Policy Generation for Continuous-time Stochastic Domains with Concurrency

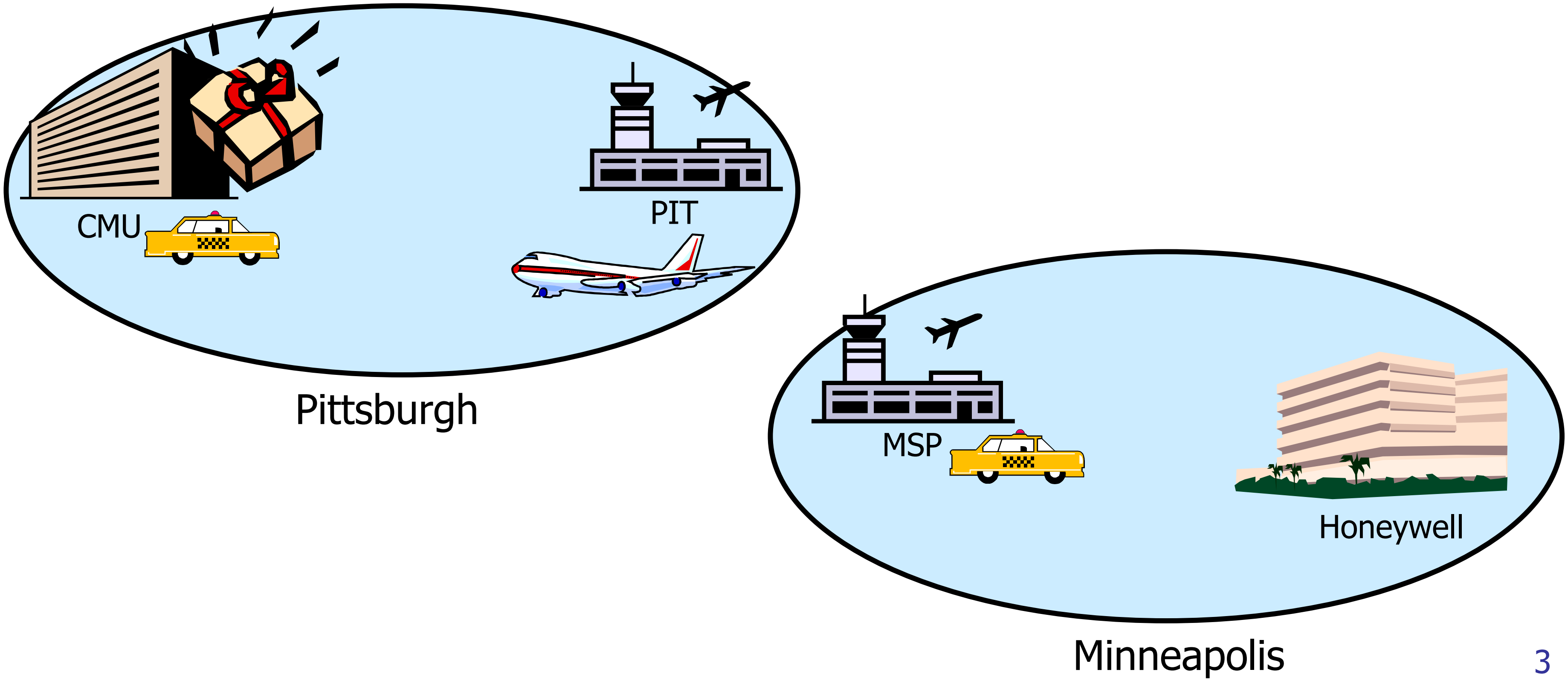
Håkan L. S. Younes Reid G. Simmons
Carnegie Mellon University

Introduction

- Policy generation for asynchronous stochastic systems
- Rich goal formalism
- policy generation and repair
  - Solve relaxed problem using deterministic temporal planner
  - Decision tree learning to generalize plan
  - Sample path analysis to guide repair

Motivating Example

- Deliver package from CMU to Honeywell



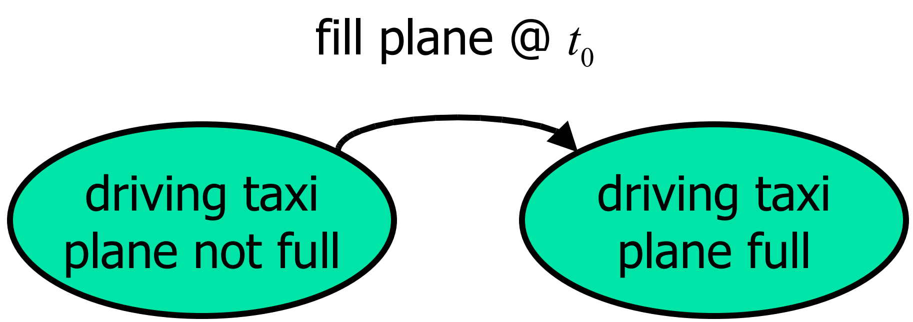
Elements of Uncertainty

- Uncertain duration of flight and taxi ride
- Plane can get full without reservation
- Taxi might not be at airport when arriving in Minneapolis
- Package can get lost at airports

