A Framework for Simulation-Based Online Planning

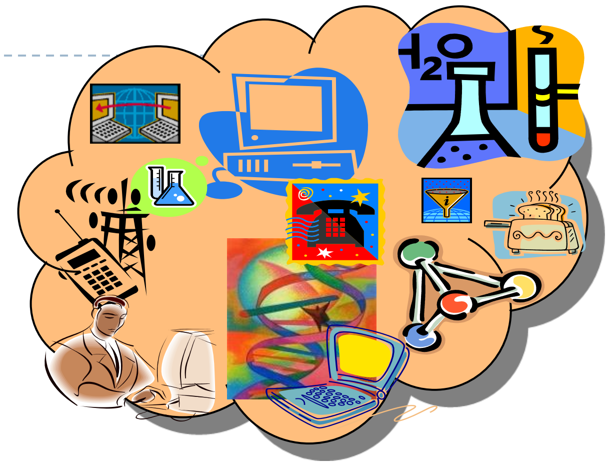
Martin Wirsing
In Kooperation mit Lenz Belzner und Rolf Hennicker,
FACS 2015, LNCS 9539, 2015, 1-30

Modellierung Dynamischer und Adaptiver Systeme
WS 2017/18

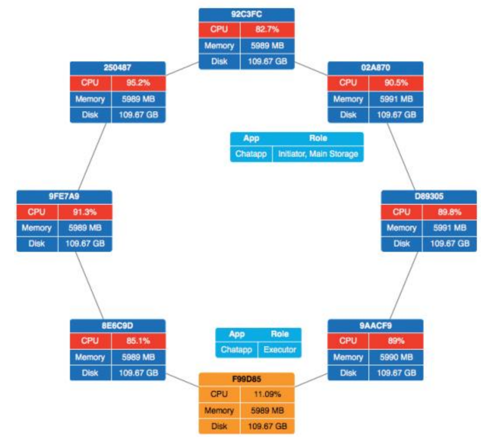
Autonomous Systems

- Autonomous systems
  have to adapt to
  - environmental conditions and
  - new requirements

  at runtime even if they are defined at design time

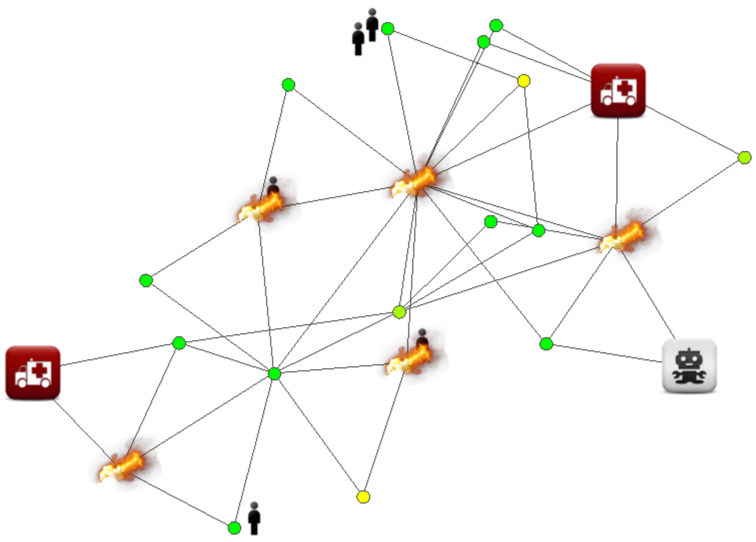


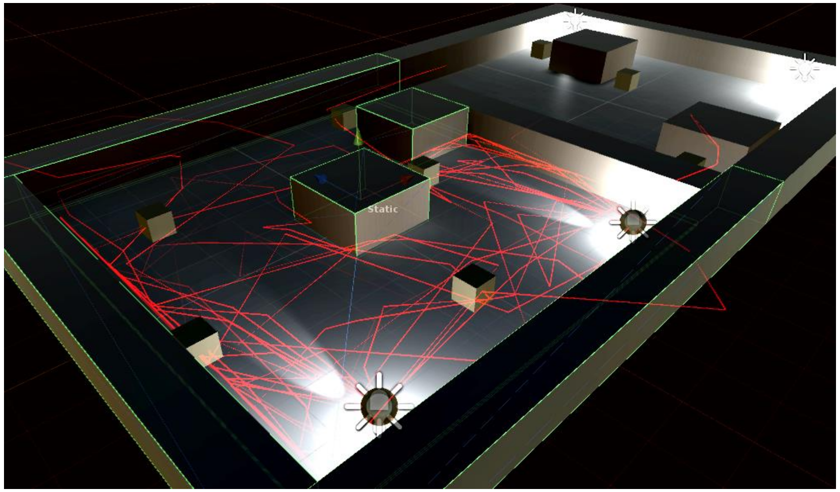
- ASCENS project
  - 2010-2015, EU-funded Integrated Project
  - 15 partners from 7 countries
  - Developed systematic approach for
    engineering autonomous ensembles including
    - SW process, formal modeling, verification,
    - monitoring, adaptation, awareness
  - Case studies on
    robotics, cloud computing, e-mobility



Decision Making under Uncertainty

- Very large state spaces ($|S| > 10^{10}$)
- Probabilistic effects
- Partially uncontrolled environment
- Incomplete design time knowledge





Contents

1. Online Planning

Online Planning

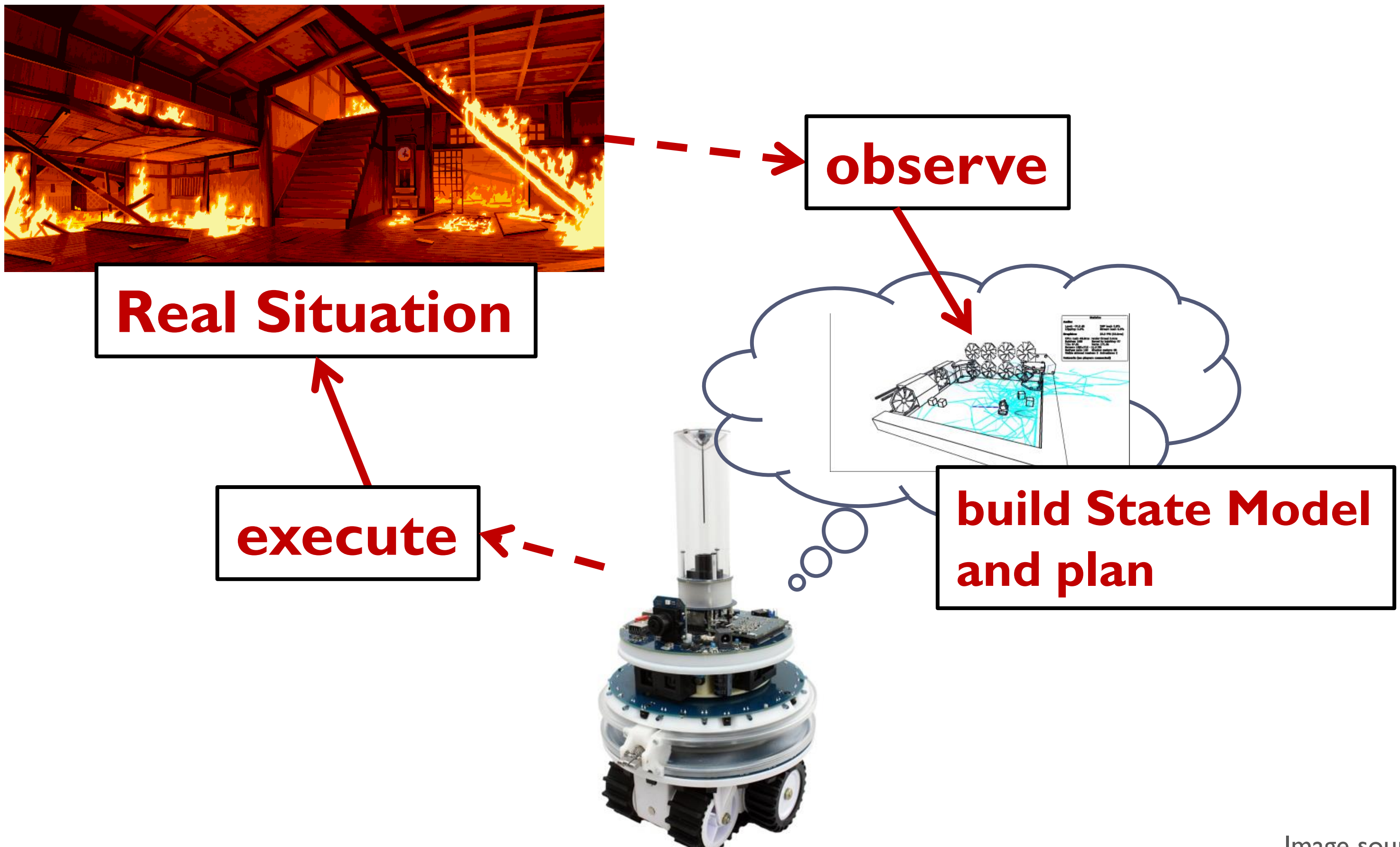


Image sources:
thegrid.soup.io/post/312159914
mobots.epfl.ch/marxbot.html

Online Planning

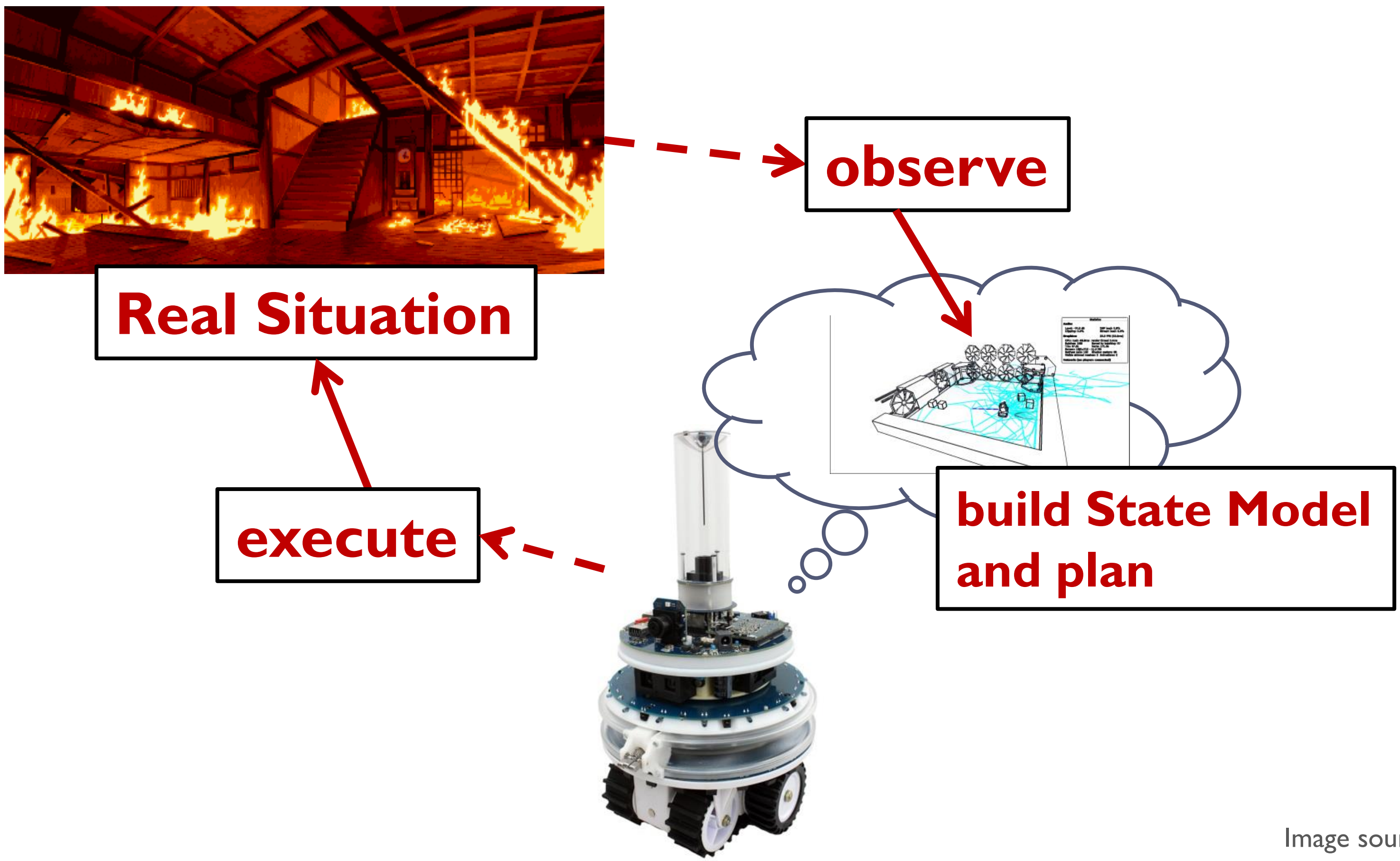


Image sources:
thegrid.soup.io/post/312159914
mobots.epfl.ch/marxbot.html

Online Planning (Informally, Sequential)

```
while true do
  observe state
  plan
  execute action w.r.t. plan
end while
```

