Acting in Dynamic Environments: Models of Agent-Environment Interaction

Lukáš Chrpa and **Pavel Rytíř** and **Rostislav Horčík**
Faculty of Electrical Engineering
Czech Technical University in Prague

Abstract

In real-world scenarios the environment is rarely static. Exogenous events might occur without the consent of the agent acting in such a dynamic environment. Events account for acts of nature without specific intentions, or actions of other agents having their own intentions.

Restricting on two actors, an agent applying actions and an environment applying events, we, in this paper, elaborate on five different models of how these actors can interact. For each actor, we assume full observability and full knowledge of actions and events. In particular, we focus on action-event conflicts that arise from the fact that one actor cannot control the other actor while concerning a classical and temporal settings.

Introduction

Automated Planning is an important tool for enabling deliberative reasoning of intelligent agents. However, many application domains consist of multiple actors, independent on each other, that can either willingly or unknowingly interfere with each other. One option is modifying the planning approach to handle multiple agents (Bowling, Jensen, and Veloso 2003; Brafman et al. 2009).

The MA-STRIPS formalism can be used to describe multi-agent planning tasks (Brafman and Domshlak 2008). In a nutshell, MA-STRIPS is an extension of STRIPS such that each agent has its set of actions and goals. The task is to find a plan for each agent such that plans of the agents do not collide. In non-cooperative settings, the existing techniques for generating non-conflicting multi-agent plans focus on *congestion games* (Jordán and Onaindia 2015; Jonsson and Rovatsos 2011). Another possibility is to try to prevent conflicts between agents' plans from occurring by imposing *social laws* (Nir and Karpas 2019). In adversary settings, agents deliberately try to harm each other (Applegate, Elsaesser, and Sanborn 1990; Pozanco et al. 2018) and therefore conflicts between agents' plans are usually unavoidable. Dynamicity of the environment, however, might not only be caused by actions of other intelligent agents but also by unintentional "acts of nature" that might also cause conflicts. Extensional definition of all combinations of actions and events (like in MDP or FOND) can be leveraged to handle conflicts but such a solution is rather impractical. Conflicts can also be handled ad hoc (Chrpa, Rytíř, and Horčík 2020), which is not very practical either.

Actions of other agents as well as acts of nature can be represented as non-deterministic exogenous events over which the agent has no control (Dean and Wellman 1990). There is a range of techniques that tackle plan generation and execution under presence of (non-deterministic) exogenous events. For example, there are techniques based on Markov Decision Process (MDP) (Kolobov, Mausam, and Weld 2012), Monte-Carlo Tree Search (Patra et al. 2019), reasoning about "dangerous states" (Chrpa, Gemrot, and Pilát 2017) or "safe states" and "robust plans" (Chrpa, Gemrot, and Pilát 2020).

In this paper, we consider a two-actor scenario, an agent that applies actions and an environment that applies events. The environment actor might be in form of another intelligent agent, an adversary, or a nature which acts without specific intentions. One actor cannot control actions or events of the other actor and vice versa. The key question this paper investigates is how both actors interact in the environment keeping in mind that both actors act simultaneously. Our paper, therefore, discusses possibilities how to "combine" application of actions and events in a more practical way. We design and discuss five models that describe how these actors interact during acting and each model is discussed in terms of strengths and limitations on three case studies we have considered. The key issue that each model has to somehow address is how to deal with action/event conflicts. One possibility, in the classical setting, is to alternate actions and events as in two-player games such as Chess (Chrpa, Pilát, and Gemrot 2019; Chrpa, Gemrot, and Pilát 2020). Another possibility is to apply actions and events in one step while conflicts are resolved by taking out incompatible actions or events. In the temporal setting, conflicts between actions and events might occur after an action or event in question already started being applied. In consequence, precondition of an action or event can be invalidated even after its application has started. Such conflicts can be handled by suspending an action or event until its precondition is satisfied again. The paper besides discussing expressive power of each described model also points out possible engineering challenges that may arise for particular models.

Technical Background

This section introduces the terminology we use in this paper.

