Live traces

- So far, we can generate unacceptable traces:
  - Finite:
    - BuyOffer(1), SellOffer(1)
  - Infinite:
    - BuyOffer(1), SellOffer(1), BuyOffer(1), SellOffer(1), … (But(1)/Sell(1) never happen)
- Back to enumerating all acceptable traces?
- Why are such traces unacceptable?

Live traces

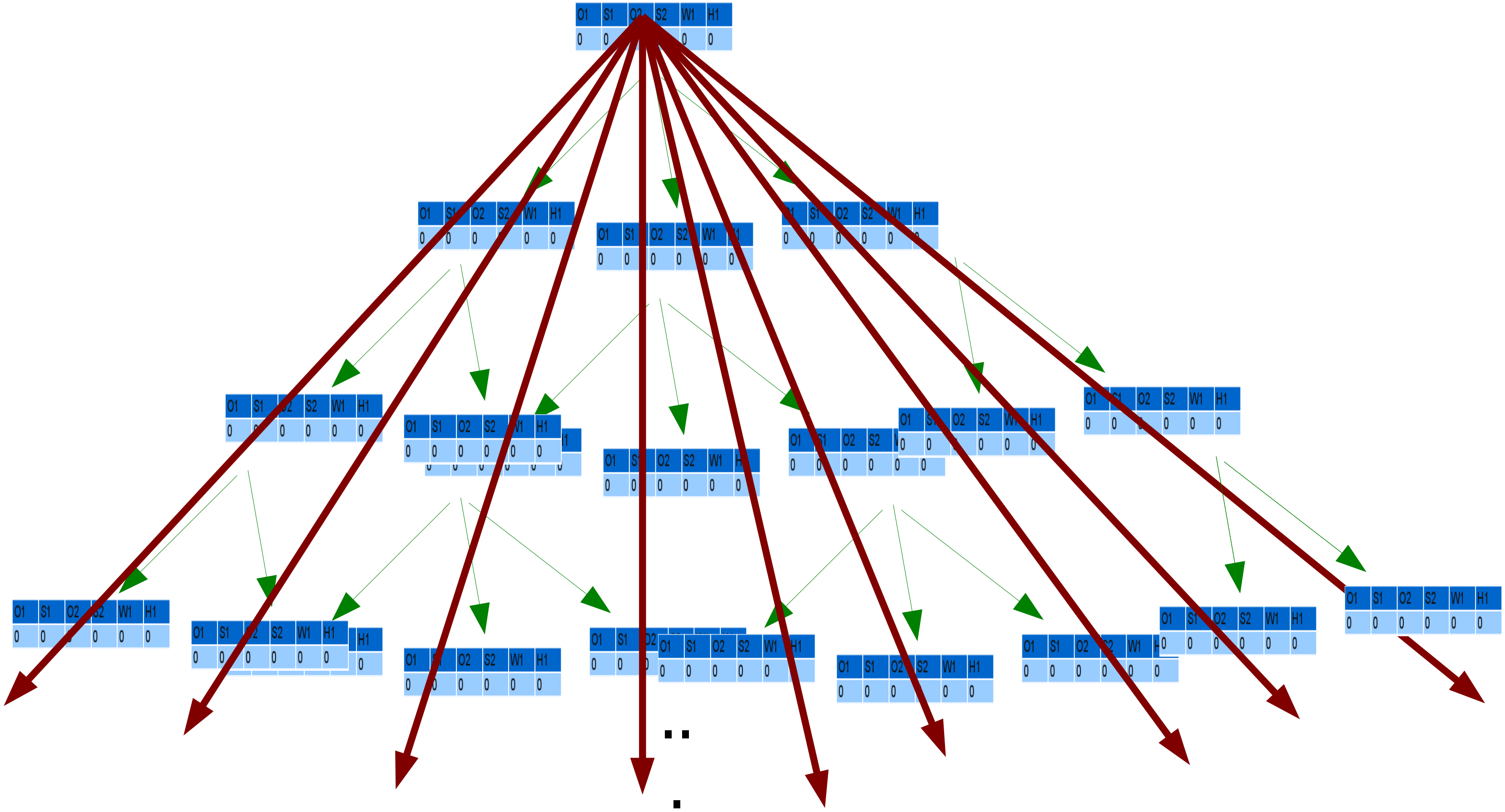
- Weak fair scheduling classes:
  - { Buy(i), for all i }
  - { Sell(i), for all i }
  - Don't care about BuyOffer(i) and SellOffer(i)
- A class cannot have transitions enabled forever without ever being taken

Liveness properties

- Does this specification meet the desired properties?
- Liveness property:
  - If there are sellers and buyers for at least k items, eventually k items are sold and bought