Online Planning (Informally, Concurrent)

```
while true do
  observe state
  execute || plan
end while
```

Online Planning: Parameters

- State space S
- Action space A
  - Attribute $actionRequired : Agent \rightarrow Bool$
  - Operation $observe : Agent \rightarrow S$
- Operation *execute* : *RealAction* → *()*

- Planning
  - Reward function $R : S \rightarrow \mathbb{R}$ => *getReward*
  - Strategy $P_{Action}(A \mid S)$ => *sampleAction*
  - Planning refines initial strategy according to R
- Online planning
  - Iterated execution and planning

Online Planning (Refined)

Agent || Planner where

Agent

```
while true do
  state ← observe()
  planner.state ← state
  when actionRequired do
    actionRequired ← false
    action ← planner.strategy.sampleAction(state)
  end when
  action.real.execute()
end while
```

Planner

```
while true do
  plan()
end while
```

Plug Points

Agent || Planner where

Agent

```
while true do
  state ← observe()
  planner.state ← state
  when actionRequired do
    actionRequired ← false
    action ← planner.strategy.sampleAction(state)
  end when
  action.real.execute()
end while
```

Planner

```
while true do
  plan()
end while
```

domain specific

Plug Points

Agent || Planner where

Agent

```
while true do
  state ← observe()
  planner.state ← state
  when actionRequired do
    actionRequired ← false
    action ← planner.strategy.sampleAction(state)
  end when
  action.real.execute()
end while
```

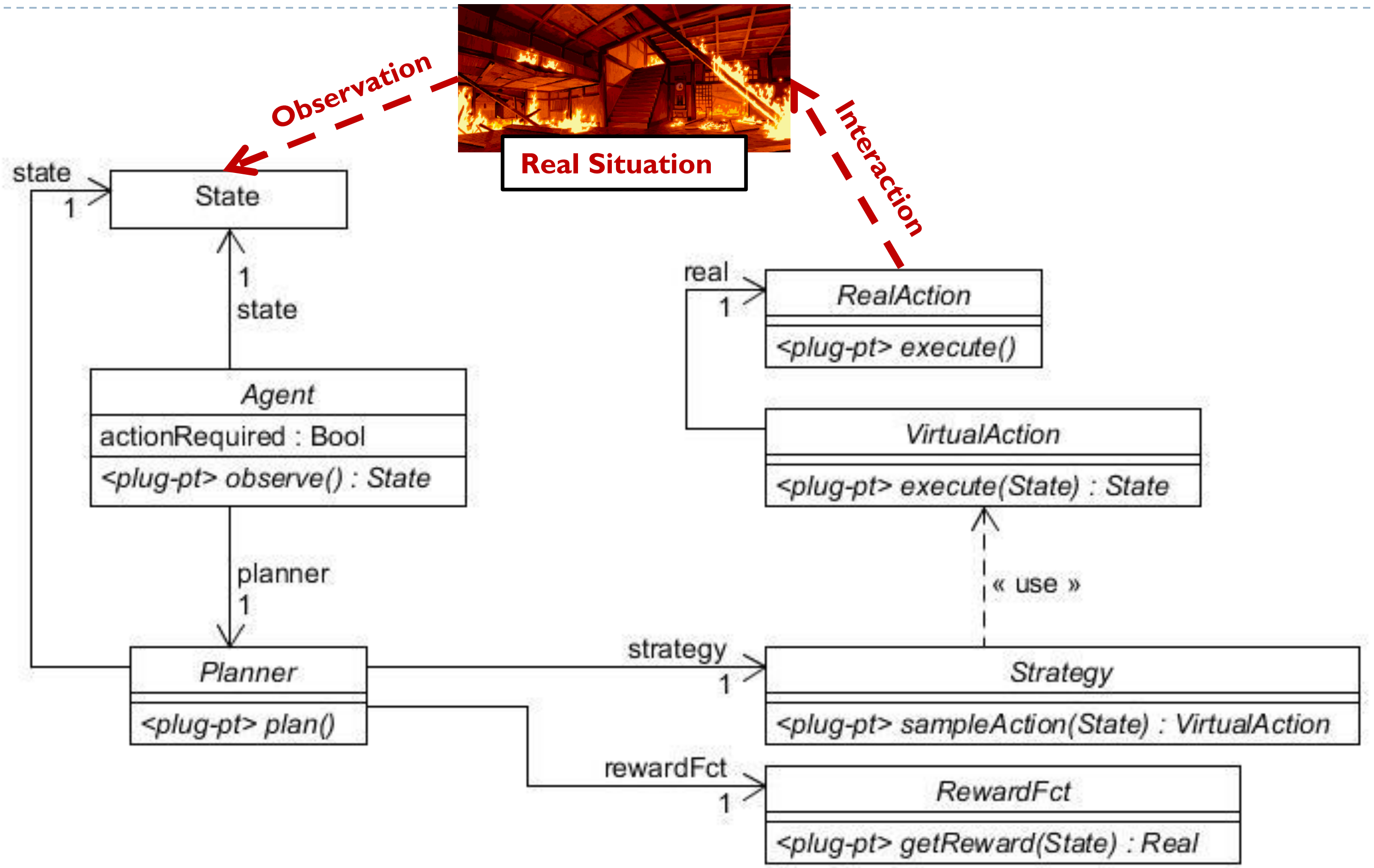
Planner

```
while true do
  plan()
end while
```