Classical Planning

Let V be a set of ***variables*** where each variable $v \in V$ is associated with its domain $D(v)$. An ***assignment*** of a variable $v \in V$ is a pair (v, val), where its value $val \in D(v)$. Hereinafter, an assignment of a variable is also denoted as a ***fact***. A (partial) ***variable assignment*** p over V is a set of assignments of individual variables from V, where $vars(p)$ is a set of all variables in p and $p[v]$ represents a value of v in p. A ***state*** is a complete variable assignment (over V). We say that a (partial) variable assignment q ***holds*** in a (partial) variable assignment p, denoted as $p \models q$, if and only if $vars(q) \subseteq vars(p)$ and for each $v \in vars(q)$ it is the case that $q[v] = p[v]$.

An ***action*** is a pair $a = (\mathit{pre}(a), \mathit{eff}(a))$, where $\mathit{pre}(a)$ is a partial variable assignment representing a's precondition and $\mathit{eff}(a)$ is a partial variable assignment representing a's effects. We say that an action a is ***applicable*** in state s if and only if $\mathit{pre}(a)$ holds in s. The ***result*** of applying a in s is a state s' such that for each variable $v \in V$, $s'[v] = \mathit{eff}(a)[v]$ if $v \in vars(\mathit{eff}(a))$ while $s'[v] = s[v]$ otherwise.

A ***planning task*** is a quadruple $\mathcal{P} = (V, A, I, G)$, where V is a set of variables, A a set of actions, I a complete variable assignment representing the ***initial state*** and G a partial variable assignment representing the ***goal***.

Durative actions

To accommodate the notion of time, we consider a sequence of (discrete) timestamps to represent a timeline. Hereinafter, we denote that a state s holds in a timestamp t as $s(t)$. An ***action*** is a tuple $a = (\mathit{dur}(a), \mathit{pre}^{\vdash}(a), \mathit{pre}^{\vdash\dashv}(a), \mathit{pre}^{\dashv}(a), \mathit{eff}^{\vdash}(a), \mathit{eff}^{\dashv}(a))$, where $\mathit{dur}(a)$ represents duration of a's application and the other elements are sets of partial variable assignments. In particular, $\mathit{pre}^{\vdash}$ represents action precondition before its application, $\mathit{pre}^{\dashv}$ represents action precondition before finishing its application, $\mathit{pre}^{\vdash\dashv}$ represents action precondition for the whole time interval of its application, $\mathit{eff}^{\vdash}(a)$ represents action effects taking place after starting its application and $\mathit{eff}^{\dashv}(a)$ represents action effects taking place after finishing its application. We say that an action a is ***applicable*** in a state s in time t if and only if $\mathit{pre}^{\vdash}(a)$ holds in $s(t)$, $\mathit{pre}^{\dashv}(a)$ holds in $s(t + \mathit{dur}(a))$ and $\forall t' \in [t, t + \mathit{dur}(a)] : \mathit{pre}^{\vdash\dashv}(a)$ holds in $s(t')$. The ***result*** of applying a in a state s in time t (if possible) is that $\mathit{eff}^{\vdash}(a)$ becomes true in $s(t)$ and $\mathit{eff}^{\dashv}(a)$ becomes true in $s(t+\mathit{dur}(a))$. Note that by a partial variable assignment q *becoming true* in $s(t)$ we mean that for each $v \in vars(q) : s(t)[v] = q[v]$. We also consider *frame axioms*, that is, the values of variables that are not affected by effects of actions are carried forward to the next timestamp, i.e., from a state $s(t)$ to a state $s(t + 1)$.

Non-deterministic Events

Similarly to the definition of an action, a (non-deterministic) ***event*** is a tuple $e = (\mathit{pre}(e), \mathit{eff}(e))$, where $\mathit{pre}(e)$, $\mathit{eff}(e)$ are partial variable assignments representing e's precondition and effects respectively. Applicability of an event in a state as well as the result of an application of an event is defined in the same way as for actions.

We can also define a (non-deterministic) ***durative event*** as a tuple $e = (\mathit{dur}(e), \mathit{pre}^{\vdash}(e), \mathit{pre}^{\vdash\dashv}(e), \mathit{pre}^{\dashv}(e), \mathit{eff}^{\vdash}(e), \mathit{eff}^{\dashv}(e))$, where the elements of that tuple are analogous to those of a durative action.

Technically, a non-deterministic event can (but does not necessarily have to) occur in a state where event's preconditions are met modifying the state of the environment according to event's effects. The agent that is planning and executing its actions has no control over whether or not an event will occur if its precondition is met.