Asynchronous events $\Rightarrow$ not semi-Markov

Asynchronous Events

- While the taxi is on its way to the airport, the plan may become full



Arrival time distribution: $F(t)$ $F(t|t > t_0)$

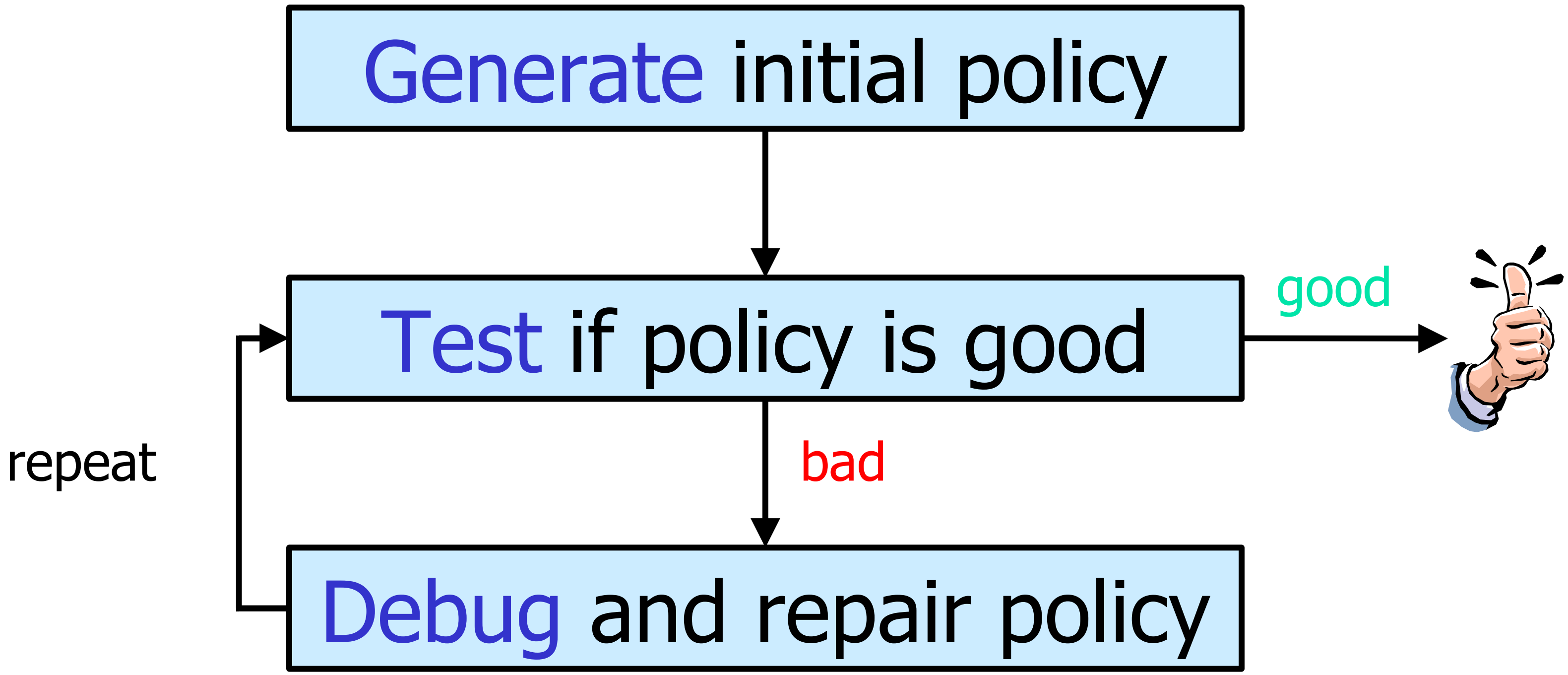
Rich Goal Formalism

- Goals specified as CSL formulae
  - $\phi ::= \mathrm{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{P}_{\geq\theta}(\phi\, \mathsf{U}^{\leq T}\, \phi)$
- Goal example:
  - "Probability is at least 0.9 that the package reaches Honeywell within 300 minutes without getting lost on the way"
    - $\mathsf{P}_{\geq 0.9}(\neg \mathit{lost}_{\mathrm{package}}\, \mathsf{U}^{\leq 300}\, \mathit{at}_{\mathrm{pkg,honeywell}})$

Problem Specification

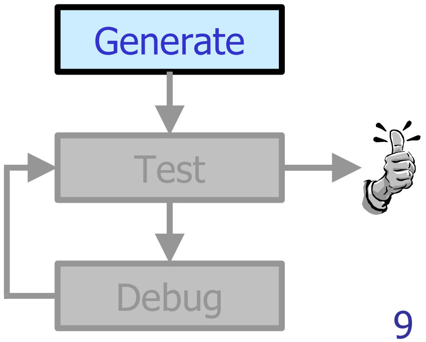
- Given:
  - Complex domain model
    - Stochastic discrete event system
  - Initial state
  - Probabilistic temporally extended goal
    - CSL formula
- Wanted:
  - Policy satisfying goal formula in initial state

Generate, Test and Debug [Simmons, AAAI-88]



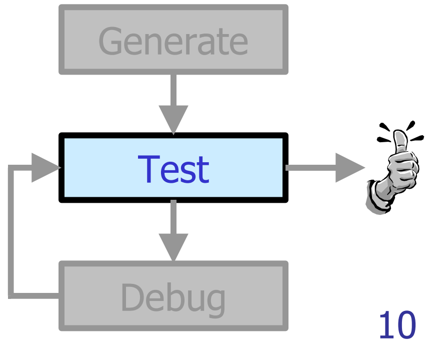
Generate

- Ways of generating initial policy
  - Generate policy for relaxed problem
  - Use existing policy for similar problem
  - Start with null policy
  - Start with random policy



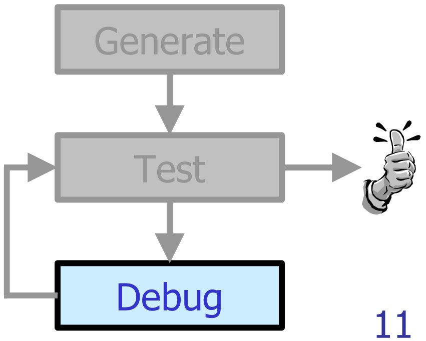
Test [Younes et al., ICAPS-03]

- Use discrete event simulation to generate sample execution paths
- Use acceptance sampling to verify probabilistic CSL goal conditions

