Multi-Agent Planning with Interacting Actions

Shashank Shekhar

LIRMM & LAAS, CNRS

February 20, 2023

Automated Planning

- It is an explicit **deliberation process** of "selecting" and "organizing" actions by anticipating their (possible) outcomes.
- In automated planning, we study this explicit deliberation process computationally.

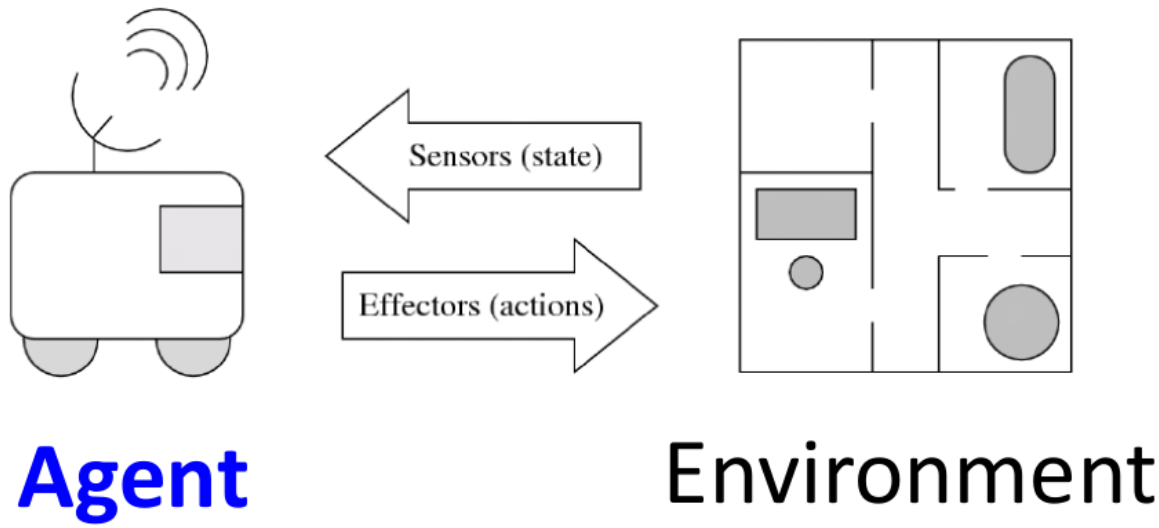
Talk Outline

- Basic terminologies for automated (multi-agent) planning
  - General motivation
  - Notations, problem specifications, models, etc.
- Planning with **interacting actions**
  - Compact representation from which one can quickly determine the effect of every joint action
  - MAP formalism, agents' actions interact - specifications
- Extension to human-robot interaction/collaboration
  - How does interacting actions in HRC challenge us differently?

Background

Single-Agent Planning

- Synthesizes a plan that satisfies a desired goal from the given initial situation
  - Centralized process
    - Done by a single planning entity or agent
  - Assumptions:
    - full-observability, instantaneous actions, deterministic outcomes, etc.