Relations between Actions and/or Events

In classical planning, it is often the case that a plan, a solution of a planning task, is a totally ordered sequence of actions transforming the environment from the given initial state to some state in which the goal is satisfied.

However, we might observe that independent actions can be applied in any order while resulting in the same state. The notion of independent actions (defined below) has been leveraged, for example, in the well known Graphplan algorithm (Blum and Furst 1997) that considers sets of independent actions to be applied in a single step, so the plan is in the form of a sequence of sets of independent actions.

Definition 1. *We say that actions a_i and a_j are* **independent** *if and only if $vars(\mathit{eff}(a_i)) \cap vars(\mathit{pre}(a_j) \cup \mathit{eff}(a_j)) = \emptyset$, $vars(\mathit{eff}(a_j)) \cap vars(\mathit{pre}(a_i) \cup \mathit{eff}(a_i)) = \emptyset$ and for each $v \in vars(\mathit{pre}(a_i)) \cap vars(\mathit{pre}(a_j))$ it holds that $\mathit{pre}(a_i)[v] = \mathit{pre}(a_j)[v]$. Independence of events is defined analogously.*

Speaking about situations in which two actions cannot be applied consecutively, we define the notion of a *cloberrer* (Chapman 1987) which is an action that invalidates the precondition of another action.

Definition 2. *We say that an action a_i is a* **cloberrer** *to an action a_j if and only if there exists $v \in vars(\mathit{eff}(a_i)) \cap vars(\mathit{pre}(a_j))$ such that $\mathit{eff}(a_i)[v] \neq \mathit{pre}(a_j[v])$. The relation of being cloberrer between events is defined analogously.*

Durative actions, on the other hand, explicitly consider time in which their precondition must hold and when their effects take place. Plans are in form of collections of pairs – timestamp, action – that provides an information when a given action is going to be applied. Therefore, it is common that more actions are being applied at the same time. It is however important that during their application actions do not *interfere* with each other.

Definition 3. *Let a_i and a_j be durative actions, where $d_i = \mathit{dur}(a_i)$ and $d_j = \mathit{dur}(a_j)$. We say that a_i and a_j (in this order) are in*

- 0-**effect-effect conflict** *iff* $vars(\mathit{eff}^{\vdash}(a_i)) \cap vars(\mathit{eff}^{\vdash}(a_j)) \neq \emptyset$

- d_i-**effect-effect conflict** *iff* $vars(\mathit{eff}^{\dashv}(a_i)) \cap vars(\mathit{eff}^{\vdash}(a_j)) \neq \emptyset$
- $(d_i - d_j)$-**effect-effect conflict** *iff* $vars(\mathit{eff}^{\dashv}(a_i)) \cap vars(\mathit{eff}^{\dashv}(a_j)) \neq \emptyset$
- 0-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\vdash}(a_i)) \cap vars(\mathit{pre}^{\vdash}(a_j)) \neq \emptyset$
- $-d_j$-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\vdash}(a_i)) \cap vars(\mathit{pre}^{\dashv}(a_j)) \neq \emptyset$
- $[-d_j, 0]$-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\vdash}(a_i)) \cap vars(\mathit{pre}^{\vdash\dashv}(a_j)) \neq \emptyset$
- d_i-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\dashv}(a_i)) \cap vars(\mathit{pre}^{\vdash}(a_j)) \neq \emptyset$
- $d_i - d_j$-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\dashv}(a_i)) \cap vars(\mathit{pre}^{\dashv}(a_j)) \neq \emptyset$
- $[d_i - d_j, d_i]$-**effect-precondition conflict** *iff* $vars(\mathit{eff}^{\dashv}(a_i)) \cap vars(\mathit{pre}^{\vdash\dashv}(a_j)) \neq \emptyset$

In plain words, the above definition means that actions a_i and a_j (in this order) are in k-effect-effect conflict if applying a_j k time units after applying a_i results in a situation that both a_i and a_j try to modify at least one variable at the same time. We also consider situations in which two or more actions modify a variable at the same time to the same value as an effect-effect conflict too. Similarly, if actions a_i and a_j (in this order) are in k-effect-precondition conflict, then if a_j is applied k time units after a_i (or before a_i if k is negative), a_i modifies at least one variable at the same time a_j requires the variable to have a certain value. Note that $[k, l]$-effect-precondition conflict is a shortcut for $\forall t \in [k, l] : t$-effect-precondition conflict.