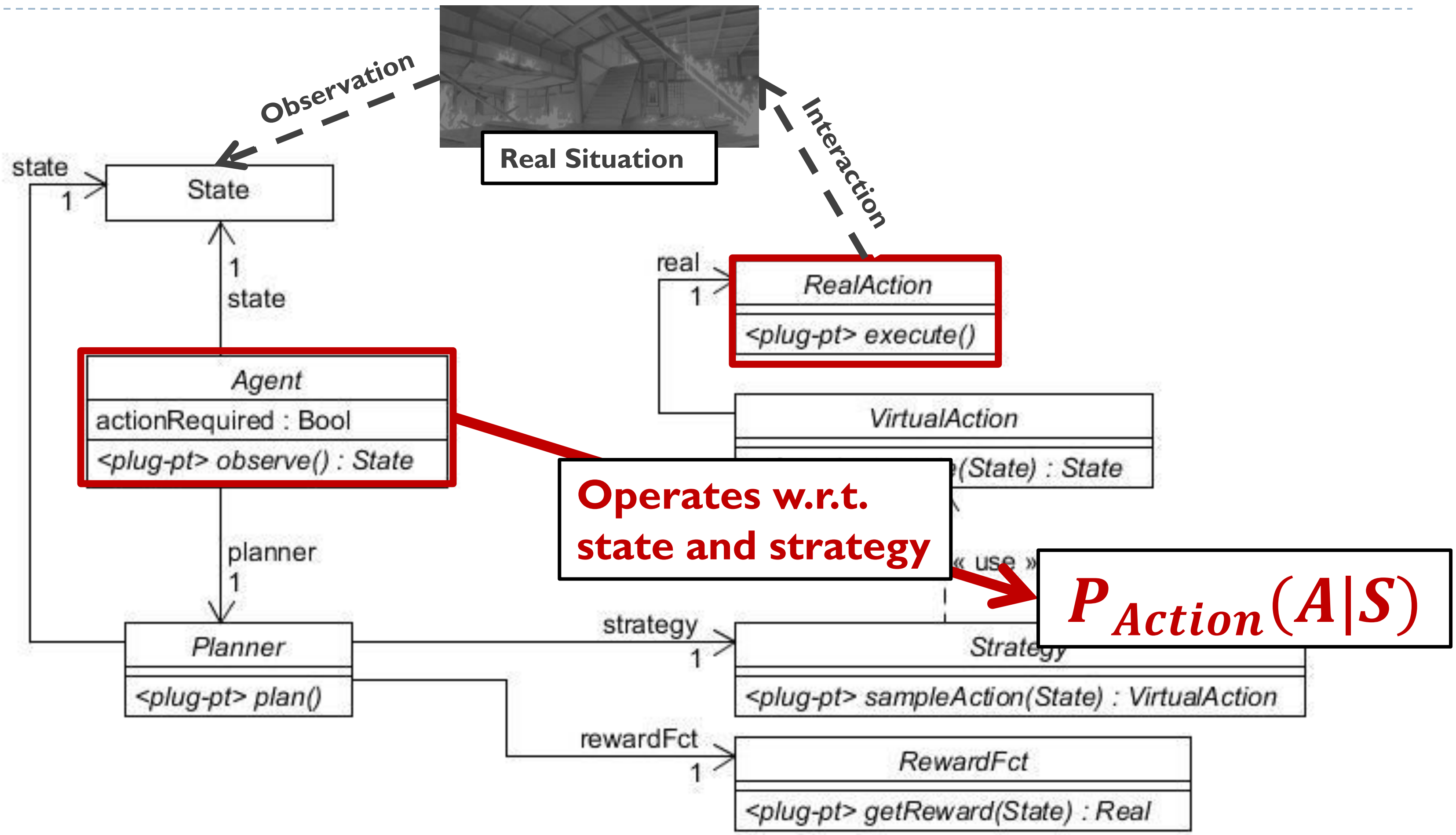
domain specific

design choice

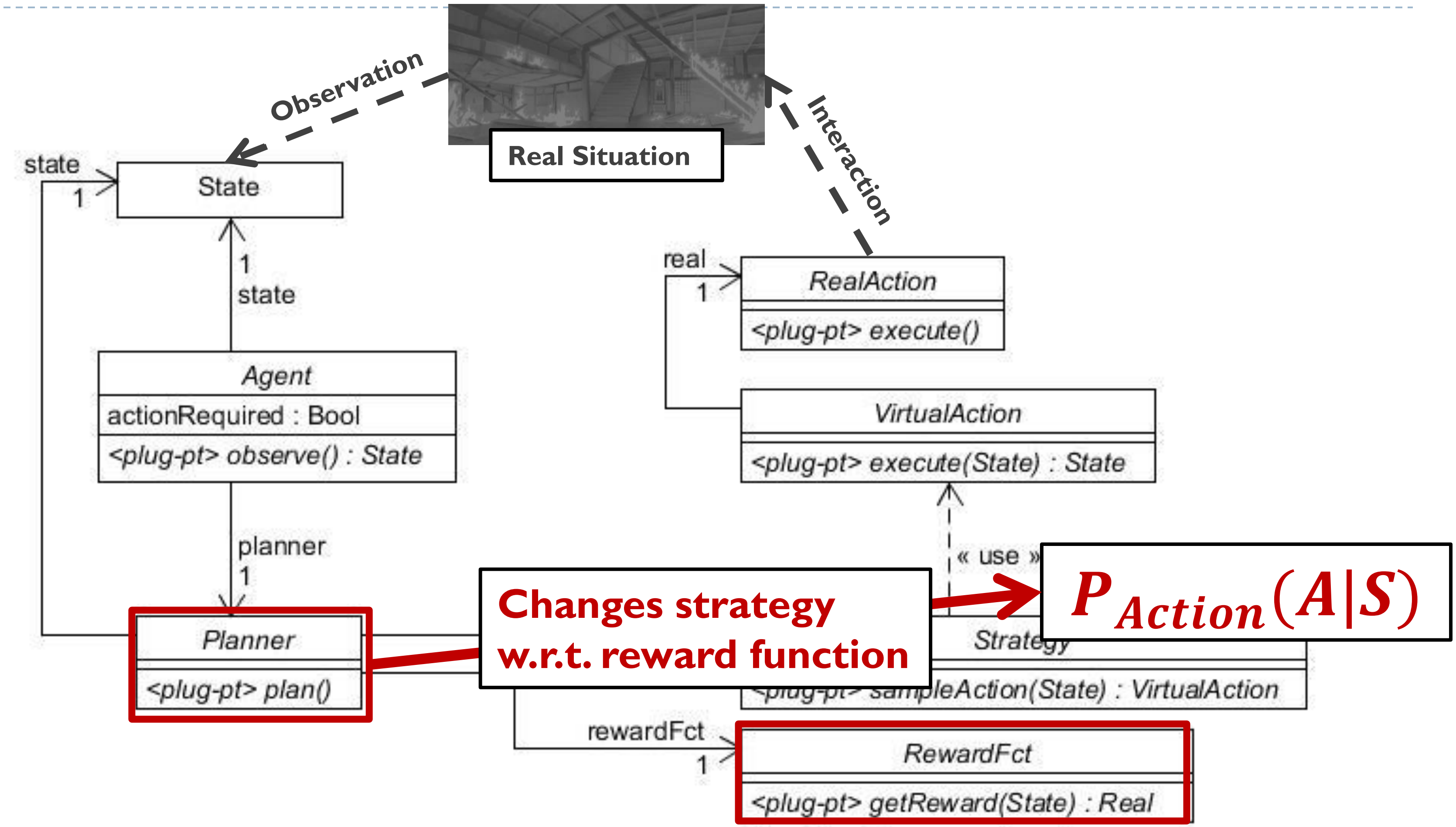
A Framework for Online Planning



A Framework for Online Planning



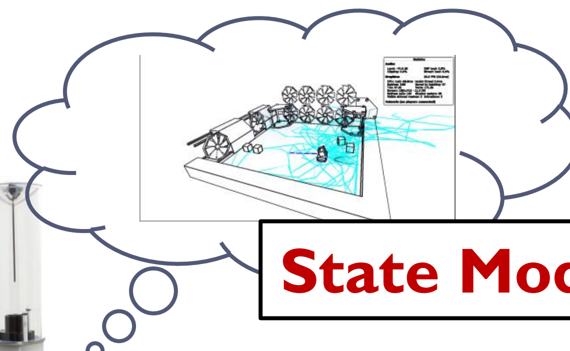
A Framework for Online Planning



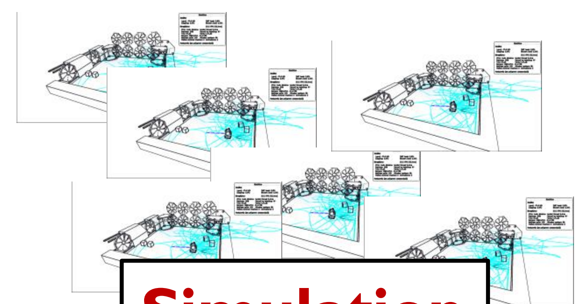
3. Simulation-Based Online Planning

Three Types of State

Real Situation



State Model

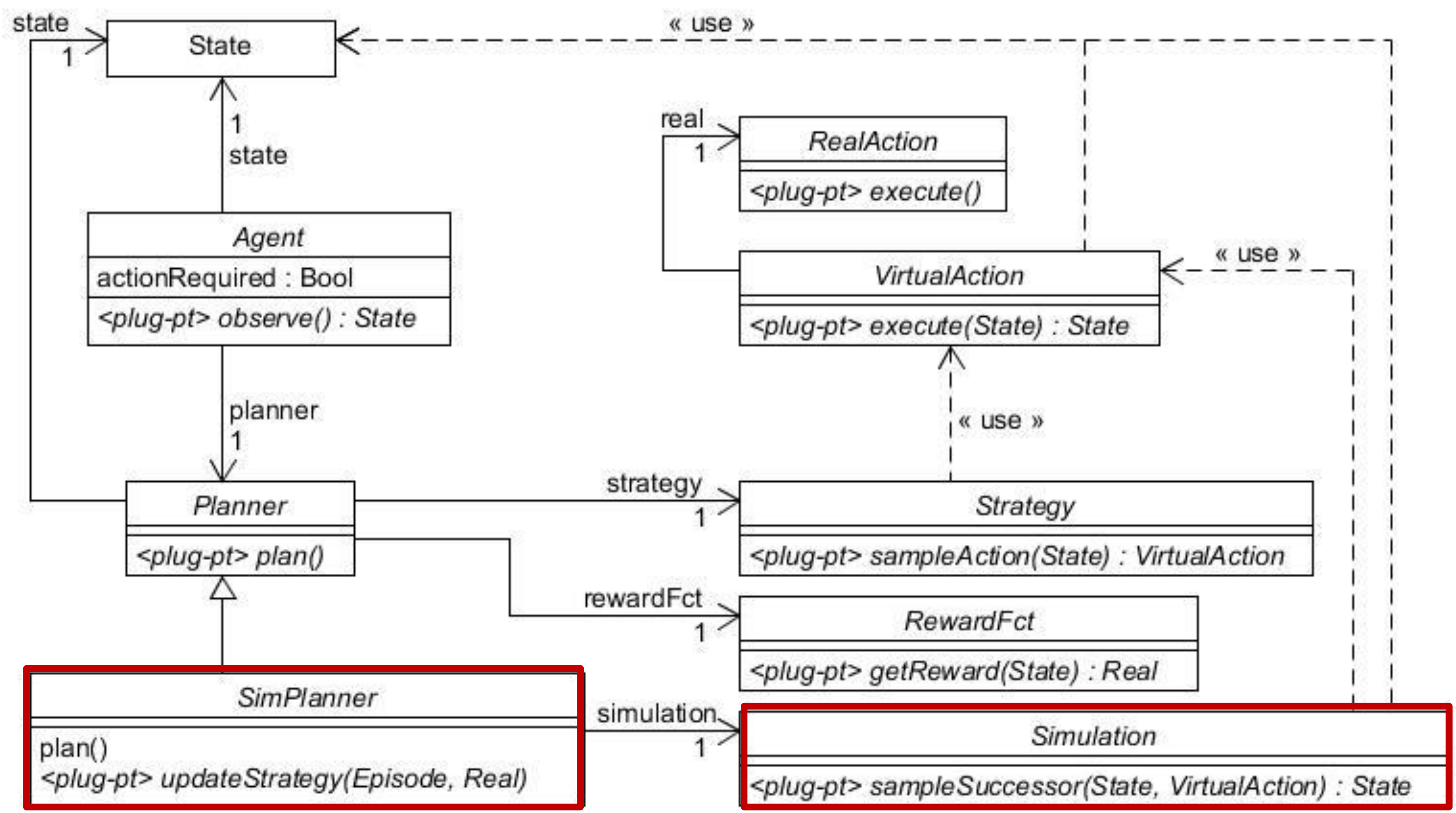


Simulation

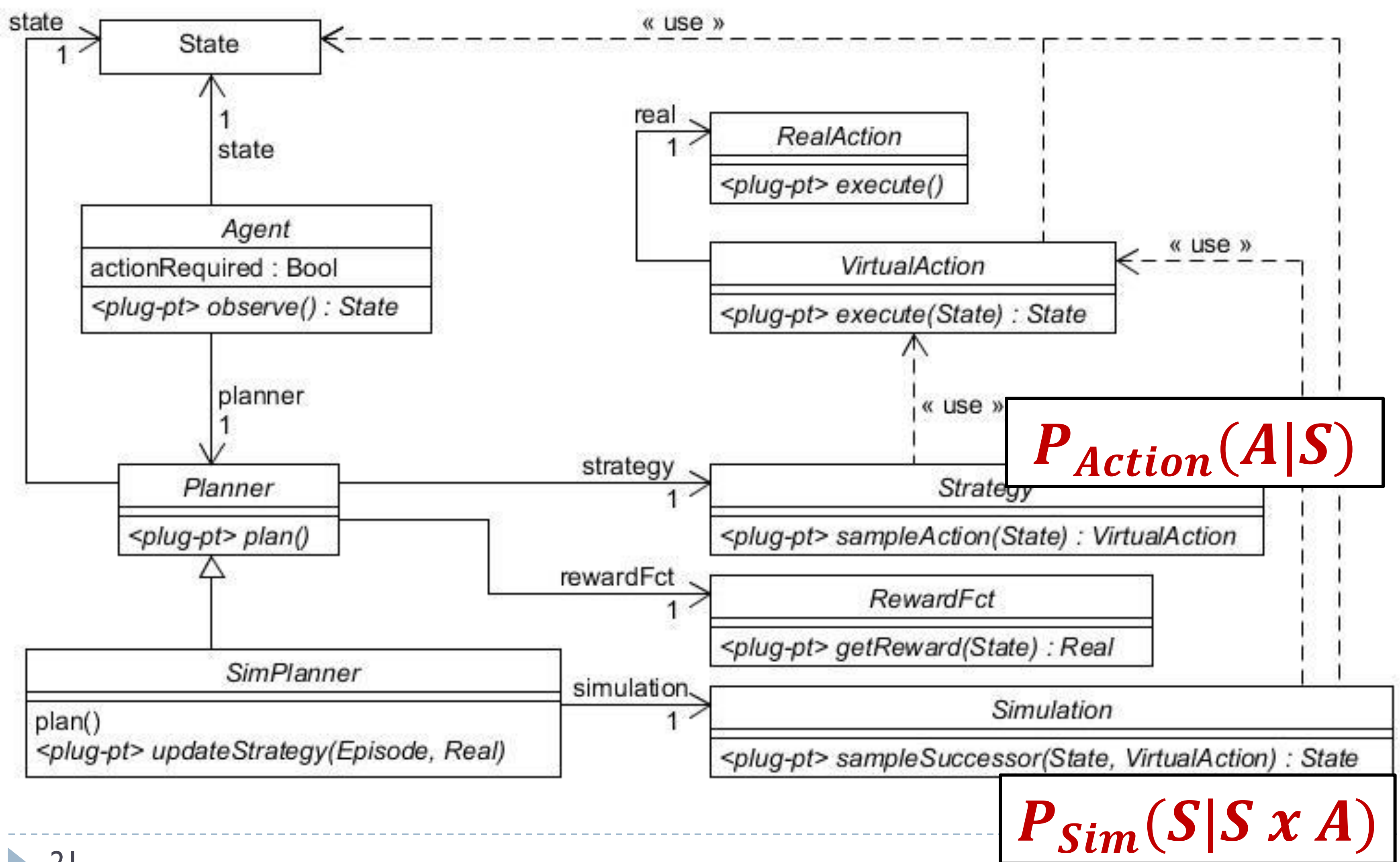
Approach

- Refine strategy $P_{Action}(A|S)$ by Simulation-Based Planning
  - Provide agent with simulation of itself and domain
  - Generate simulations of future episodes
  - Evaluate simulation episodes wrt. reward function
  - Use estimates to refine simulations
  - Finally: Execute a real action that performed well in simulation
  - Repeat