AI Planning: Key Concepts

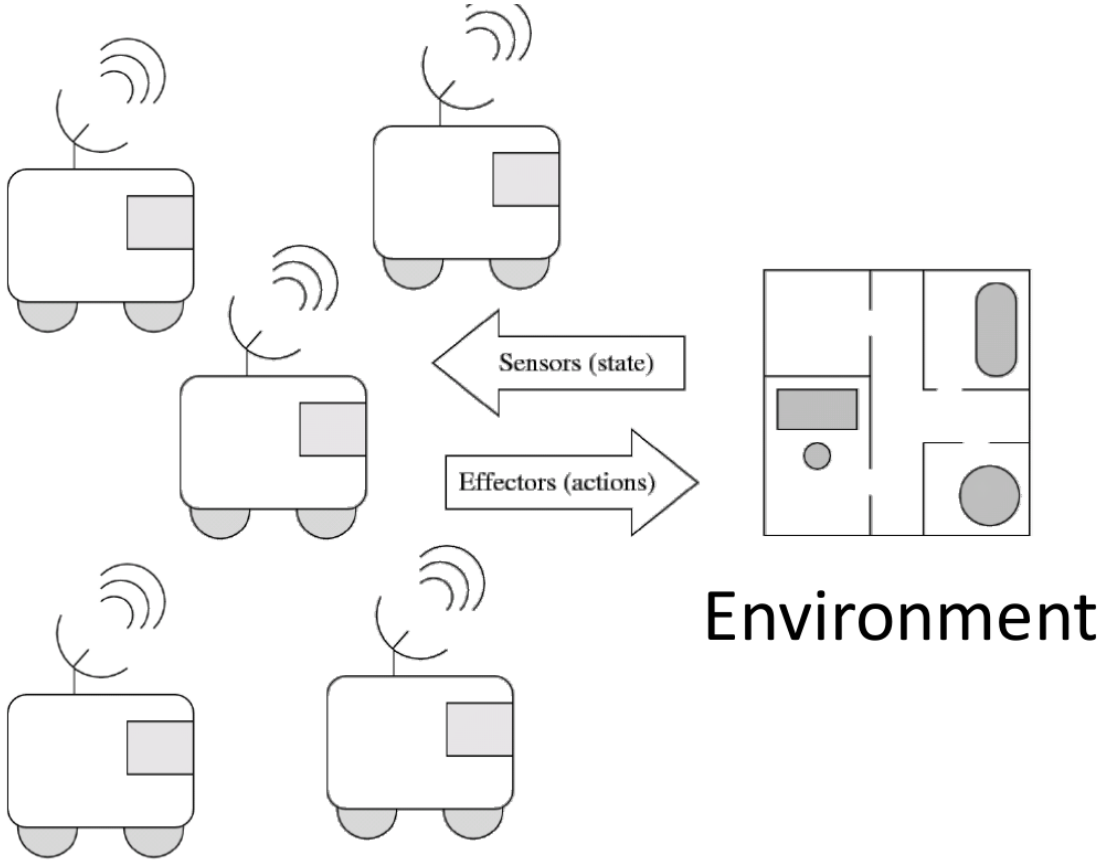
- **Language** – for describing a (planning) model implicitly
  - Plays two roles...
    - **Specification:** concise model description
    - **Computation:** reveals useful information about the problem's structure
  - **Examples:** STRIPS, SAS$^+$, PDDL, HDDL, etc.
- **Model** – describes the aspects of the world that needs to be captured
  - Should be intuitive
  - Could be very large and difficult to work with
    - We should be clear with what we want to solve
  - Captures the **assumptions** about the world, e.g., the **closed world assumptions**

AI Planning: Key Concepts

- **Interpretation** – maps b/w the expressions in the language and the model
  - Tells us how to build the whole search-space eventually
- **Example:** state model for classical AI planning
  - Finite state space S
  - An initial state $s_0 \in S$
  - A set $G \subseteq S$
  - Applicable actions $A(s) \subseteq A$ for $s \in S$
  - A transition function $s' = f(a, s)$ for all $a \in A(s), s \in S$
  - Cost function: $A^* \rightarrow [0, \infty)$
- **Query** – a question we wish to answer, e.g., find a plan
  - Planning algorithms

Multi-Agent Planning (MAP)

- MAP generalizes the problem of automated planning
- Several agents plan and act together by combining their knowledge, perspectives, capabilities, etc.



Why Multi-Agent Planning?

