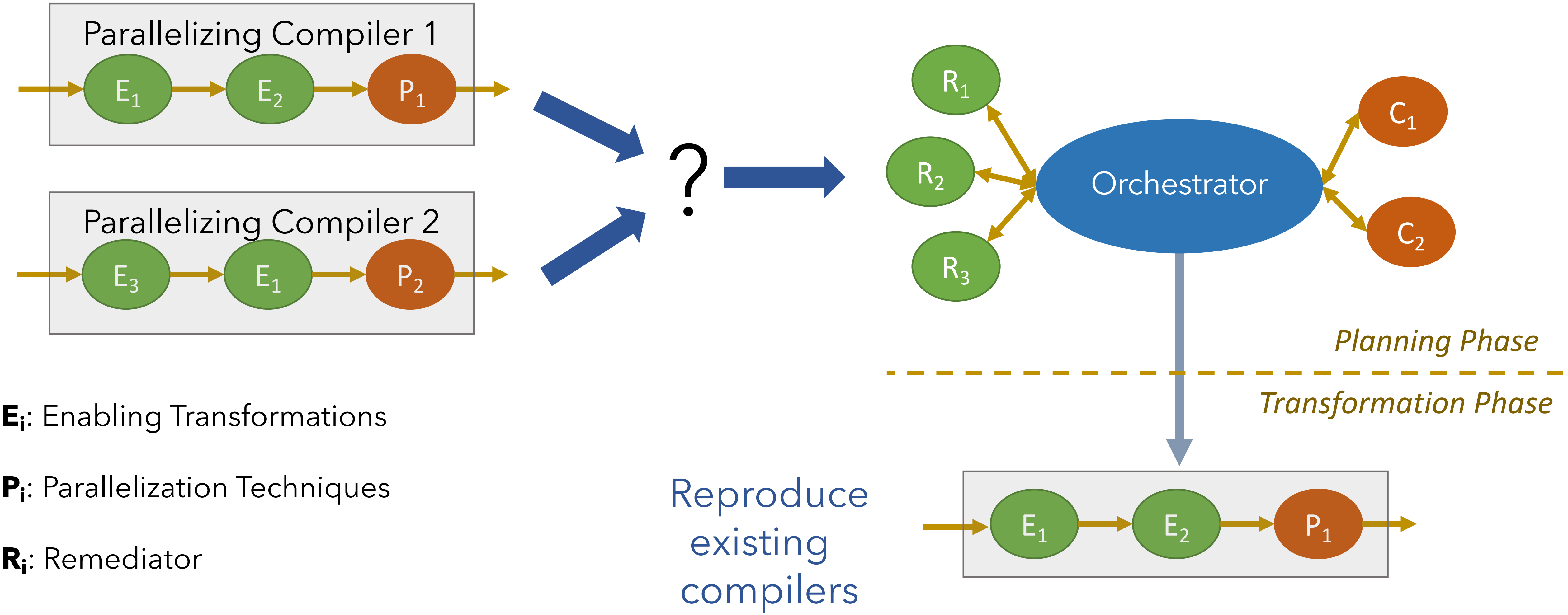
E_i: Enabling Transformations

P_i: Parallelization Techniques

R_i: Remediator

C_i: Critic

Collaborative Parallelization Framework



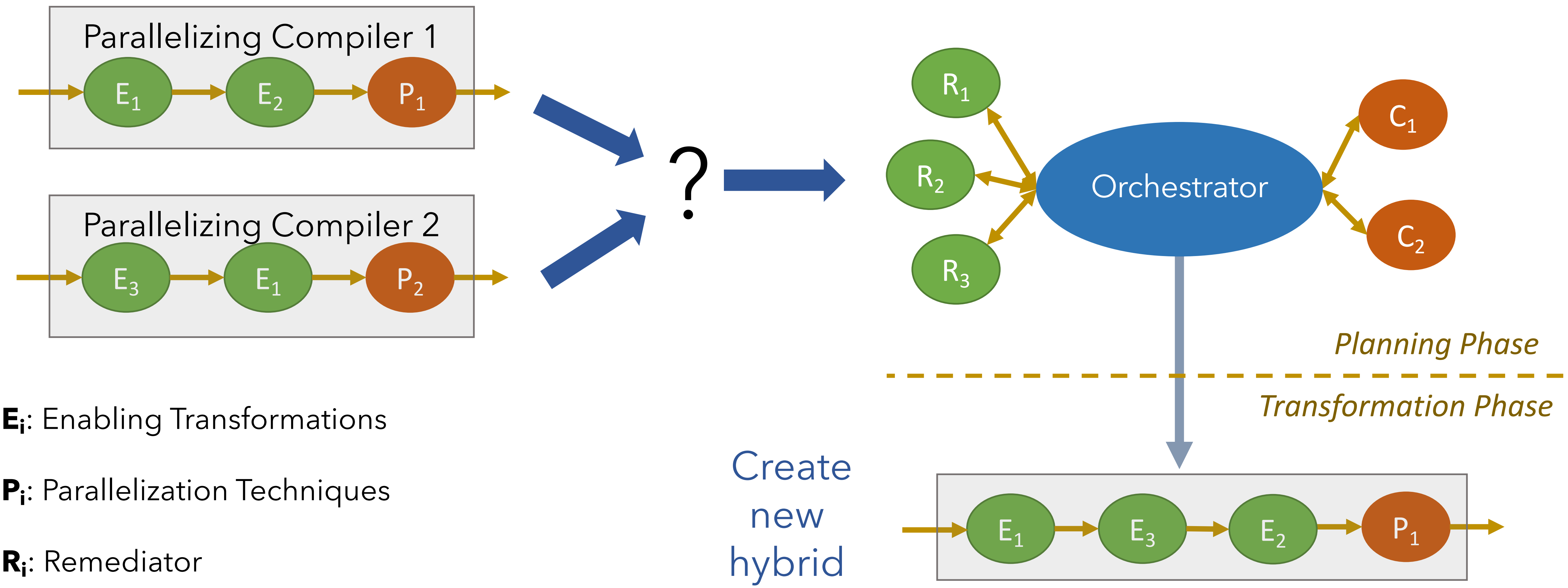
E_i: Enabling Transformations

P_i: Parallelization Techniques

R_i: Remediator

C_i: Critic

Collaborative Parallelization Framework



Create new hybrid versions

E_i: Enabling Transformations

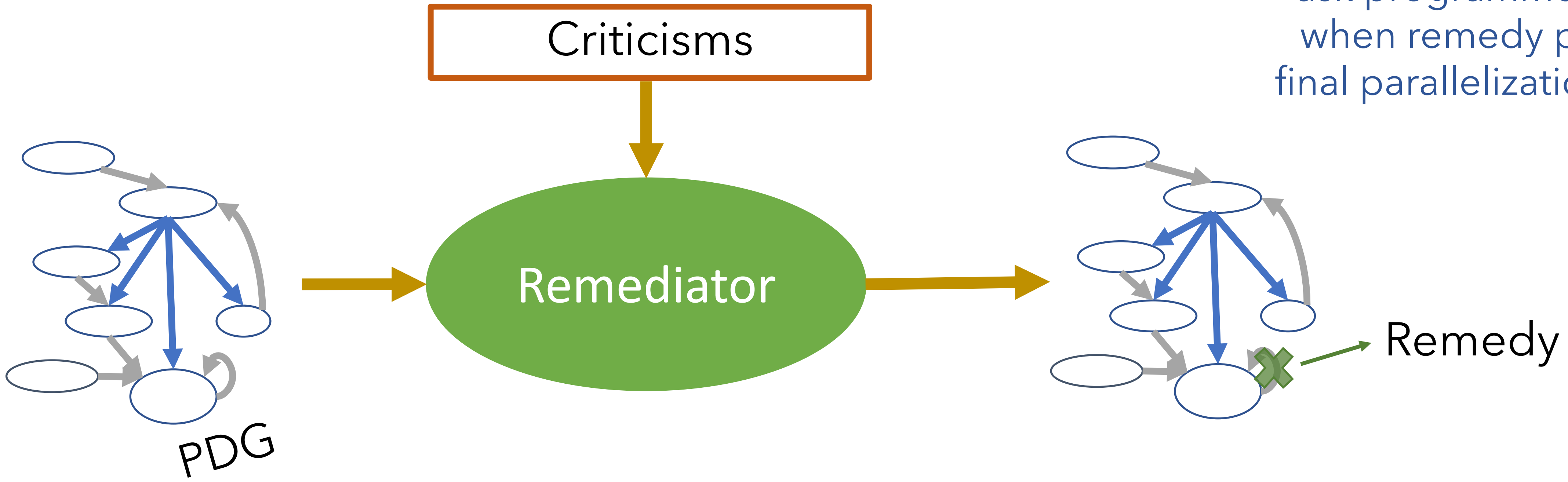
P_i: Parallelization Techniques

R_i: Remediator

C_i: Critic

Programmer as a Remediator

Assume expected answer, ask programmer only when remedy part of final parallelization plan



- Programmer Remediator:
  - Applicability: Checks if criticisms can be translated to high-level yes/no questions
  - Profitability: High-probability assumptions

Conclusion

- Combine compiler advancements on automatic parallelization into an unified compiler framework
  - Better automated and robust parallelization decision process
    - Transformations communicate through criticisms and remedies
    - New supervisory compiler component, called The Orchestrator
- Modularity
  - Easy to add new transformations to the system
  - Every transformation developed independently
- Minimize programmer involvement
  - Seek help from the programmer only when necessary

Thank you

Questions?