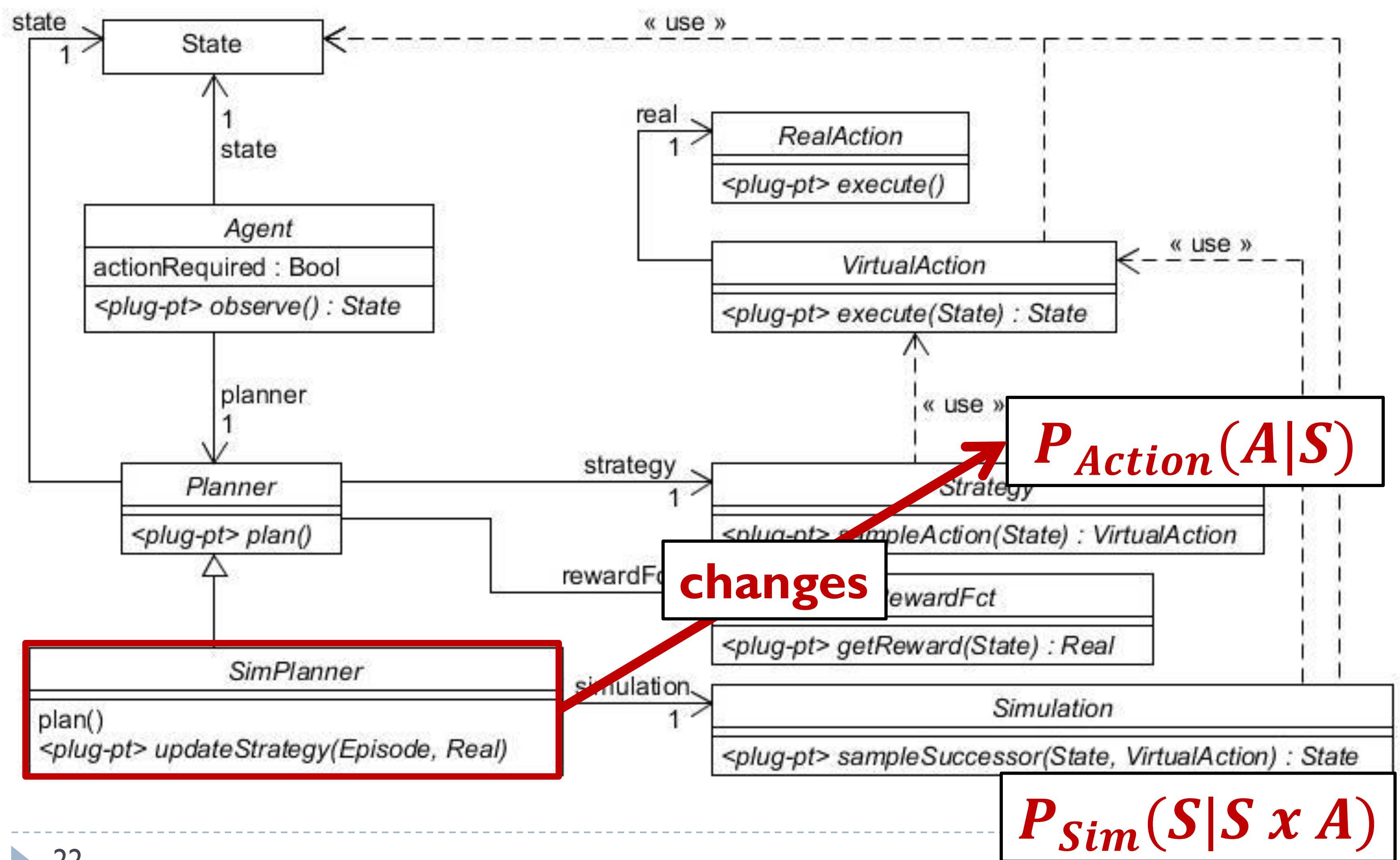
3.1 The Framework for Simulation-Based Planning



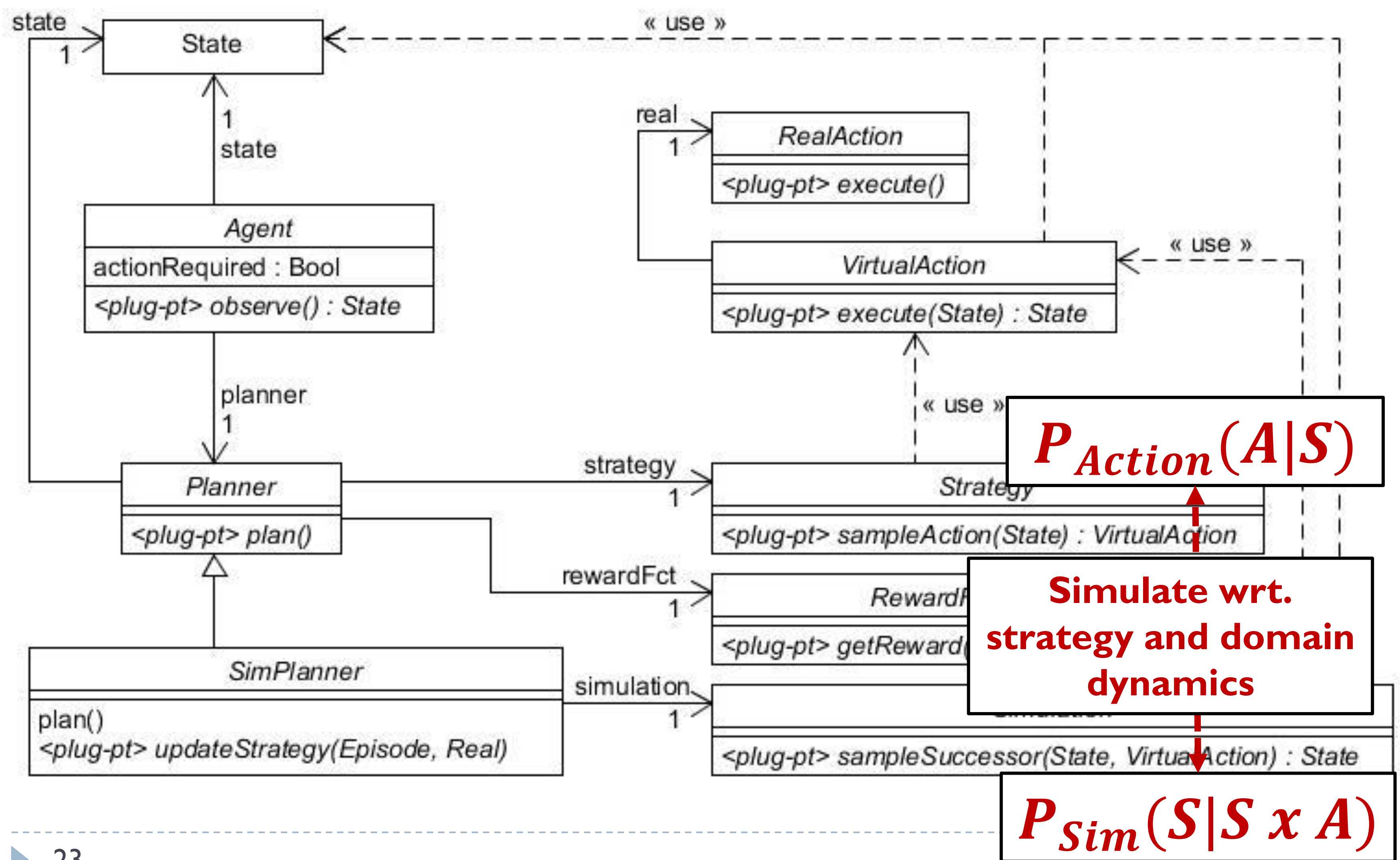
The Framework for Simulation-Based Planning



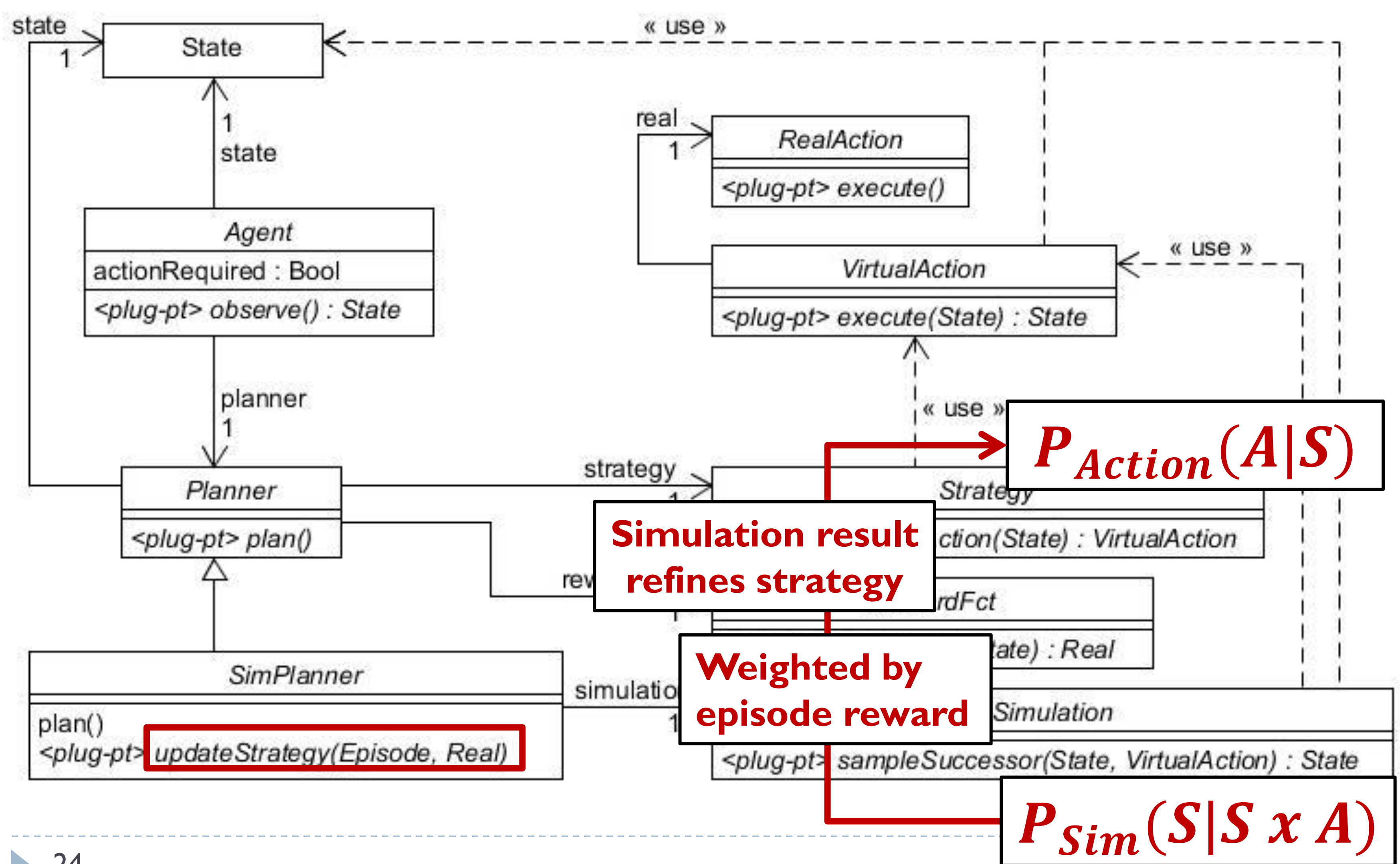
The Framework for Simulation-Based Planning



The Framework for Simulation-Based Planning



The Framework for Simulation-Based Planning



SBP Parameters

- Simulation $P_{Sim}(S \mid S \; x \; A)$
  - Agent‘s model/knowledge of domain dynamics
  - Can be changed at runtime
  - May differ from real domain dynamics
  - Can be learned/refined from observations
- Maximum search depth h_{max}
  - Impacts simulation effort
  - Less simulation steps: Fast but shallow planning
  - Can be dynamically adapted

Simulation-Based Planning Algorithm

```
op plan()
  vars s, r, episode, a
  s ← state
  r ← rewardFct.getReward(s)
  episode ← nil
  for 0 .. h_max do
    a ← strategy.sampleAction(s)
    s ← simulation.sampleSuccessor(s, a)
    episode ← episode::(s, a)
    r ← r + rewardFct.getReward(s)
  end for
  strategy ← updateStrategy(episode, r)
end op
```

Simulation-Based Planning: Plug Points

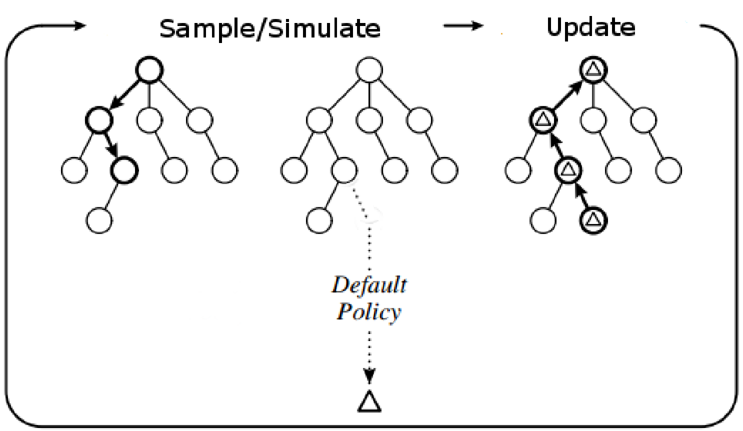
```
op plan()
  vars s, r, episode, a
  s ← state
  r ← rewardFct.getReward(s)
  episode ← nil
  for 0 .. h_max do
    a ← strategy.sampleAction(s)
    s ← simulation.sampleSuccessor(s, a)
    episode ← episode::(s, a)
    r ← r + rewardFct.getReward(s)
  end for
  strategy ← updateStrategy(episode, r)
end op
```