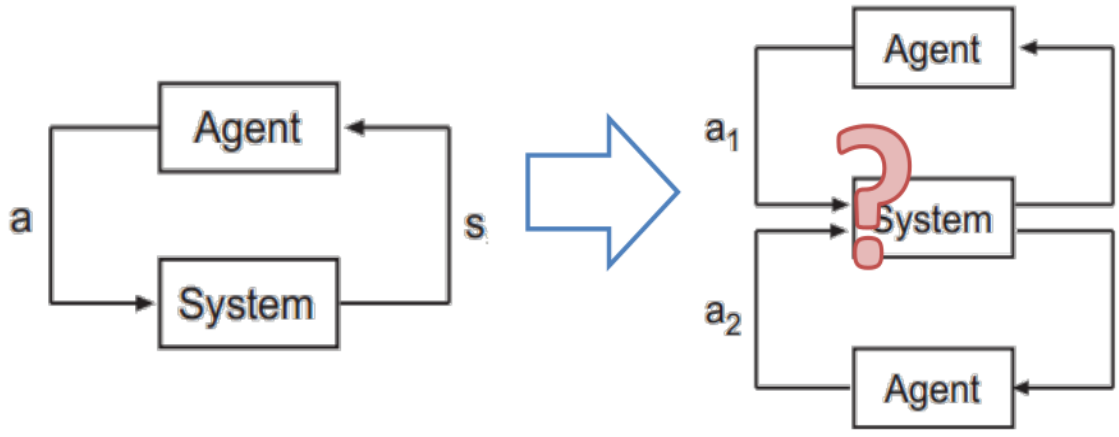
- Internet of Things (IoT)
- (Ad-hoc) Multi-Robot System & Human-Robot Collaboration (and Interaction)
- (Limited) collaboration among various entities
  - Examples: companies, contractors, etc.
- Useful abstraction for factored planning and distributed planning
  - Generates faster solutions for single-agent cases

...content taken from Ronen Brafman's talk at ICAPS 2016.

The Spectrum of MAP Models

- Who are the agents?
- Do they share capabilities?
- Can they act simultaneously (concurrently)?
- Who is handling the planning part?
- Orthogonal issues: uncertainty, time, resources, etc.

Our Focus: MA-STRIPS Model



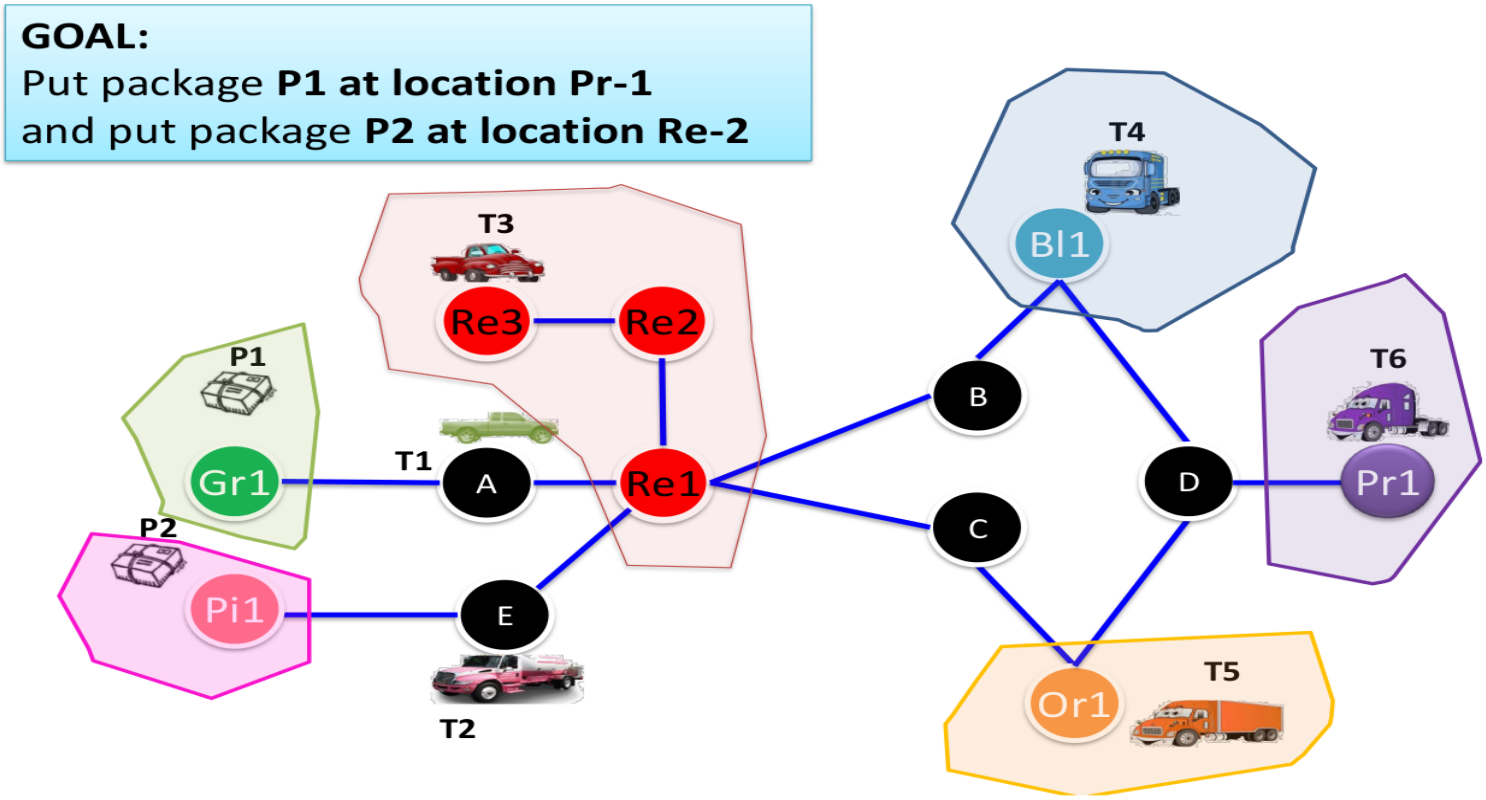
- **STRIPS** vs **Multi-Agent STRIPS** (minimalistic extension)
- **MA-STRIPS** associates each action with an agent
- $\langle P, A, I, G \rangle$ vs $\langle P, \{A\}_{i=1}^{k}, I, G \rangle$
- No interaction, cooperative agents, and easy to enhance