In general, let a_i be an action applied at time t and a_j be an action applied at time $t + k$. We say that a_i and a_j (in this order) are in *conflict* if and only if a_i and a_j are in k-effect-effect or k-effect-precondition conflict. Otherwise, we say that a_i and a_j (in this order) are *conflict-free*.

PDDL 2.1 semantics (Fox and Long 2003), roughly speaking, deals with conflicts by "ϵ-shifts", that is, shifting action application by ϵ, which is usually a very small constant, in order to avoid conflicts with other actions. It is however not very practical for discretized timelines. Another approach, primarily designed for temporal planning with SAT modulo Theories, is tailored for discretized (integer) timelines (Rintanen 2015).

Inspired by Rintanen's approach, we consider the following action execution model. Let $\{(a_1, k_1), \ldots, (a_n, k_n)\}$ be a set of actions with a relative time of their application. We also assume that for each a_i it is the case that $k_i \leq \mathit{dur}(a_i)$ and all actions are pairwise conflict-free (with respect to their application time). Note that k_i might be negative which means that the action a_i will be applied after $|k_i|$ time units. In a current state $s(t)$, we initially check whether preconditions of the actions are met (if not we fail) and then we apply action effects and move to a state $s(t + 1)$. In particular, the precondition check is done as follows:

- for each a_i with $k_i = 0$ we check whether $s(t) \models \mathit{pre}^{\vdash}(a_i)$
- for each a_i with $k_i = \mathit{dur}(a_i)$ we check whether $s(t) \models \mathit{pre}^{\dashv}(a_i)$
- for each a_i with $0 \leq k_i \leq \mathit{dur}(a_i)$ we check whether $s(t) \models \mathit{pre}^{\vdash\dashv}(a_i)$

Action effects are applied such that $s(t + 1) = \bigcup_{\{i|k_i=0\}} \mathit{eff}^{\vdash}(a_i) \cup \bigcup_{\{i|k_i=\mathit{dur}(a_i)\}} \mathit{eff}^{\dashv}(a_i)$ and for each variable $v \notin vars(s(t + 1))$ we apply the frame axiom $s(t + 1)[v] = s(t)[v]$. Then, all actions a_i having $k_i = \mathit{dur}(a_i)$ are removed from the set while the k values of the remaining actions are incremented by 1.

Note that all the notions defined above can be analogously extended to events (as well as between actions and events).

Case Studies

In this section, we will introduce case studies that will be used throughout the paper.

AUV Domain

Inspired by the recent work of Chrpa et al. (2015) concerning task planning for Autonomous Underwater Vehicles (AUV), Chrpa, Gemrot, and Pilát (2020) designed an "AUV domain" that simulates the situation where an AUV has to perform sampling of some objects of interest while there might be ships passing by that might endanger the AUV. We have a 4-grid environment, an AUV, ships and several resources. Resources can be found on given cells. Each cell is either free, has the AUV on it, or the ship on it (presence of a resource does not interfere with any cell status). The AUV can move to an adjacent cell, if the cell is free. The AUV can sample a resource if it is at the same cell. The task for the AUV is to sample the resources and return back to the place of origin.

Ships, however, are not controlled by the agent, i.e., ships are controlled by the environment. Ships can move only on some cells from the grid or might not be present in the area. Each ship can enter the area at its entry cells, can move to adjacent cells it is allowed to move, and leave the area at its exit cells. A ship can appear in its entry cell, if the ship is not already in the area. A ship can leave the area, if it is in its exit cell. Two "move" events are considered, move-ship-to-free and move-ship-to-auv. Both require that the ship can move to the destination cell. The effect of both events is that the ship moves to the destination cell. If the ship moves to a free cell, then besides the cell becoming not free for a moment, nothing else happens. However, if the ship moves to the cell with the AUV, then the AUV is destroyed (and can no longer perform any action).