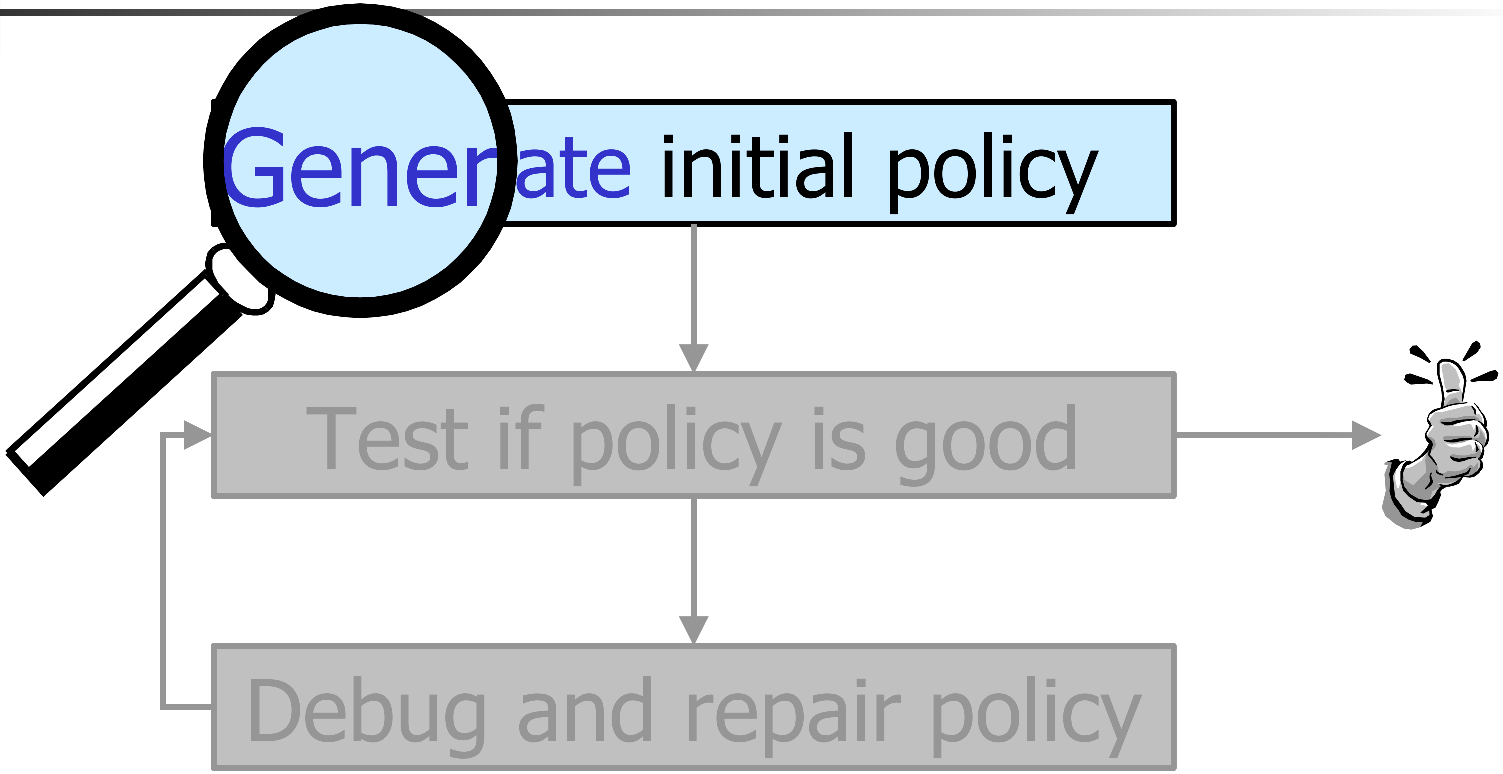


Debug

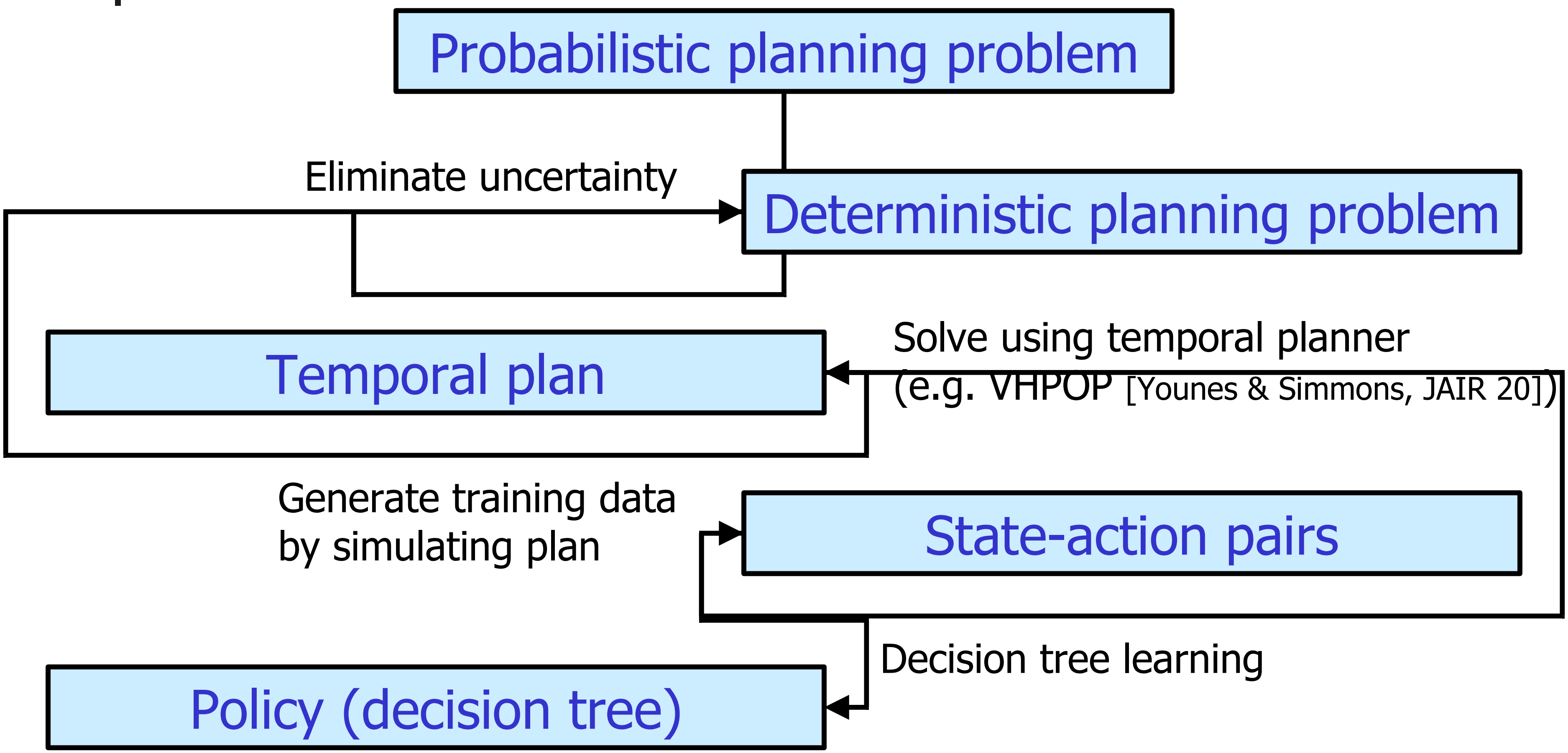
- Analyze sample paths generated in test step to find reasons for failure
- Change policy to reflect outcome of failure analysis