Logistics Example (Useful Abstraction)

- A **single-agent** planning domain with **natural components**
- To deliver packages to different places using vehicles
- Vehicles operate in different regions
- We model each vehicle as an agent (an abstraction)
- Agent's actions
  - **Move:** (move ?vehicle ?from ?to)
  - **Load/Unload:** (load ?pkg ?vehicle ?locA)

Factored Planning (based on Roni Stern's slides)

- AGENTS = **abstractions** (of these natural components)
- Do we gain anything by viewing this as a MAP problem?
  - From STRIPS to MA-STRIPS
  - Potential exponential gain!



Collaborative MAS - Short/Long Term Goals

- Our **short/long** term goals are to enhance the abilities of multi-agent systems consisting of robots (and also humans, sometimes) that are collaborative by nature
  - To efficiently and autonomously exploit their combined capabilities, knowledge, perspectives, etc.
  - By providing efficient *representations*, effective *models*, and *algorithms* for collaborative (task) planning and execution

Planning with Interacting Actions

Concurrent and Interacting Actions

- Sometimes the effect of an agent's action is **dependent** on what others are doing at that moment
- Agents **must coordinate** their actions carefully
- **Relevant scenarios**
  - If a table is "not lifted" from both sides **concurrently**, objects on it will fall
  - Pushing a heavy box:
    - An agent alone may be unable to push it
    - Multiple agents **simultaneously** can
  - In advanced RoboCup teams:
    - An agent passes the ball to a free region while the intended receiver moves to this area **simultaneously**
  - The Human-Robot handover task

Concurrent and Interacting Actions

- What happens when agents perform actions concurrently?
  - In principle, every combination of their actions defines a different transition model
  - Number of joint-actions is exponential in the number of agents
  - Impractical to explicitly model each joint action

Concurrent and Interacting Actions

There should be a way to deduce the effect of the concurrent execution $\langle a_1, a_2, ..., a_n \rangle$ of n agents

there should be a way to combine the smaller effects

- Can we use logical language? *(one way: a nonclassical logic)*
  - describing each combination by a set of formulas in it
  - each such set represents a unique transition model
- **Challenges:**
  - To describe a model for joint-actions
    - Given a large set of agents, and/or;
    - Large sets of individual actions
  - The model should
    - Be **succinct** in natural settings;
    - Support **efficient** planning

Relevant Part (ICAPS 2018 and AI Journal 2020)

1. An **intuitive formalism** for specifying joint-actions in a compositional manner
   - Relevant to collaborative and non-collaborative domains
2. An **effective compilation approach** to collaborative planning
   - Different schemes for different interaction types
3. **Experimental Evaluation**
   - New planning domains
   - Evaluation of our proposed compilation approach

...work done jointly with Ronen Brafman.