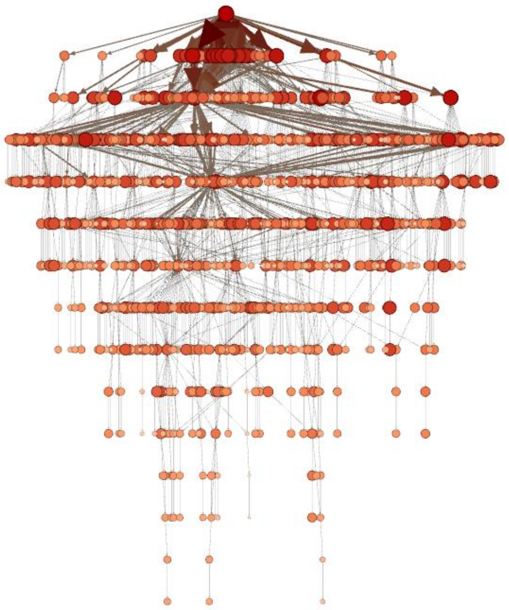
Simulation-Based Planning: Variants

- Variants define `updateStrategy(Episode, Real)`
  - Vanilla Monte Carlo
  - Genetic Algorithms
  - **Monte Carlo Tree Search**
    - **for discrete domains**
  - **Cross Entropy Planning**
    - **for continuous domains**

3.2 Monte Carlo Tree Search for Discrete Domains

- Strategy as tree
  - Nodes represent states and action choices
  - Add a node per simulation
  - Aggregate simulation data in nodes
    - Reward and frequency
  - Sample actions w.r.t. aggregated data



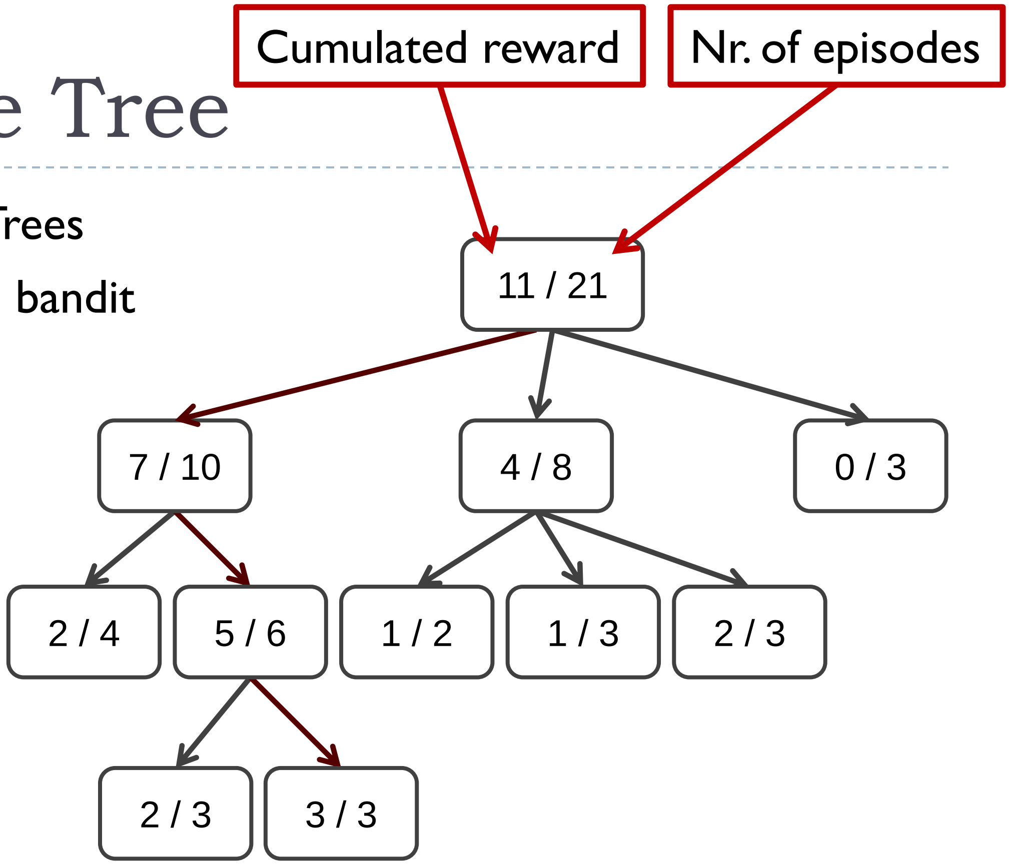


Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. *A survey of monte carlo tree search methods.* Computational Intelligence and AI in Games, IEEE Transactions on, 4(1):1 - 43, 2012.

Strategy Inside the Tree

- E.g. Upper Confidence Bounds for Trees
- Treat action selection as multiarmed bandit
- Select actions that maximize

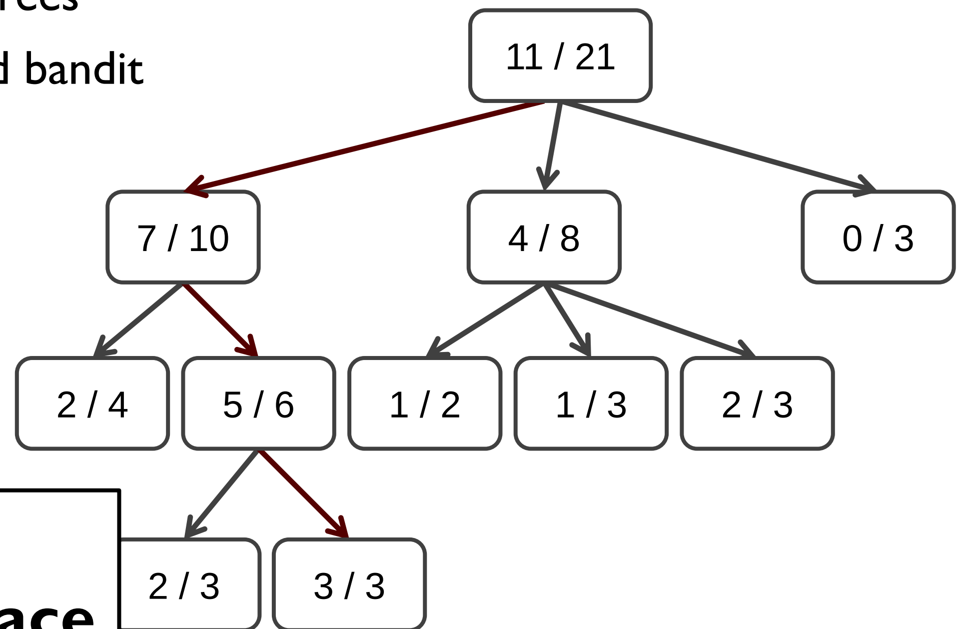
$$UCT_j = X_j + 2C\sqrt{\frac{2 \ln n}{n_j}}$$



Kocsis, Levente, and Csaba Szepesvári. *Bandit based monte-carlo planning*. Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293.

Strategy Inside the Tree

- E.g. Upper Confidence Bounds for Trees
- Treat action selection as multiarmed bandit
- Select actions that maximize



$$UCT_j = X_j + 2C\sqrt{\frac{2\ln n}{n_j}}$$

Exploit observations

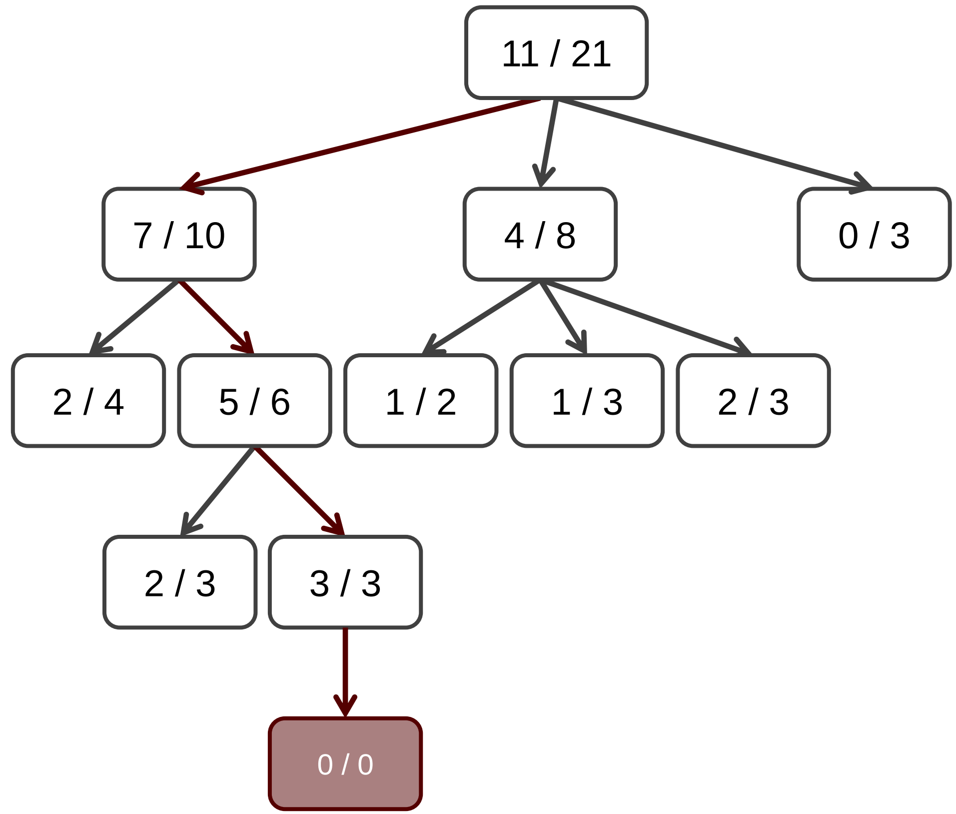
Explore solution space

- X_j: Average reward of child node j
- n: Nr. of episodes from current node
- n_j: Nr. of episodes from child node j
- C: UCT exploration constant

Kocsis, Levente, and Csaba Szepesvári. *Bandit based monte-carlo planning*. Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293.

Expand the Tree

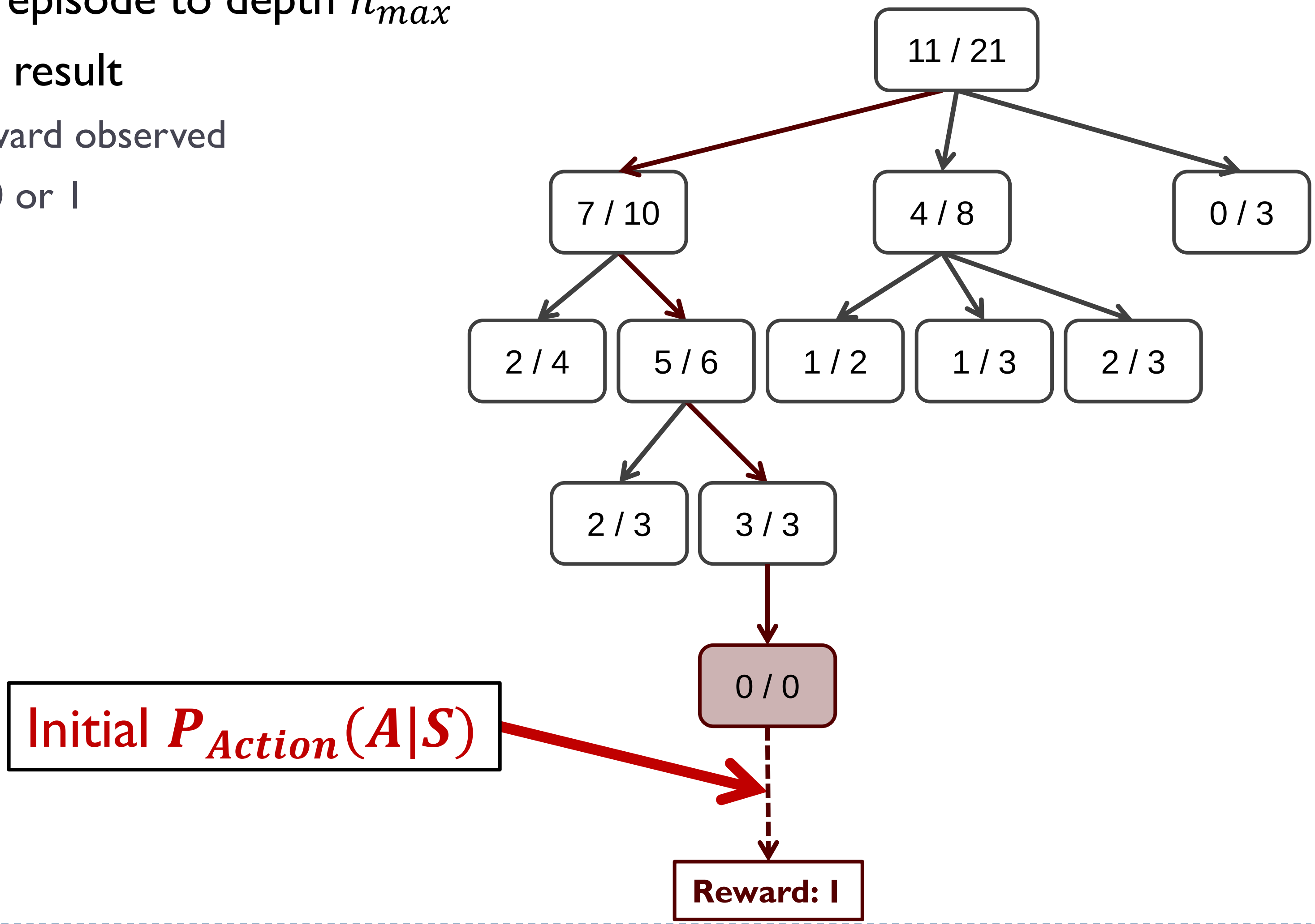
- Add a new node
  - When an episode leaves the tree



Kocsis, Levente, and Csaba Szepesvári. *Bandit based monte-carlo planning*. Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293.