    (i.e. $\Sigma O \geq k$ and $\Sigma W \geq k$ then eventually $\Sigma S \geq k$ and $\Sigma H \geq k$)
- Observation:
  - This is <u>not</u> a state invariant

Liveness properties

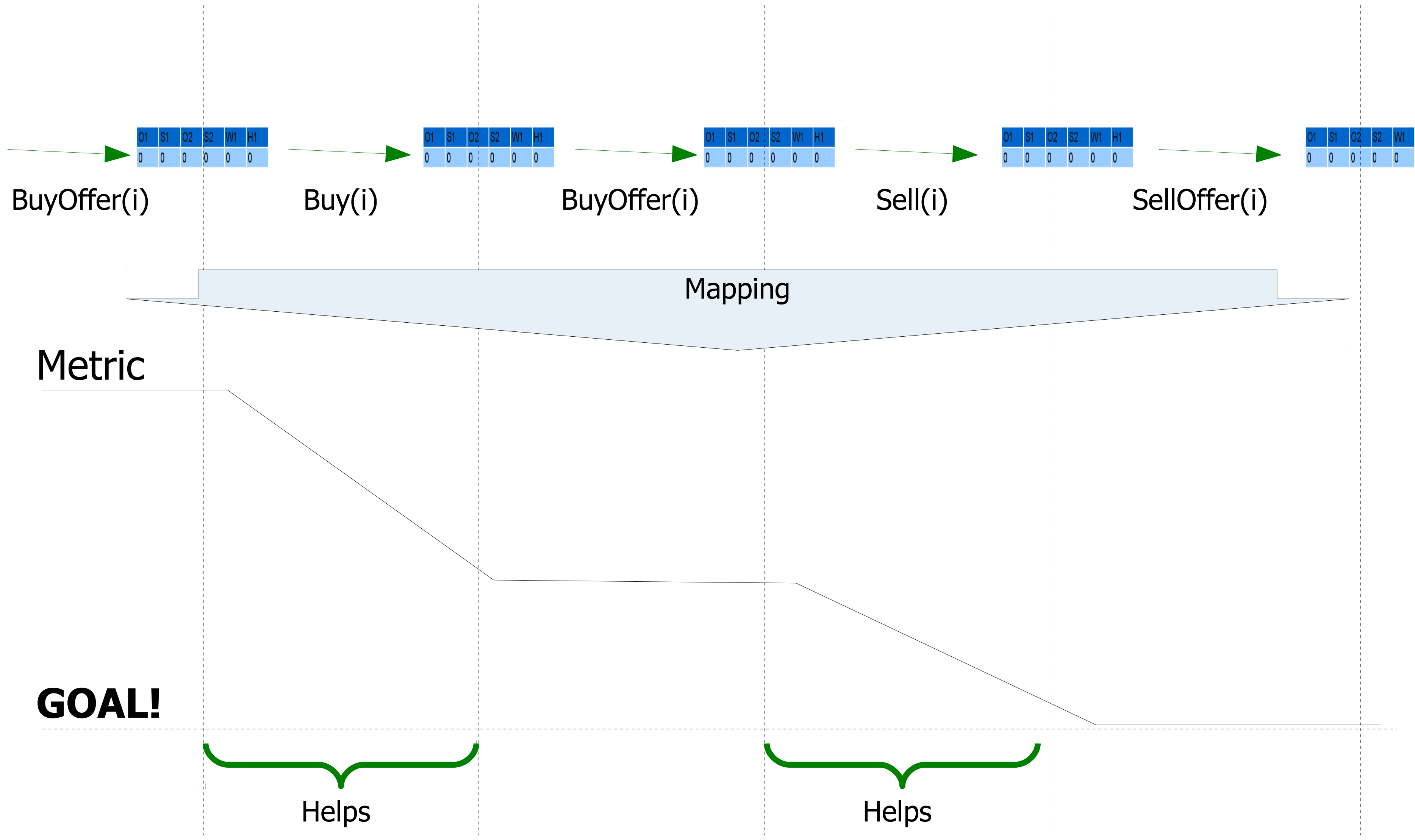


- Again, cannot evaluate all possible traces, but...

Liveness properties

- Define a “distance to goal” metric:
  - k-min(ΣS,k)+k-min(ΣH,k)
- Map each transition to the metric:
  - BuyOffer(i): don't care
  - SellOffer(i): don't care
  - Buy(i): helps, until k bought
  - Sell(i): helps, until k sold

Liveness properties



Liveness properties

- Until the goal is met, at least one weakly fair helper transition is enabled
  - State invariant!
- No transition makes the metric grow
  - State invariant!
- Can easily prove both...

Live traces

- Unfortunately, we can still generate unacceptable traces:
  - BuyOffer(1), SellOffer(1), BuyOffer(2), Sell(1), Buy(2), … (Buy(1) never happens)
- Back to enumerating all acceptable traces?
- Why are such traces unacceptable?