Related Work (Boutilier and Brafman, 2001)

- Boutilier and Brafman's (BB) approach was the first to extend STRIPS-like languages to address interacting actions
- **Example:** In the **box pushing** case, if two agents **push** a heavy box it moves, although the movement is conditional

```
Action: PushHeavy(agent, box, location)
        Pre: at(agent, location), at(box, location), heavy(box)
        Concurrency: (exists agent') PushHeavy(agent', box, location),
        agent' != agent
        Effect: NOT at(box, location).
```

- It is semantically clean and clear but has some drawbacks...
  - Non-standard *syntactic* requirement of concurrency
  - Schema generally requires existential quantifiers
    - **E.g.**, Agent's identity does not impact the interaction

Related Work

- Work in KR focus mainly on the representational issues, and they use non-monotonic reasoning (Poole 1997)
- Non-monotonic reasoning is hard to integrate with the modern planning algorithms
- The action language $\mathcal{A}_c$ (Baral and Gelfond 1997):
  - Use statements like "p is an effect of A, if c"
  - "p is also an effect of every $B \supseteq A$ given c", if there is no:
    - Other set of actions D such that $B \supseteq D \supseteq A$, and
    - $\neg p$ is an effect of D given c, which is non-monotonic in nature
- Simplest and intuitive, and closest to our formalism as well
- Our formalism is a **monotonic** variant of this action language

The Planning Model

- MAP model is tuple $\langle S, A, s_0, G, \Phi, \{A_i : 1 \leq i \leq n\}\rangle$
  - A is a set of **joint-actions**
- Formally, $a \in A$ is a vector of the form $\langle a_1, a_2, a_3, \ldots, a_n\rangle$
  - Each a_i is an action of agent φ_i or a *no-op*$_i$
- A plan $\pi = ja_1, ja_2, \ldots, ja_k$
  - It is a sequence of joint-actions
  - Such that $ja_k(\cdots(ja_1(s_0))) \in G$

Important Terminology

- **Collaborative Action**
  - A minimal combination of SA actions that cannot be defined as the union of its components
  - **E.g.**, $2push(a_1, a_2, B)$ – composed of two SA *push* actions, *s.t.*,

$$\boxed{eff(2push(a_1, a_2, B)) \neq eff(push(a_1, B)) \cup eff(push(a_2, B))}$$

- **Multi-Action**
  - Set of single-agent and collaborative actions (its *components*) with consistent preconditions and consistent effects
  - No agent participates in more than one action in this set

$$\textbf{E.g.}, \; a_m = \{2push(a_1, a_2, B), move(a_3)\} \text{ – for 3 agents}$$

  - Its **elements** are: SA actions + SA actions comprising the collaborative actions

$$\textbf{E.g.}, \; e(a_m) = \{push(a_1, B), push(a_2, B), move(a_3)\}$$

The Language

- Domain specification consists of $\langle P, I, g, \Phi, \{A_1, \ldots A_n\}, A_c \rangle$
  - P is a set of ground propositions
  - $I \subset P$, $g \subset P$, and Φ is a set of agents
  - A_i is a set of SA actions and A_c is a set of collaborative actions
- SA action has the form $a = \langle symbol, pre(a), eff(a) \rangle$, denoting the action's name, its preconditions and effects
- $A_c \ni a_c = \langle symbol, pre(a_c), eff(a_c), e = \{a_1, \ldots, a_k\} \rangle$ denotes a collaborative action
  - Element set $e(a_c)$, contains only SA action symbols
  - No two action symbols in $e(a_c)$ belong to the same agent