Closer Look at Generate Step



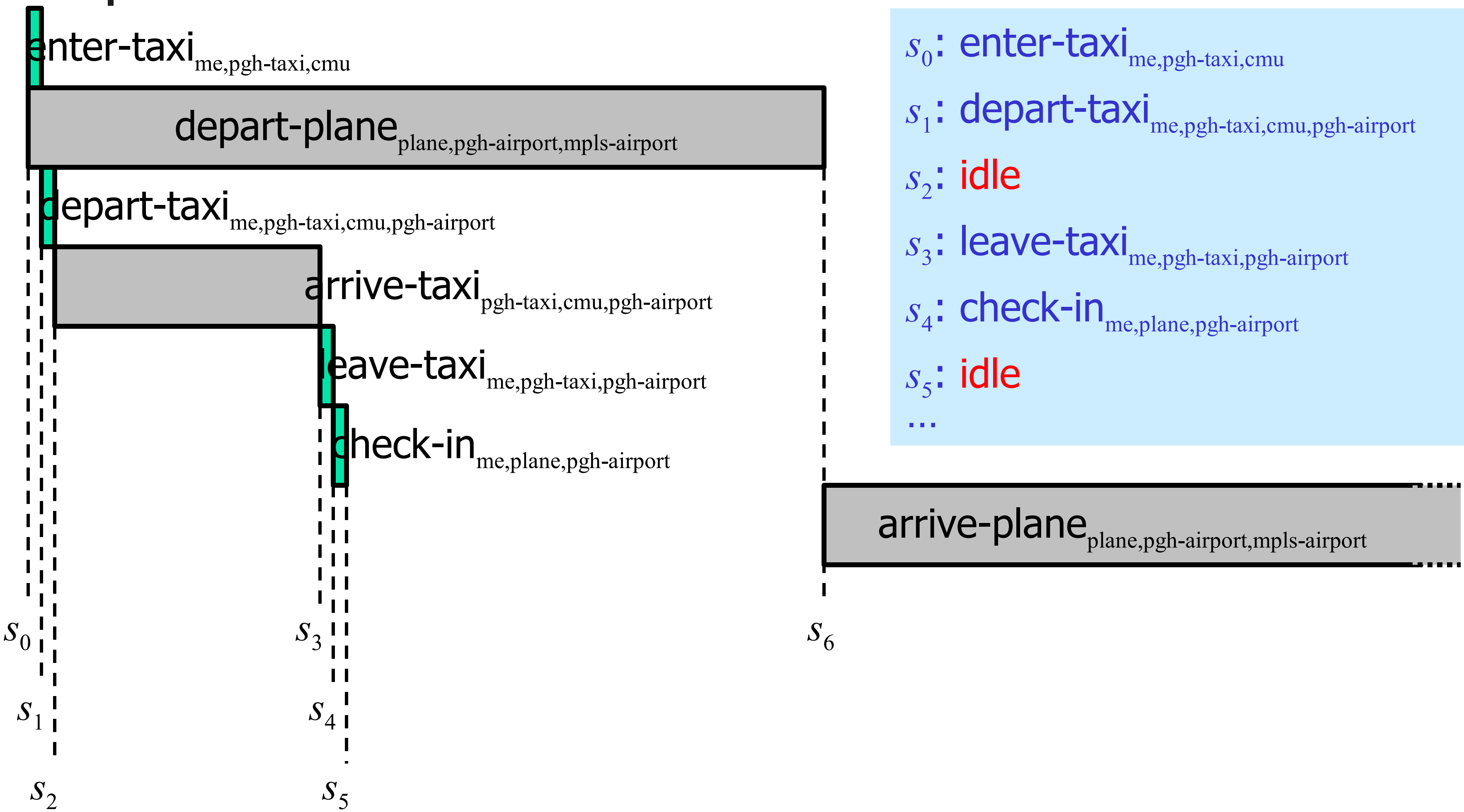
Policy Generation



Conversion to Deterministic Planning Problem

- Assume we can control nature:
  - Exogenous events are treated as actions
  - Actions with probabilistic effects are split into multiple deterministic actions
  - Trigger time distributions are turned into interval duration constraints
- Objective: Find some execution trace satisfying path formula $\varphi_1 \mathrm{U}^{\leq T} \varphi_2$ of probabilistic goal $\mathrm{P}_{\geq\theta}(\varphi_1 \mathrm{U}^{\leq T} \varphi_2)$

