**Service Composition in Stochastic Settings**

**Ronen Brafman**  
**Giuseppe De Giacomo & Massimo Mecella**  
**Sebastian Sardina**

---

### Service Composition

**Task:** Synthesize orchestrator that realizes virtual target service $T$ by coordinating available services $B_1 \ldots B_n$ in environment $E$.

- **Available services:** logic of machine, web service, etc.
- **Environment:** common ground for available services and target
- **Target:** desired virtual service.
- **Orchestrator:** delegates target actions to the available services

### Motivations

Classical service composition approaches:

- Operate on deterministic and nondeterministic (devilish) settings
- Deal only with *exact* solutions

### Objectives

Handle cases that do not admit exact solutions by:

1. Rewarding *actions in the target*
2. Allowing several sources of uncertainty:
   - **stochastic target**
   - **stochastic environment**
   - **stochastic available services**
3. Solving composition by MDP – basing optimality on target “expected realizability”

### Stochastic Composition

- Reward for each action request
- Stochastic model for target action requests
- Stochastic transition evolutions in avail. services & environment

---

### Orchestrator Evaluation

**Value of orchestrator:** Measures degree of target’s *expected realizability*

Rewards gained on:

- **successful action delegation:**  
  Probability of action request $\times$ reward for the action request
- **one target step:**  
  Reward $R_i$ gained for each delegable target action request
- **on infinite runs:**  
  $R_1 + \gamma R_2 + \gamma^2 R_2 + \ldots$, using discount factor $0 \leq \gamma < 1$

**Results**

If an exact orchestrator exists, then it is optimal

---

### Reduction to MDP

- **Encoded into MDP $M = (Q, A, p, r)$:**
  - $Q$ is the finite set of state encoding the states of the system, the state of the target, and the next requested action
  - $A = \{1, \ldots, n\}$ is the set of available service indexes
  - $p(q, i, q')$ is the stochastic transition relation encoding the possible next system state and action requested
  - $r(q, i)$ is the reward allocated on correct delegation (0 otherwise)

**Results**

- Optimal policy for $M \equiv$ optimal orchestrator
- Existence of exact orchestrator can be checked by calculating optimal policy of MDP

---

### Extensions

- **Handling exceptions:** undo actions!
- **Separate rewards specification:** composition is a notable example
- **Non-Markovian rewards** expressed in LTL$_F$ or LDL$_F$ or through programs, e.g., for:
  - extended constraints, e.g., action empty must be suitable only after action pluck has been executed
  - preferences, e.g., in certain conditions Plucker-bot uses less energy than the Multi-bot
- **Reinforcement learning:** learn probabilities and rewards online