Interpretation

- Joint-actions **correspond** to **"good"** multi-actions
- Each multi-action $a_m = \{a_1, \ldots, a_k\}$ (some SAs, some CAs)
  - $e(a_m) = e(a_1) \cup \cdots \cup e(a_k)$
  - No two *elements* belong to the same agent
  - $pre(a_m) = pre(a_1) \cup \cdots \cup pre(a_k)$
  - $eff(a_m) = eff(a_1) \cup \cdots \cup eff(a_k)$
- **Subsumption ($\sqsupseteq$) Criteria** (informal)

$$\{push(a_1, B), push(a_2, B), move(a_3)\} \sqsubseteq \{2push(a_1, a_2, B), move(a_3)\}$$

  - Components can be combined to form a **more complex** collaborative action contained in other
  - They must contain **the same** element set

Interpretation

- **Box-Pushing Domain:**
  - Actions: *push*, *2push*, and *3push*
  - Multi-action $\{push(a_1, B), push(a_2, B), push(a_3, B)\}$:
    - Is subsumed by $\{2push(a_1, a_2, B), push(a_3, B)\}$
    - The above two are subsumed by $a_m = \{3push(a_1, a_2, a_3, B)\}$
- A **well-defined** multi-action is one that *never* gets subsumed
- Every **well-defined** multi-action a_m **defines** a joint-action

$$\{3push(a_1, a_2, a_3, B)\} \equiv \langle push(a_1, B), push(a_2, B), push(a_3, B)\rangle$$

  - $no\text{-}op_i$ is added to it for each **non-acting** agent φ_i

Compilation Approach

- Reduces centralized MA planning to centralized SA planning
- Different schemes depending on whether multi-actions contain **interfering components**
  - **Interfering components:** one deletes *precond* of the other
- **No interference:** all possible *interactions* are already captured by collaborative actions
  - Concurrent and sequential executions of the components produce the same effect
  - A sequential algorithm can be used, and later, a parallelization approach can be used to reduce makespan
- **Interference:** concurrent and sequential executions of components differ

  E.g., $sail(a_1, boat_1, l_1, l_2)$ and $sail(a_2, boat_1, l_1, l_2)$
  - Compilation is required

Pre/Eff Interactions: High-Level Steps

- Approach builds on [Crosby, Jonsson, and Rovatsos, 2014]
- Multi-action $\{a_1, \ldots, a_k\}$ is represented by action sequence $\langle a_{start}, a'_1, \ldots, a'_k, a_{end} \rangle$
- **Order of execution** of components of a well-defined multi-action is *unimportant*, therefore,
  - When emulating multi-action execution, must maintain the **same truth assignments** to the propositions as in state before a_{start}, but
  - Must keep track of their effects

Pre/Eff Interactions: High-Level Steps

- New bookkeeping propositions are added
  - Two new copies of every proposition: allow maintaining same "true" state throughout, while tracking component effects
  - Propositions that track which elements and components where done and by which agents: allow ensuring multi-action is consistent and well-defined
- Original actions modified to update added propositions
- Preconditions enhanced to ensure multi-action is well-defined
- Actions a_{start} and a_{end} are added to indicate start and end of the multi-action components
  - a_{end} updates true state and resets value of bookkeeping propositions

Empirical Evaluation

- No implemented algorithms to compare against
- No established MA-PDDL domains with interacting actions
- Two new domains and two updated ones with interacting actions, and the problem instances
  - Maze, TableMover, BoxPushing, ApartmentMover
  - We show only partial results
- The compilation approach generates centralized versions of the MAP domains

Results – centralized SA planning

Domain	Ins (#agents)	Length	Makespan	Time (sec)
Maze	P01 (3)	12	12	0.5
	P02 (4)	44	40	11.6
	P03 (4)	37	33	18.6
	P04 (5)	27	25	4.4
	P05 (4)	29	24	182.0
	P06 (5)	51	38	811.6
Table-Mover	P01 (3)	23	15	1.0
	P02 (4)	29	20	9.1
	P03 (4)	64	49	392.3
	P04 (5)	56	41	399.6
	P05 (5)	60	43	2753.5

Interacting actions: Quick Summary

- We proposed an **intuitive formalism** for planning with interacting actions
  - Specifies joint-actions in a compositional manner
  - Supports efficient planning as well
- **Irrationality**: semantics allows for *some cases* where the interpretation is not the intuitive one
- We **improved** (simplified) semantics and **refined** compilation to enforce it
  - A refined interpretation + new compilation approach to support efficient planning (AIJ 2020)
  - We quickly discuss it next...