Generating Training Data



s_0: enter-taxi$_{\text{me,pgh-taxi,cmu}}$

s_1: depart-taxi$_{\text{me,pgh-taxi,cmu,pgh-airport}}$

s_2: idle

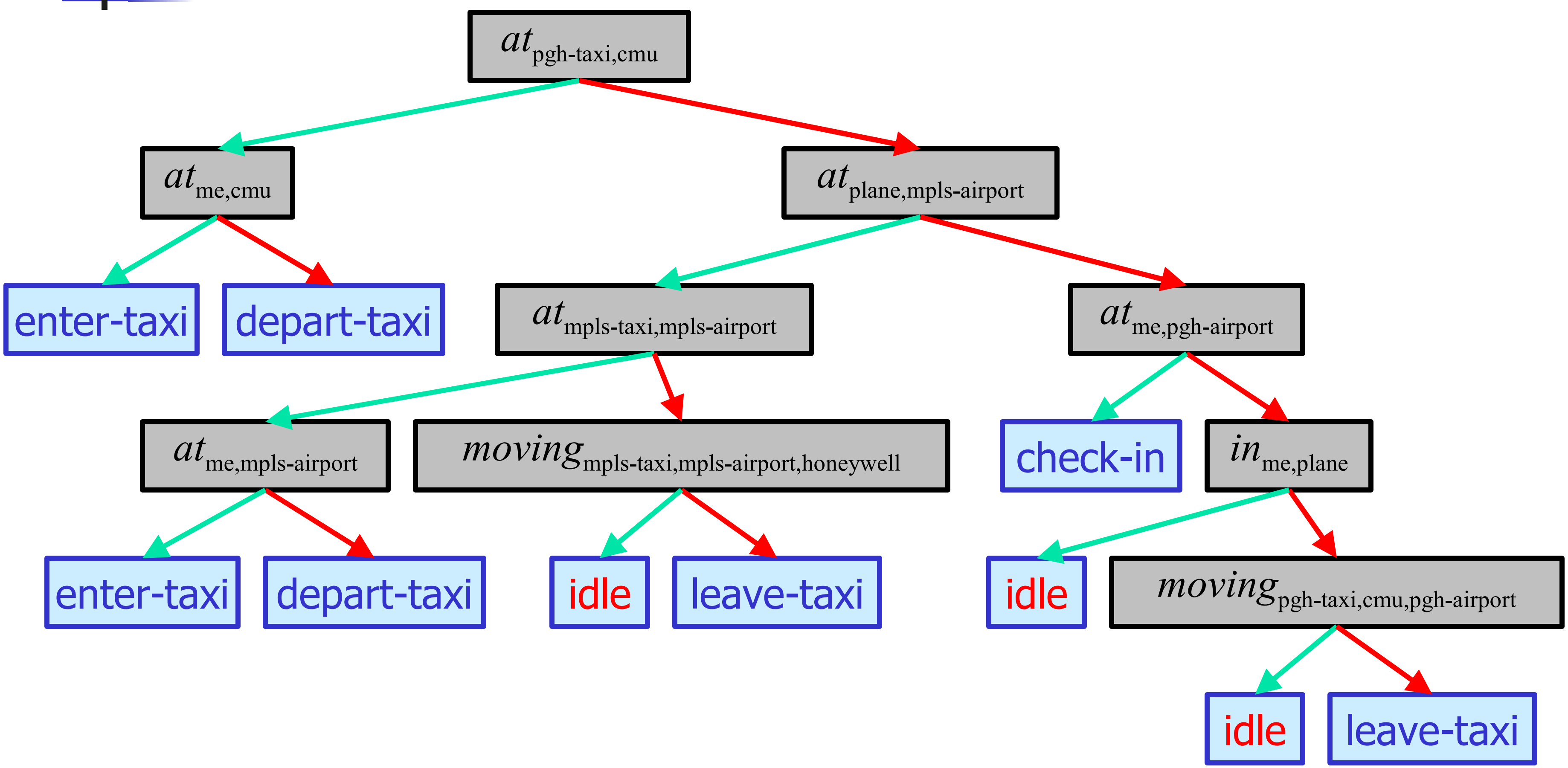
s_3: leave-taxi$_{\text{me,pgh-taxi,pgh-airport}}$