Dark Dungeon Domain

Dark Dungeon Domain has been introduced by Chrpa, Gemrot, and Pilát (2017). The game is played on a map consisting of rooms and corridors connecting the rooms. A single hero is present in the map whose goal is to get to a specified room. However, there are also monsters, which can threaten and kill the hero, and traps that might kill the hero if they are not disarmed immediately. The hero can find a sword in the map and use it to kill monsters. The rules of the game are summarised in the following points:

- In each step, the hero can make one of the following actions (or do nothing):
  - move to a neighbouring room,
  - pickup a sword, if the sword is in the same room as the hero,
  - drop a sword, if the hero is holding it,
  - disarm a trap, if the hero does not have a sword and is in the same room as the trap.
- If the hero is in the same room as a monster and does not have a sword, the hero is killed. If the hero has the sword, the monster is killed.
- If the hero is in a room with trap, any other action than "disarm trap" cannot be performed.
- If the hero does not immediately disarm the trap, the trap can trigger and kill the hero.
- Disarming a trap or killing a monster removes them from the map.
- Monsters cannot move to a room with a trap, with another monster, or with the hero carrying a sword.

Resource Hunting Domain

We introduce a simplification of a two-player zero sum game called Resource Hunting, recently specified by Rytíř, Chrpa, and Bošanský (2019). The map of the game is modelled as a graph in which the vertices represent locations of interest and edges connect neighbouring locations. The (soft) goals for each player are to collect resources that are placed in some locations on the map. Each player controls a group of unmanned aerial vehicles (UAVs).

There are two types of actions each player can take: the *move* action, which moves an UAV from one location to another such that the locations are connected by an edge, and the *collect* action, where UAVs collect a resource present at the same location as the UAVs.

After a resource is collected it is no longer available. If both players attempt to collect the same resource at the same time, then only one player succeeds (there is a equal chance for both players to succeed).

We also introduce a *combat variant* of the domain in which at most one UAV can be at the same location at the same time and an UAV might shoot down a competitor's UAV if their current locations are at proximity to each other.

Execution Models

In classical planning, the task is to find a partially or totally ordered sequence of actions, a plan, that transforms the environment from the initial state to one of the goal states. Since classical actions do not consider duration of their application, we assume that the actions in the plan are applied (or executed) consecutively, following the ordering constraints.

With (non-deterministic) events in play, however, the plan execution model becomes less straightforward as, given no notion of time, it is not implicit when events might occur. One possible solution for the issue is inspired by two-player games in which players' turns alternate (e.g. Chess or Checkers). In our terminology, it means that during plan execution actions of the agent alternate with non-deterministic events of the environment (Chrpa, Pilát, and Gemrot 2019). Such a solution, however, is a considerable oversimplification that can be reasonably used only in very limited situations (e.g. one AUV and one ship).

Inspired by the Graphplan approach (Blum and Furst 1997) in which plans are in form of sequences of sets of independent actions that can be applied simultaneously, we amend the above idea such that sets of independent actions of the agent alternate with sets of independent (non-deterministic) events of the environment. It should be noted that such a model is an extension of the model of Chrpa, Gemrot, and Pilát (2019; 2020) in which only a single action was considered in each step.

Model 1. *Let A be a set of agent's actions and E be a set of events. In a state s, the agent applies a set of independent actions $A^i \subseteq A$ (might be empty) such that each $a \in A^i$ is applicable in s, that results in a state s'. After that, a set of independent events $E^i \subseteq E$ (might be empty), where each $e \in E$ is applicable in s', is applied.*

One issue with Model 1 is the assumption that only independent actions/events can be performed in a single step. To give an example, we can have a situation where two AUVs (or monsters) want to move simultaneously such that the first AUV (AUV1) moves from a location $l1$ to a location $l2$ while the second AUV (AUV2) moves from $l2$ to a location $l3$. Such move actions are not independent as AUV2 makes $l2$ free while AUV1 makes $l2$ occupied again. On top of that the move action for AUV1 is not applicable as AUV2 occupies $l2$ that the move action requires to be free.

To deal with such an issue we introduce the notion of *action compatibility* (and *event compatibility*) that in a nutshell specifies whether actions (events) can be applied simultaneously in a single step.

Definition 4. *Let A be a set of actions and $comp_A \subseteq A \times A$ be a relation over a pair of actions. We say that $comp_A$ is a relation of* **action compatibility** *over A in and only if i) for all $a_i, a_j \in A$ being independent it is the case that $(a_i, a_j) \in comp_A$ and $(a_j, a_i) \in comp_A$, and ii) for each $(a_i, a_j) \in comp_A$ it is the case that a_i is not a cloberrer for a_j. The relation of* **event compatibility** *over a set of events E, $comp_E$, is defined analogously.*

The formal definition of action/event compatibility says that if actions/events are independent they are compatible (i.e., i) is a sufficient condition for compatibility) while an action/event cannot be a cloberrer for the other action/events in order to be considered as compatible (i.e., ii) is a necessary condition). The latter condition gives a degree of freedom for a domain engineer to decide for each pair of actions satisfying only the condition ii) whether they are considered as compatible or not.