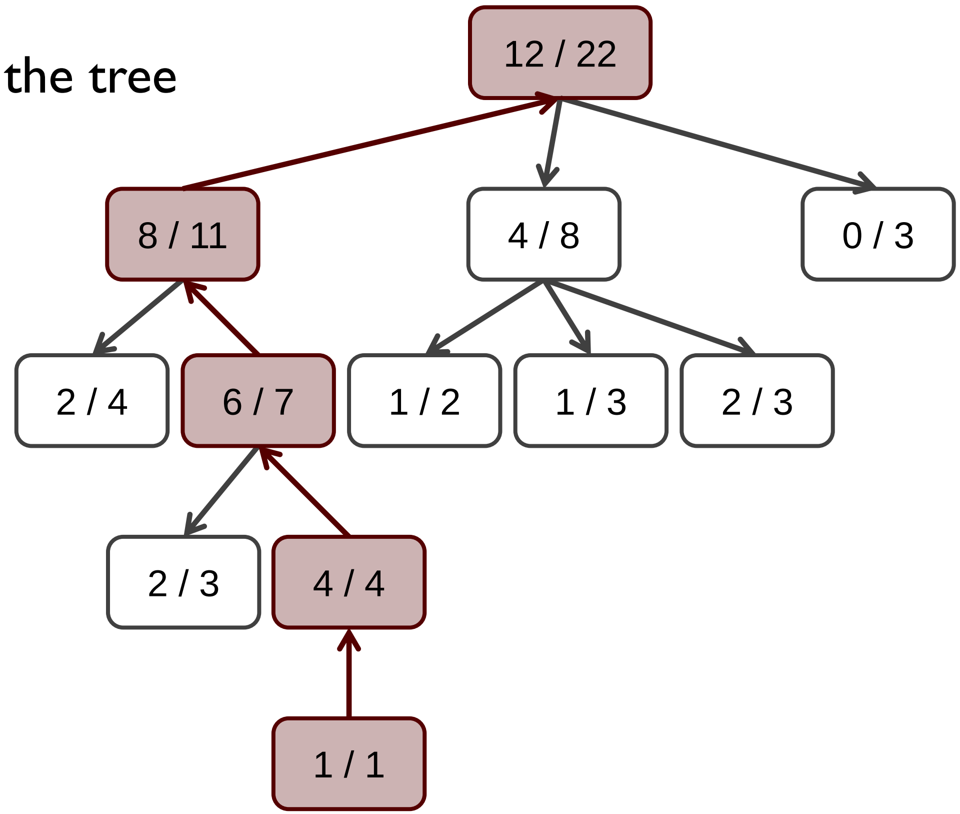
Strategy Outside the Tree

- Simulate episode to depth h_{max}
- Observe result
  - E.g. reward observed
  - Here: 0 or 1

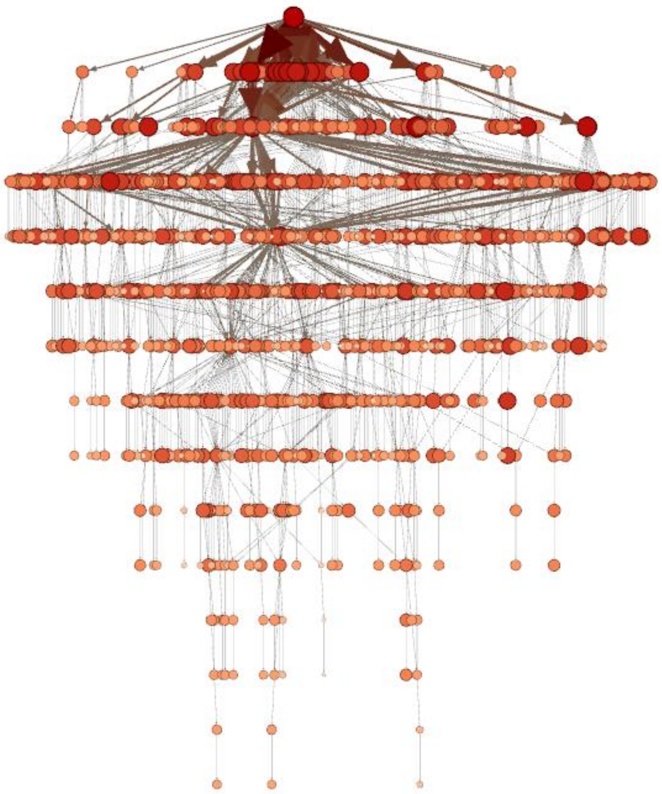


Update Strategy

- Update the statistics
- This changes the strategy inside the tree



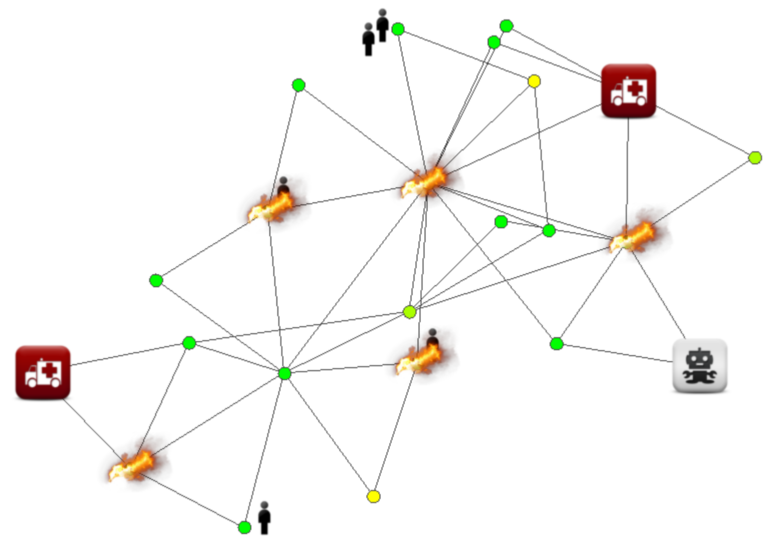
Trees Represent Strategies

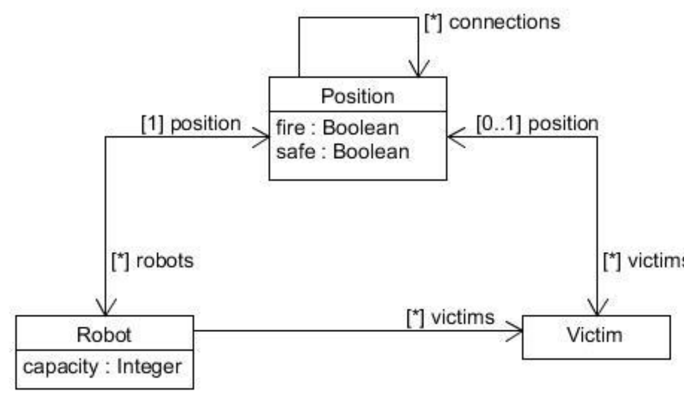


- MCTS builds a skewed tree
- Tree can be interpreted as $P_{Action}(A|S)$
- Promising parts of the strategy space are prefered

Example Domain

- Search and Rescue
  - Victims, fires and ambulances
  - Unknown topology
  - Unknown initial situation
- Agent actions
  - Noop, Move
  - Load or drop a victim
  - Extinguish fire if adjacent
- Noise
  - Actions may fail
  - Fires ignite and cease
- Experiment
  - Monte Carlo Tree Search
  - Large state space ($> 10^{12}$)
  - Large branching factor (2^{18})
  - 0.2 seconds/decision
  - $P_{Sim}(S \mid S\ x\ A)$ models domain perfectly

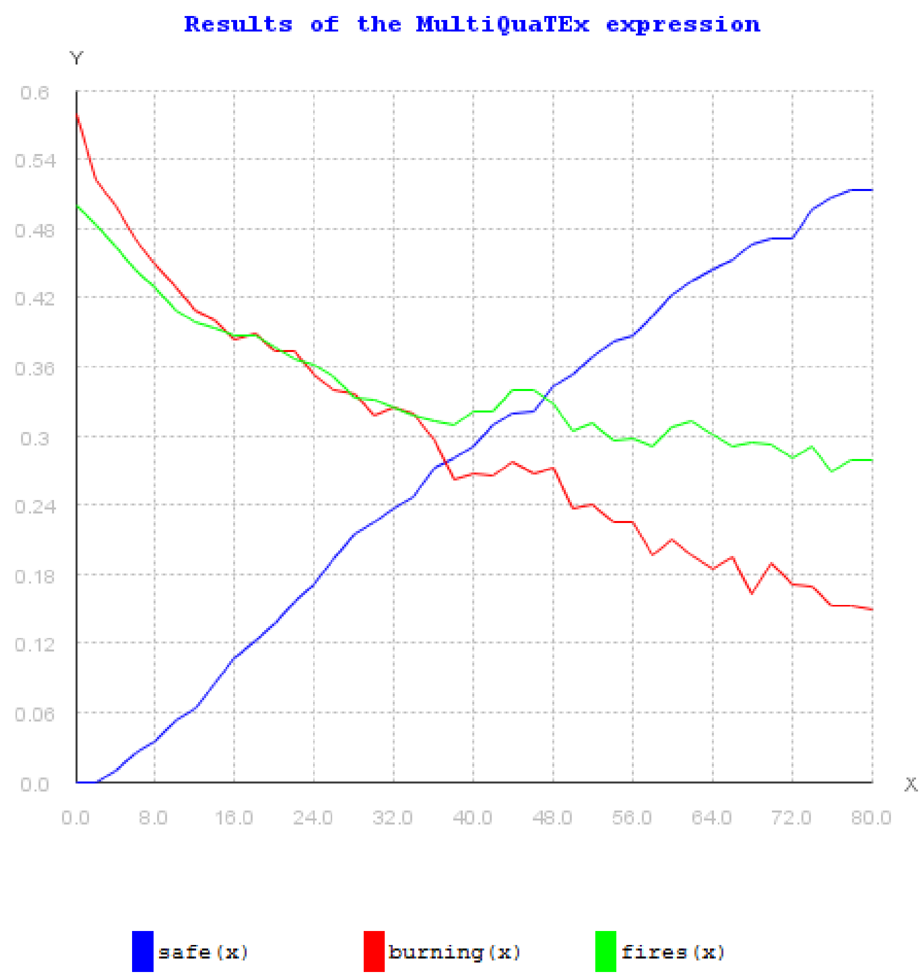




Experimental Results (I)

- Measured (in %)
  - Victims at ambulance (blue)
  - Victims in a fire (red)
  - Positions on fire (green)
- Provided reward
  - Victim at ambulance: +100
- System synthesized sensible behavior
- Results in 0.95 confidence interval
  - Checked with MultiVeStA