Live traces

- Strong fair scheduling classes:
  - { Buy(i) } for all i
  - { Sell(i) } for all i
  - Don't care about BuyOffer(i) and SellOffer(i)
- A class cannot have transitions enabled infinitely often without ever being taken

Liveness properties

- Does this specification meet the desired properties?
- Liveness property:
  - If multiple buyers are competing, make sure no one is left behind

    (i.e. $W_i \geq k$ and eventually $\Sigma O \geq l$, for any l, then eventually $H_i \geq k$)
- Observation:
  - This is <u>not</u> a state invariant

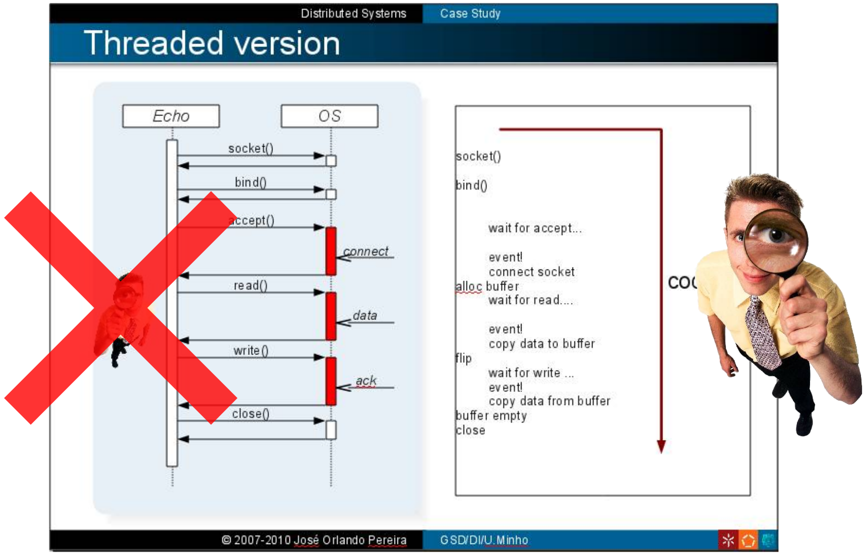
Liveness properties

- Until the goal is met, at least one strongly fair helper transition is eventually enabled
  - Liveness property (might require additional fairness assumptions...)
- No transition makes the metric grow
  - State invariant!
- Can now prove both

Conclusion

- Specification:
  - State machine
    +
  - Fairness classes
- Can prove both safety and liveness properties:
  - Effort proportional to # of transitions!

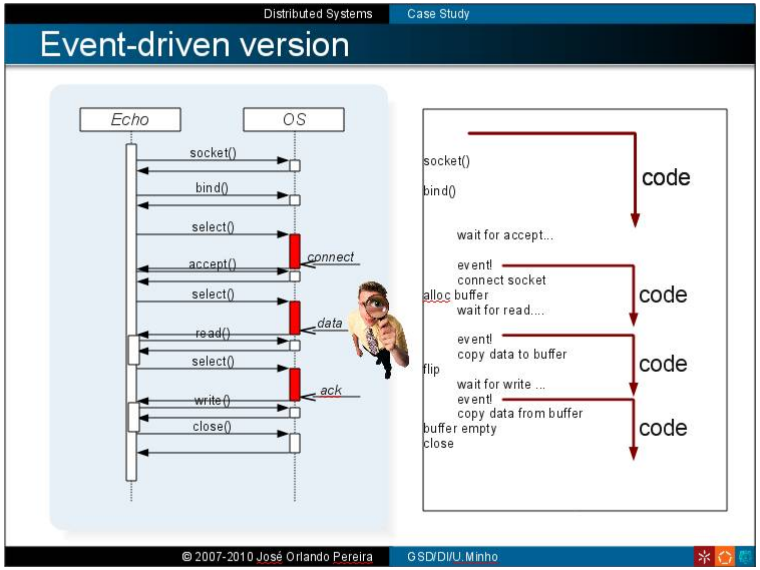
Consequences



Distributed Systems | Case Study

Threaded version

© 2007-2010 José Orlando Pereira | GSD/DI/U.Minho



Distributed Systems | Case Study

Event-driven version

© 2007-2010 José Orlando Pereira | GSD/DI/U.Minho

- Regardless of how code is written…
  - Always think in terms of transitions
- Consider the impact of each transition in:
  - Safety properties
  - “Metric to goal” for liveness

Consequences

- Trivially ensure weak fairness:
  - Round-robin for threads and processes
  - FIFO for mutexes/conditions
    - Use java.util.concurrent.*
  - Iterate over all events before going back to waiting on select()

Consequences

- Strong fairness is harder to ensure
- Usually, requires keeping explicit queues ordered by last service time