To illustrate the notion of action compatibility, we extend our above example by adding the third AUV (AUV3) that wants to move from a location $l4$ to $l2$. We can see that moving AUV2 and AUV1 (in this order) is compatible as well as moving AUV2 and AUV3 (in this order) is compatible. On

the other hand, moving AUV1 and AUV3 is not compatible as moving AUV1 is a clobberer for moving AUV3 (and vice versa) since both actions make $l2$ occupied while both actions require $l2$ to be free. Also, for example, two consecutive move actions of a single AUV follow the necessary condition, however, they intuitively cannot be performed in a single step and thus should not be considered as compatible.

The applicability and the result of application of compatible actions/events is defined as follows.

Definition 5. *Let A be a set of actions and $comp_A$ be a relation of action compatibility over A. We say that a set of actions $A^c \subseteq A$ is* **compatible** *if and only if for each $a_i, a_j \in A^c$, where $a_i \neq a_j$, it is the case that $(a_i, a_j) \in comp_A$ or $(a_j, a_i) \in comp_A$. We say that A^c is* **applicable** *in a state s if and only if for each action $a \in A^c$ it is the case that for each $v \in vars(pre(a)) : pre(a)[v] = s[v]$ or $pre(a)[v] = eff(a')[v]$ for some $a' \in A^c$, where $(a', a) \in comp_A$. The* **result** *of applying A^c in s is a state s' such that $s'[v] = s[v]$ for each $v \notin \bigcup_{a \in A^c} vars(eff(a))$, $s'[v] = eff(a)[v]$ where $a \in A^c, v \in vars(eff(a))$ and there does not exist $a' \in A^c$ such that $v \in vars(eff(a'))$, $eff(a')[v] \neq eff(a)[v]$ and $(a, a') \in comp_A$. A set of compatible events, its applicability and the result of its application is defined analogously.*

We can extend Model 1 by considering sets of compatible actions/events rather than sets of independent action/events as follows.

Model 2. *Let A be a set of agent's actions and E be a set of events. In a state s, the agent applies a set of compatible actions $A^c \subseteq A$ (might be empty) such that A^c is applicable in s, that results in a state s'. After that, a set of compatible events $E^c \subseteq E$ (might be empty), where E^c is applicable in s', is applied.*

Model 2 is more expressive than Model 1 since sets of independent actions/events are also sets of compatible actions/events. The reverse implication does not generally hold.

Focusing on the aspect of alternating action and event turns that Model 1 and 2 introduce, we analyse our case studies in terms of whether Model 1 or 2 supports the specified requirements for each domain.

For the AUV domain, both models reasonably describe possible interaction between AUVs and ships. If an AUV moves into a cell in front of a ship in the action turn, then ship might run over it in the event turn. If a ship blocks the cell an AUV wants to move, the AUV has to wait until the ship moves away. Although it might be possible that the AUV and the ship move simultaneously, the AUV might not know whether the ship will move or not and hence it has to wait anyway. The requirements specified in the domain description are hence met by Model 2 and by Model 1 for a single AUV scenarios where ships do not "share" their cruising corridors.

In Dark Dungeon, we have some issues. Firstly, after the hero enters a room with a trap (the action turn), the trap might trigger and kill the hero (the event turn) without giving the hero a chance to disarm the trap. Secondly, the hero carrying a sword can move to the room with a monster and kill it in the action turn while leaving no opportunity for the monster to escape (in the event turn). Hence, the requirements specified in the domain description are neither met by Model 1 nor by Model 2.

In Resource Hunting, one issue is when an agent's UAV and a competitor's UAV want to collect a resource at the same time. The agent goes first and collects the resource in the action turn. Then, the competitor can no longer collect the resource in the event turn. Thus the player whose turns are before the other player's turns is in advantage and hence the requirements specified in the domain description are neither met by Model 1 nor by Model 2.