s_4: check-in$_{\text{me,plane,pgh-airport}}$

s_5: idle

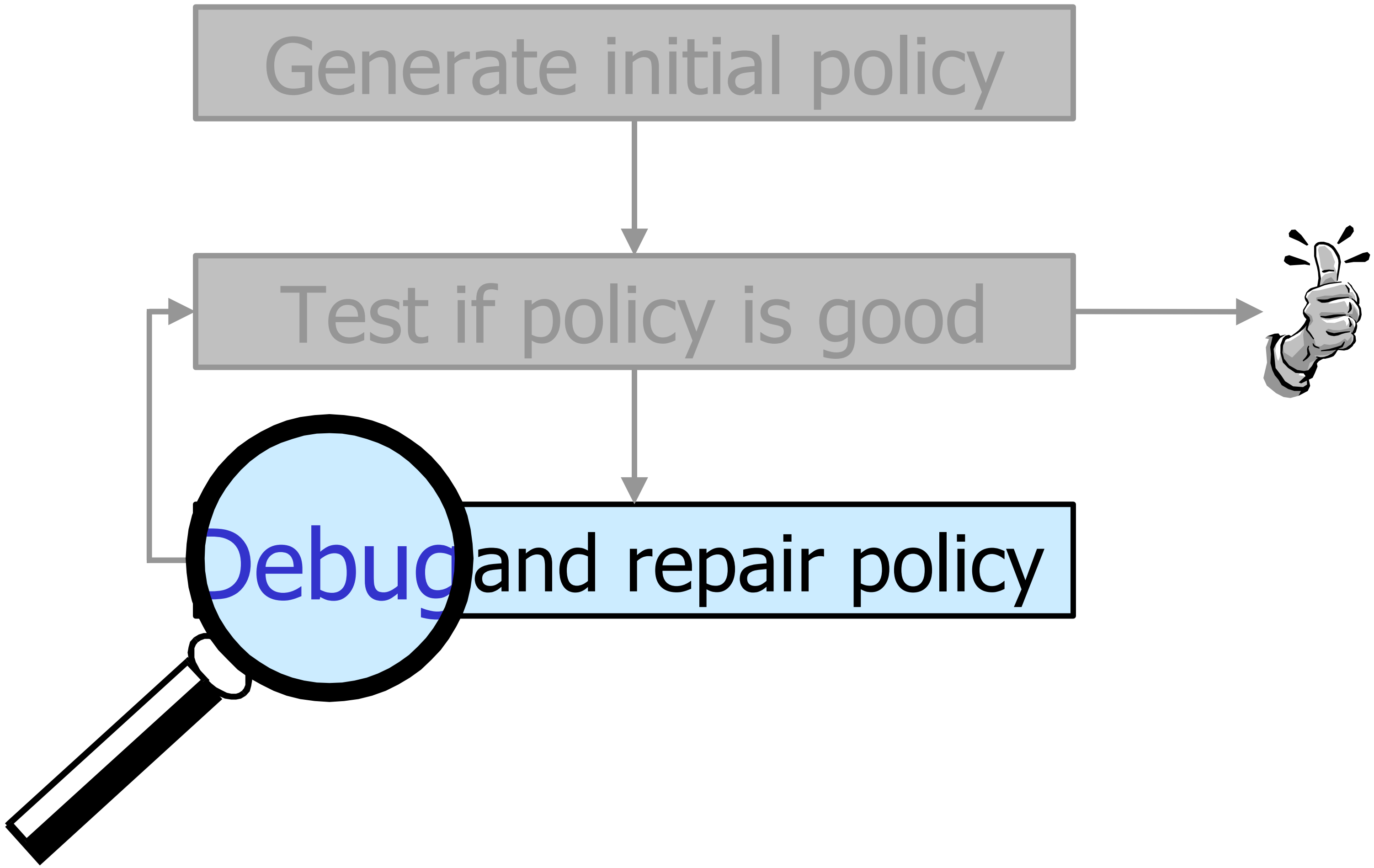
…

Policy Tree

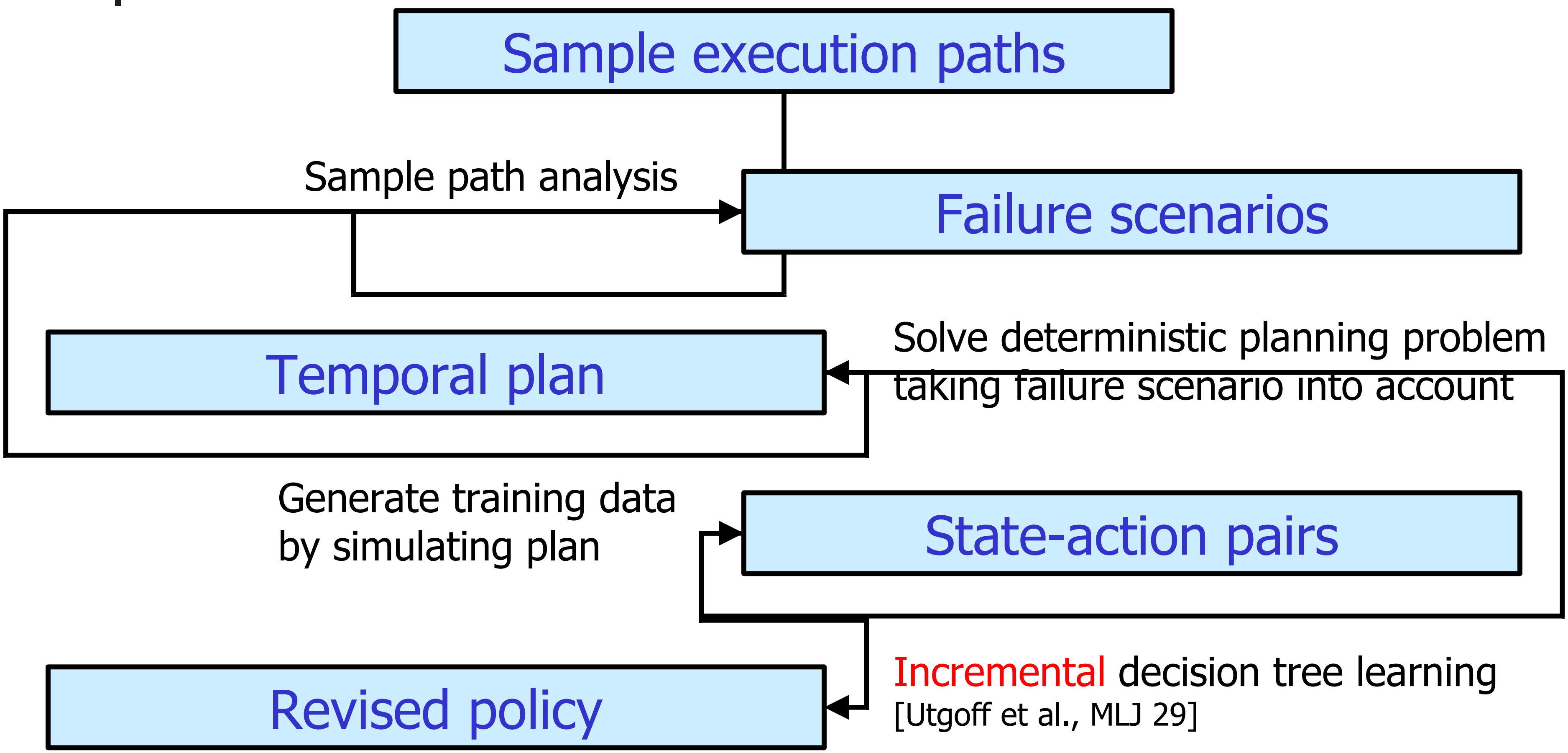


$at_{pgh\text{-}taxi,cmu}$
$at_{me,cmu}$
$at_{plane,mpls\text{-}airport}$
enter-taxi
depart-taxi
$at_{mpls\text{-}taxi,mpls\text{-}airport}$
$at_{me,pgh\text{-}airport}$
$at_{me,mpls\text{-}airport}$
$moving_{mpls\text{-}taxi,mpls\text{-}airport,honeywell}$
check-in
$in_{me,plane}$
enter-taxi
depart-taxi
idle
leave-taxi
idle
$moving_{pgh\text{-}taxi,cmu,pgh\text{-}airport}$
idle
leave-taxi

Closer Look at Debug Step



Policy Debugging

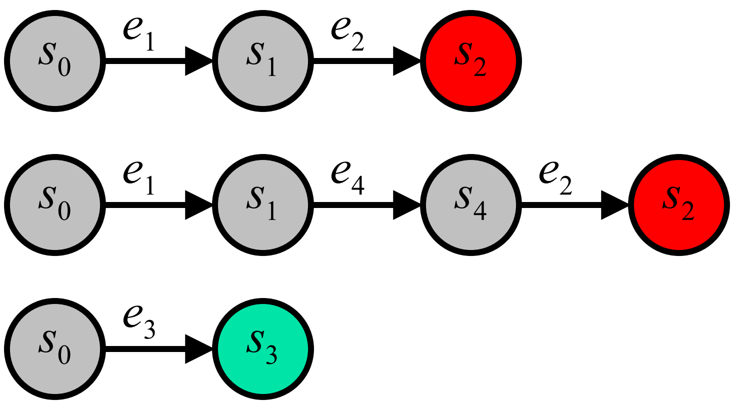