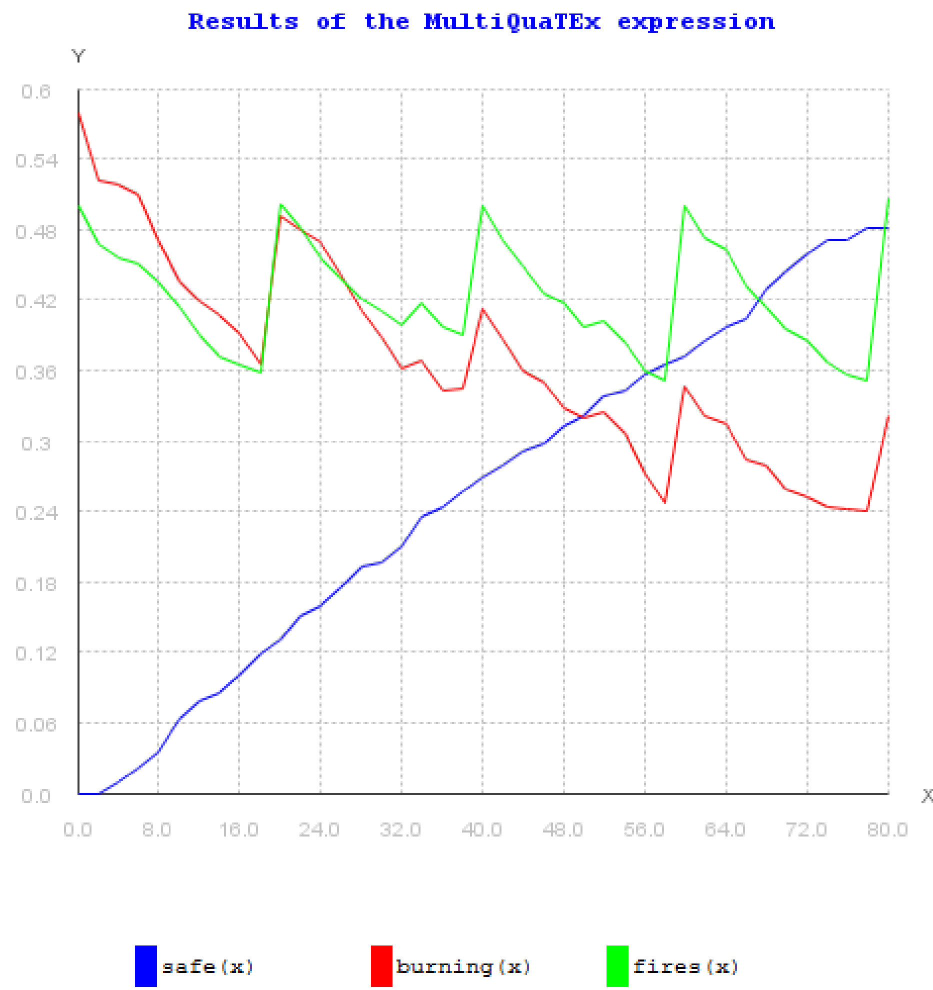
Stefano Sebastio and Andrea Vandin. *MultiVeStA: statistical model checking for discrete event simulators.* ValueTools '13. 2013. 310-315.



Autonomy

Experimental Results (II)

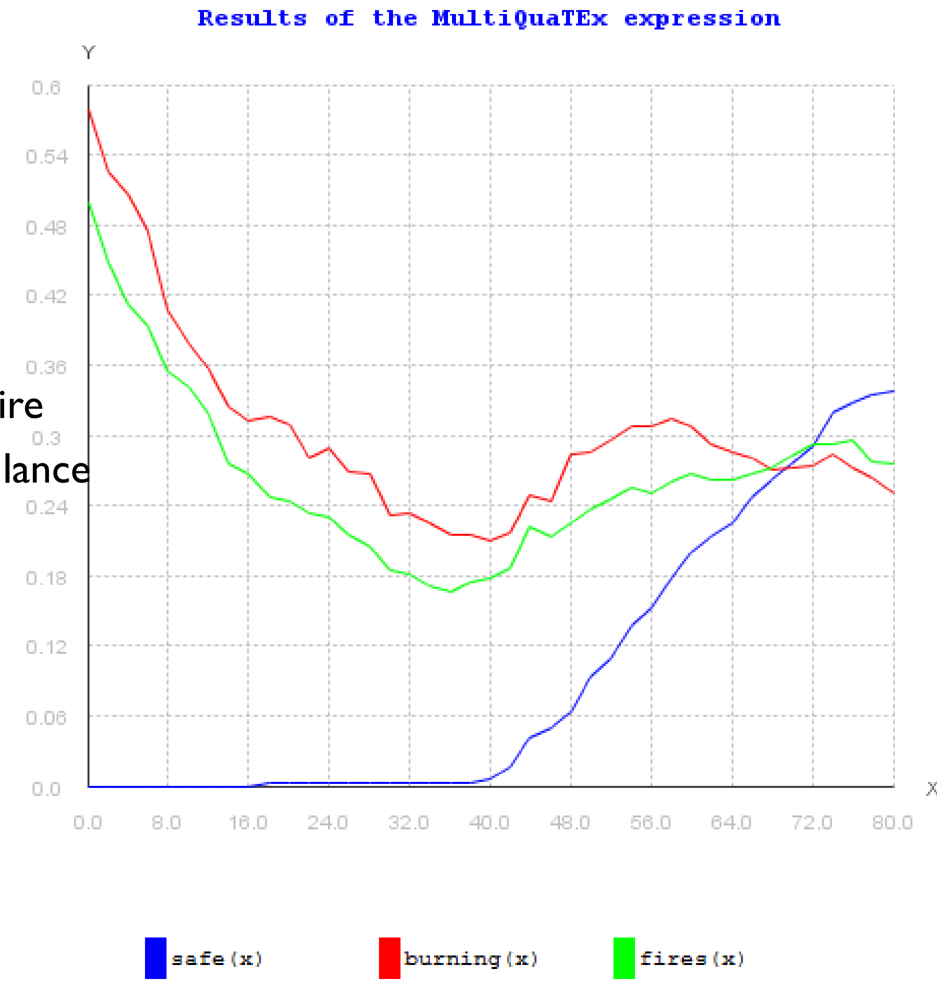
- Measured (in %)
  - Victims at ambulance (blue)
  - Victims in a fire (red)
  - Positions on fire (green)
- Expose system to **unexpected events**
  - At steps 20, 40, 60, 80
  - All carried victims are dropped
  - New fires break out
  - Events NOT simulated by planner
  - New situation incorporated by planner
- System showed sensible reactions
- Results in 0.95 confidence interval



Robustness

Experimental Results (III)

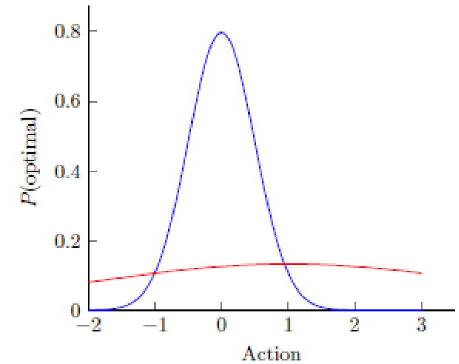
- Measured (in %)
  - Victims at ambulance (blue)
  - Victims in a fire (red)
  - Positions on fire (green)
- **Change system goals** while operating
  - Change of reward function
    - Steps 0-40: Reward for victims not in a fire
    - Steps 40-80: Reward for victims at ambulance
  - Change NOT simulated by planner
  - But planner incorporates new situation
- System adapted behavior wrt. goals
- Results in 0.95 confidence interval

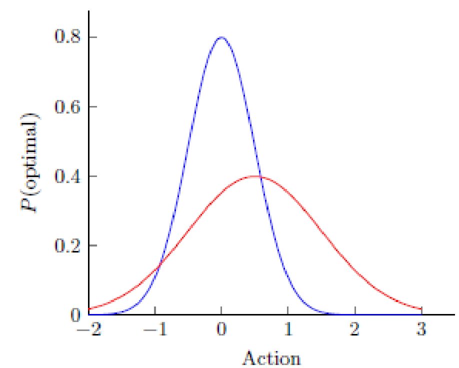


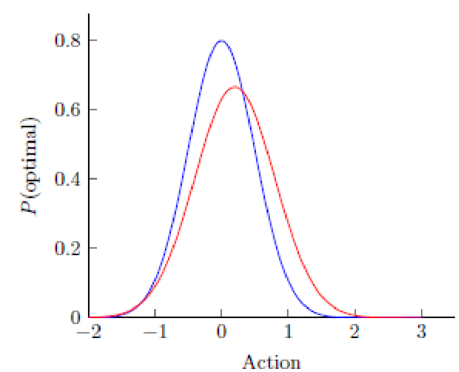
Flexibility

From Discrete to Continuous Domains

- Actions
  - State and action space = $\mathbb{R}^n$
  - E.g. (speed, rotation, duration) for actions
- Cross Entropy Planning
  - Approximate (unknown) target distribution
    - Multivariate Gaussian distribution
    - Sample state space (locally) and choose „elite“ samples for updating the strategy (‚sharpen‘ the Gaussian)
  - Here: Gaussians over sequences of actions
    - Sequence length = planning depth