The intuitive way how to handle the above issues is to apply actions and events simultaneously in a single step. We can extend Definition 4 to consider compatibility between actions and events, i.e., $comp_{AE} \subseteq (A \cup E) \times (A \cup E)$ (A is a set of actions while E is a set of events). This is, however, only a partial solution for the above issues. It solves, for example, the problem where a monster escapes the armed hero. The other issues, however, are related to incompatibility between actions and events.

Since the agent has no control over what events the environment selects and analogously the environment has no control over what actions the agent selects, there has to be some mechanism that decides which actions or events will prevail over the incompatible events or actions. As indicated in the above examples, we have a couple of ways how the action/event conflicts have to be resolved. In Dark Dungeon, the "disarm-trap" action has to always be preferred against the "trigger-trap" event. On the other hand, in Resource Hunting, the "collect" action has to have an equal chance as the "collect" event. Following the intuition we define the action choice function as follows.

Definition 6. *Let A and E be sets of actions and events respectively. We define an* **action choice** *(partial) function $ac : A \times E \to [0, 1]$ such that for each $a \in A$ and $e \in E$ where $ac(a, e)$ is defined it represents that a will be preferred against e by probability $ac(a, e)$. We assume that $ac(a, e)$ is defined for each $a \in A, e \in E$ such that $(a, e) \notin comp_{AE}$ and $(e, a) \notin comp_{AE}$.*

In general, the action choice function as defined above might not properly describe situations in which more than one compatible actions are pairwise incompatible with more than one compatible event. To do so, the action choice function would have to be defined on sets of actions and events.

Hence, we introduce restrictions that the model has to comply with. For each set of compatible actions $A^c \subseteq A$ and each set of compatible events $E^c \subseteq E$ it is the case that for each $a \in A^c, e \in E^c$ such that $ac(a, e)$ is defined: i) there does not exist $a' \in A^c, a' \neq a$ or $e' \in E^c, e' \neq e$ such that $ac(a', e)$ or $ac(a, e')$ is defined, and ii) for each $a' \in A^c, a' \neq a$ it is the case that a, a' are independent and for each $e' \in E^c, e' \neq e$ it is the case that e, e' are also independent.

The following model considers applying actions and events in a single step. The idea is that both agent and environment make their selection of compatible actions and

events. Then, for actions and events that are not pairwise compatible, the action choice function determines what actions or events will prevail or be eliminated. Note that the following model considers that above restrictions.

Model 3. *Let A be a set of agent's actions, E be a set of events, and s be a state. Let ac be the action choice function, $A^c \subseteq A$ be a set of compatible actions and $E^c \subseteq E$ be a set of compatible events such that $A^c \cup E^c$ is applicable in s. Then, let $X = \{x \mid ac(a,e)$ is defined, $P(x = e) = ac(a,e), P(x = a) = 1 - ac(a,e)\}$ (x is a random variable) be a set of actions and events eliminated from $A^c \cup E^c$. After that, $(A^c \cup E^c) \setminus X$ is applied in s.*

Model 3 can reason with situations such as in Dark Dungeon where according to the specification the hero gets a chance to disarm a trap before it can trigger, a monster can escape the armed hero, or, in Resource Hunting, in which two competing players try to collect a resource at the same time. In particular, the move of the armed hero and the move of the monster is compatible (and hence can be applied in a single step), then, the value of the ac function for the "disarm-trap" action will be 1 while for the "collect" action will be 0.5. However, in the combat variant of the Resource Hunting domain, the above restrictions would prevent to correctly model situations an agent simultaneously moves more UAVs "in chain" (e.g. one from $l1$ to $l2$, one from $l2$ to $l3$ etc.) while an event might block one of the locations. Intuitively, if the event is selected and thus blocks agent's UAVs, more actions (all in the "chain") are affected and cannot be applied in the given step. Such a situation violates the condition ii) of the above restriction.

Model 3 is however problematic in situations such as moving an AUV in front of a ship that at the same time moves and runs over the AUV. We might assume that "move-ship-free" and "move-ship-auv" are two different events where the former represents that the ship moves to a free cell while the latter represents that the ship moves to the cell occupied by an AUV. The latter event is, however, inapplicable at the time of event selection since one of its precondition can be satisfied only by the "move-auv" action. Similar issue can be observed in the Dark Dungeon domain in situations in which the hero and a monster move at the same time to the same room.