Sample Path Analysis

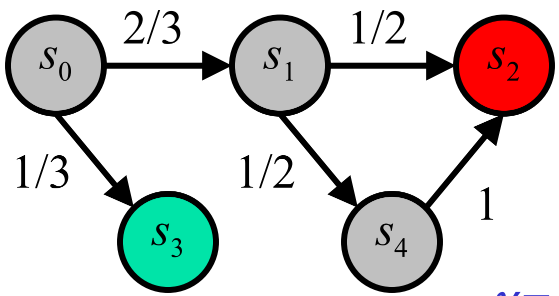
1. Construct Markov chain from paths
2. Assign values to states
   - Failure: -1; Success: $+1$
   - All other: $V(s) = \gamma \sum_{s' \in S} p(s'; s) V(s')$
3. Assign values to events
   - $V(s') - V(s)$ for transition $s \rightarrow s'$ caused by e
4. Generate failure scenarios

Sample Path Analysis: Example

Sample paths:



Markov chain:



$\gamma = 0.9$

Event values:

$V(e_1) = 2\cdot(V(s_1) - V(s_0)) = -1.284$

$V(e_2) = (V(s_2) - V(s_1)) + (V(s_2) - V(s_4)) = -0.245$

$V(e_3) = V(s_3) - V(s_0) = +1.213$

$V(e_4) = V(s_4) - V(s_1) = -0.045$

State values:

$V(s_0) = -0.213$

$V(s_1) = -0.855$

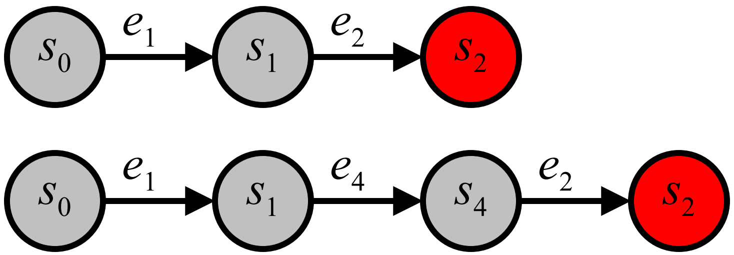
$V(s_2) = -1$

$V(s_3) = +1$

$V(s_4) = -0.9$

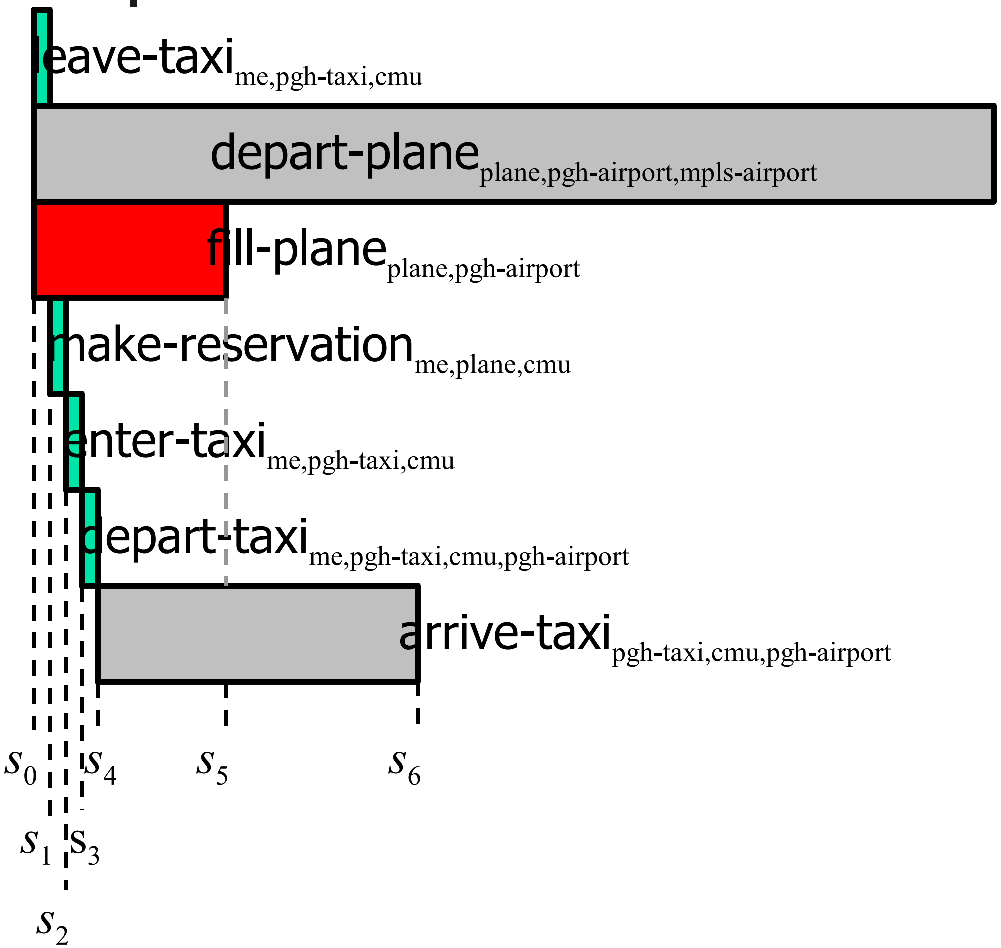
Failure Scenarios

Failure paths:



Failure path 1	Failure path 2	Failure scenario
e_1 @ 1.2	e_1 @ 1.6	e_1 @ 1.4
e_2 @ 4.4	e_4 @ 4.5	e_2 @ 4.6
-	e_2 @ 4.8	-

Additional Training Data



s_0: leave-taxi$_{me,pgh-taxi,cmu}$

s_1: make-reservation$_{me,plane,cmu}$

s_2: enter-taxi$_{me,pgh-taxi,cmu}$

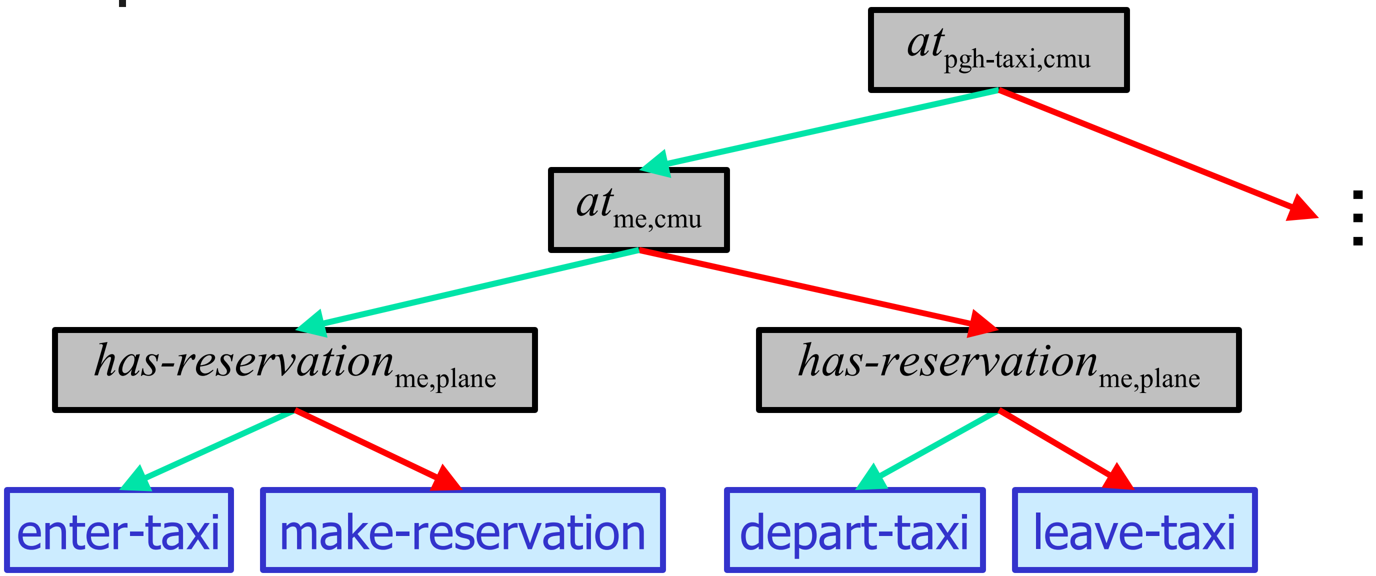
s_3: depart-taxi$_{me,pgh-taxi,cmu,pgh-airport}$

s_4: idle

s_5: idle

…

Revised Policy Tree



Summary

- Planning with stochastic asynchronous events using a deterministic planner
- Decision tree learning to generalize deterministic plan
- Sample path analysis for generating failure scenarios to guide plan repair

Coming Attractions

- Decision theoretic planning with asynchronous events:
  - “A formalism for stochastic decision processes with asynchronous events”, MDP Workshop at AAAI-04
  - “Solving GSMDPs using continuous phase-type distributions”, AAAI-04