Revised Interpretation

- **Box-Pushing Domain**
  - Actions: *push*, *2push*, and *3push*
  - Multi-action $\{push(a_1, B), push(a_2, B), push(a_3, B)\}$:
    - Is subsumed by $\{2push(a_1, a_2, B), push(a_3, B)\}$
    - The above two are subsumed by $a_m = \{3push(a_1, a_2, a_3, B)\}$
- A **well-defined** multi-action is **never** subsumed
- Consider the following two multi-actions:
  - $a_m^1 = \{2push(a_1, a_2, B), push(a_3, B)\}$;
  - $a_m^2 = \{push(a_1, B), 2push(a_2, a_3, B)\}$
  - $e(a_m^1) = e(a_m^2) = \{push(a_1, B), push(a_2, B), push(a_3, B)\}$
  - **(Collaborative) actions:** *push*, *2push*, and ~~*3push*~~
- Both multi-actions are well-defined

Revised Interpretation

Definition

A multi-action a_m is *well-formed* if no subset of its primitive elements $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\} \subseteq e(a_m)$ satisfies the following two conditions: (1) $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ contain primitive elements from *at least* two actions (components, i.e., *SA* and *CA*) in a_m. (2) There exists a collaborative action $a_c \in A_c$ such that $e(a_c) = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$.

- Consider $a_m = \{2push(a_1, a_2, b), push(a_3, b)\}$ (is well-defined)
  - $e(a_m) = \{\text{push}(a_1, b), \text{push}(a_2, b), \text{push}(a_3, b)\}$
  - But, $push(a_2, b)$ and $push(a_3, b)$ can be combined to form $2push(a_2, a_3, b)$
- a_m is not well-formed as per the above definition
- **(Skip)** the new translation scheme

Human-Robot Collaboration

Proposals based on Interacting Actions

HRC - Practical Assumptions

- Humans do not want to be controlled while working alongside robots
  - humans can be uncontrollable, but congruent, rational, etc.
- Both **proactive** (for some tasks/actions) and **lazy** (for some other) while collaborating
  - hard to model their intentions, mood, etc.
  - we can consider these as hidden factors that affect their decision making - can we estimate their effects?
  - we seek good planning models to capture such behaviors
- A collaborative **team** of humans and robots need to work together **to achieve shared goals** in many practical scenarios.
  - **examples:** service robots at homes, robots helping astronauts in space, (future) workshops, and many more...

Human-Robot Teaming

- For effective and efficient collaboration of humans and robots, the autonomy needs to **think** for the humans as well
- To consider *"human-focused perspectives"* about joint task
  - by developing strategies to understand/predict humans' intent, tasks, goals, etc.
- An **automated planning and decision making** view to understand such teaming considering *"human-awareness"*
- **Context**: how does the presence of the humans and robots challenge us differently than multi-robot planning?

HR Teaming - Concurrency & Shared Resources

- In principle, autonomy needs to **generate** contingent plans!
  - Why? (full-observability?)
  - Note that humans' actions can only be **estimated** that brings contingencies
  - Autonomy needs to **predict** the possible action interaction and take care of shared resources, i.e., *implicitly coordinated plans*

HR Teaming - Concurrency & Shared Resources

- **Joint action formation** — autonomy needs to **prioritize** and **emulate** humans' actions, followed by robots' *actions* then
- Concepts of well-definedness and well-formedness would change
- We would need a different (which is effective) representation to cater to issues like plan representation, joint-action formation, etc.
- Human operators bring challenges to plan execution, too

Thank You!