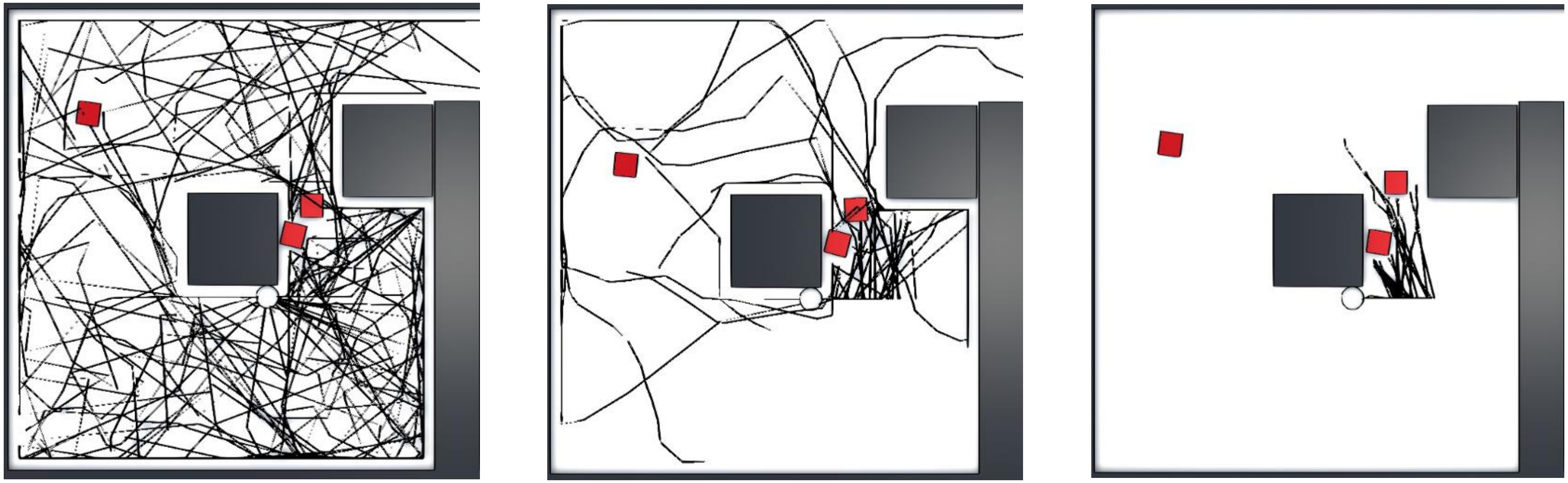




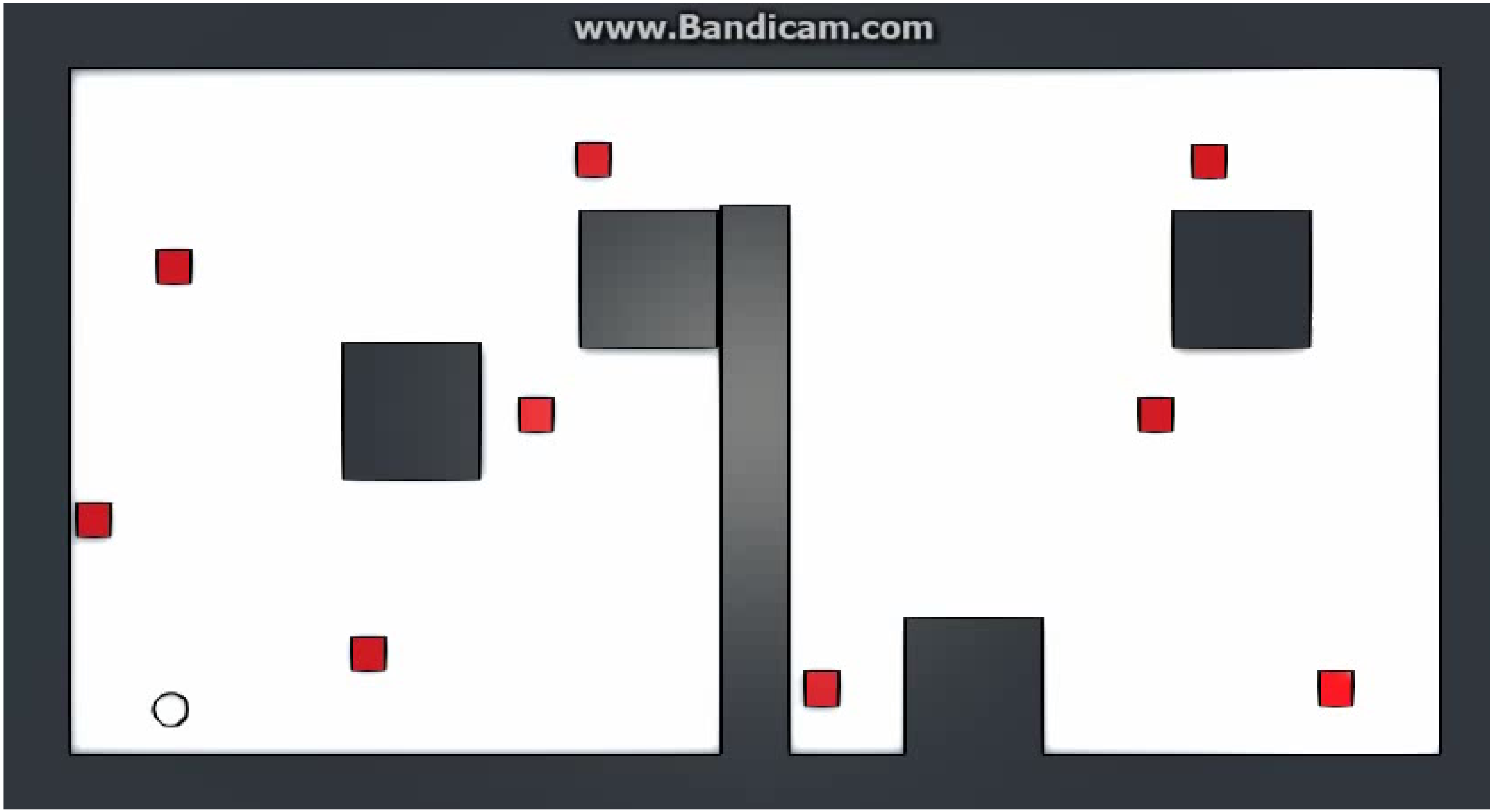
Ari Weinstein and Michael L. Littman. *Open-loop planning in large-scale stochastic domain*
Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.

Cross Entropy Planning

- White circle represents agent
- Red boxes represent moving victims
- Black lines are simulation episodes
- Action parameters are speed, rotation and duration
- Images show iterations 1, 5 and 10
  - Simulation depth is adaptive here (reduced simulation cost)
  - Note the iterative "shaping" of a promising strategy



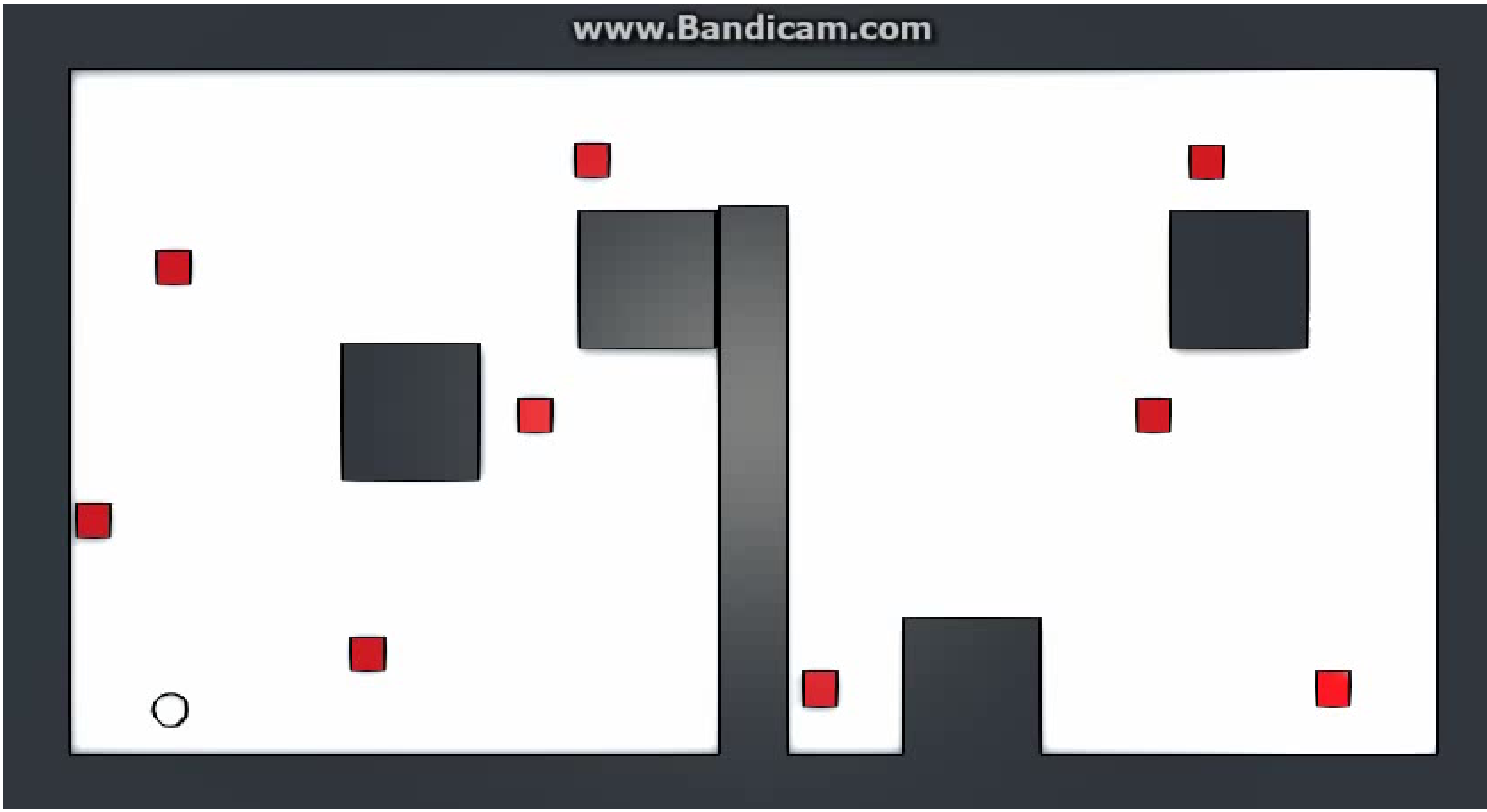
Video: Cross Entropy Planning



Video: Cross Entropy Planning

- The video showed interleaving planning and execution
- Illustrates iterative shaping of a probabilistic strategy
- When parallelizing planning and execution, this looks a little different…

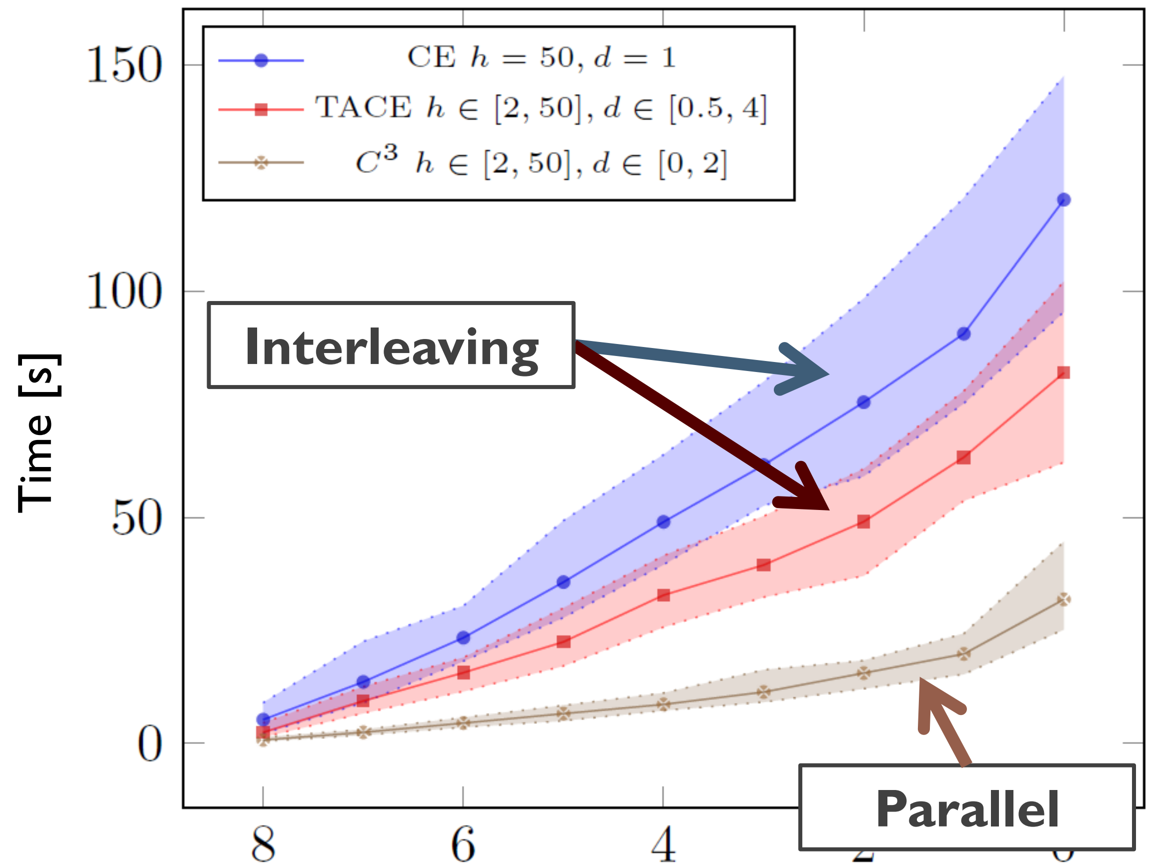
Video: Continuous CE Planning



Cross Entropy Planning Experiments

- CE: Cross Entropy Planning
- TACE: Time Adaptive CE
- C3: Continuous CE Control

- h: Planning depth
- d: Action duration



Concluding Remarks

- Motivation
  - Complex dynamic domains
  - High degrees of non-determinism
- Approach
  - Model a space of solutions, instead of a single one
  - Online planning: Refine the solution space at runtime wrt. observations and knowledge to determine a currently viable action
- This Talk
  - Component framework for Online Planning
    - Parallelization of execution and planning
  - Instantiation: Simulation Based Planning
    - Two examples: MCTS, Cross Entropy Planning
- Outlook
  - Model learning of domain dynamics
  - Soft temporal logic for formal (statistical) verification
  - Learning and planning for ensembles

References

1. Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. *A survey of Monte Carlo tree search methods*. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 2012, 1-43.
2. Kocsis, Levente, and Csaba Szepesvári. *Bandit based Monte-Carlo planning.* In *Machine Learning: ECML'06*. Lecture Notes in Computer Science 4212, 2006, 282-293.
3. Bubeck, Sébastien, and Rémi Munos. *Open Loop Optimistic Planning.* In: *23rd Conference on Learning Theory, COLT 2010*. Omnipress 2010, 477-489.
4. Ari Weinstein and Michael L. Littman. *Open-loop planning in large-scale stochastic domains*. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
5. Stefano Sebastio and Andrea Vandin. *MultiVeStA: statistical model checking for discrete event simulators*. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools* (ValueTools '13). 2013, 310-315.
6. Lenz Belzner, Rolf Hennicker, Martin Wirsing: *OnPlan: A Framework for Simulation-Based Online Planning*. In Christiano Braga, Peter Csaba Ölveczky (eds.): *Formal Aspects of Component Software - 12th International Conference, FACS 2015*, Revised Selected Papers. Lecture Notes in Computer Science 9539, 2016, 1-30.