The above issue can be addressed by considering conditional effects that, in fact, allow to "merge" actions or events into one. For example, "move-ship-free" and "move-ship-auv" can be merged into "move-ship" with a conditional effect in form "if an AUV is in the destination cell, then the AUV is no longer operational".

An alternative solution to the above issue can be introducing a "replacing" function such that if a given action is selected together with a given event, then either the action or the event is replaced by the "alternative" action or event respectively. That is, if both "move-auv" and "move-ship-free" having the same destination location are selected, then the "move-ship-free" event is replaced by the corresponding "move-ship-auv" event. Formally, the ***replacing function*** is $rep : (A \cup E) \times (A \cup E) \to (A \cup E)$ such that $rep(x, y) = z$ specifies that if x and y are selected for being applied in the same step, then x is replaced by z. Also, for each x, y_1, y_2, z_1, z_2, where $y_1 \neq y_2$ and $z_1 \neq z_2$, such that $rep(x, y_1) = z_1$ and $rep(x, y_2) = z_2$, it is the case that y_1 and y_2 are not compatible. Such a condition will prevent ambiguities that might arise, for instance, if two events are selected where for each of the events a different replacement of a selected action is defined.

Model 4. *Let rep be a replacing function and Y be the set of actions and events as in Model 3. Then, we calculate $Y' = Y \setminus \{x \mid rep(x, y) = z; x, y \in Y\} \cup \{z \mid rep(x, y) = z; x, y \in Y\}$ and apply it in the current state s.*

Model 4 hence deals with the above issues in the AUV and Dark Dungeon domains.

Durative Actions and Events

With explicit reasoning about time, the issues with action/event independence or compatibility are substituted by selecting durative actions/events that are conflict-free. Whereas we can reasonably assume that the agent will select and apply actions in the conflict-free fashion as well as the environment will do so with events, conflicts might arise between actions and events.

For situations in which actions and events are in conflict and are selected to be applied at the same time while being incompatible (by means that either an action or an event can be applied), we can leverage the action choice function analogously to Model 3. Therefore, we can handle conflicts between "disarm-trap" and "trigger-trap" or between the "collect" action and the "collect" event in the same way as discussed earlier. Similarly, we can leverage the replacing function for actions and events which start at the same time analogously to Model 4.

However, a straightforward adaptation of Model 4 to durative actions and events will not resolve all conflicts and, moreover, will not prevent situations in which an action invalidates a precondition of a currently applied event or vice versa. The latter can be resolved by either terminating application of an action or event whose precondition becomes invalidated, or by suspending its application until the precondition is met again. In the combat variant of the Resource Hunting domain, shooting down an UAV while it is collecting a resource essentially terminates the action application. On the other hand, blocking a destination location by a competitor's UAV might only suspend the agent's move action. Technically, we can consider action/event application termination as its suspension for infinite time.

Resolving conflicts between actions and events can be done similarly to resolving incompatible actions and events.

Definition 7. *Let A and E be sets of durative actions and events respectively. We define a* **conflict resolution** *(partial) function $cr : A \times E \times \mathbb{Q} \to [0, 1]$ such that $cr(a, e, k)$ is defined for each $a \in A$, $e \in E$ and k such that a and e or e and a are in k-effect-effect/precondition conflict. It means that effects/preconditions of a will prevail over e (with respect to k) with probability $cr(a, e, k)$.*

The following model wraps up the above concepts as follows.

Model 5. *Let A be a set of agent's durative actions, E be a set of durative events. Let ac be the action choice function, rep be the replacing function, and cr be the conflict resolution function. Let $s(t)$ be the current state in time t. For actions and events that start their application in time t we consider ac (the action choice function) and rep (the replacing function) analogously to Model 3 and Model 4 respectively. If a precondition of an action/event is not met in t, then the application of the action/event is suspended (the relative time of its application is not incremented). Conflicts between actions and events arising in time t are resolved according to cr (the conflict resolution function) such that in the case of effect-precondition conflicts, a precondition of an action/event which "lost" is considered as unsatisfied (and the action/event is suspended), while effects of an action/event which "won" take place (applies also for effect-effect conflicts).*

Model 5 to a large extent correctly captures the required behaviour for all the considered domains. However, there are several aspects that are problematic. Firstly, situations such as an AUV and a ship simultaneously moving to the same cell are not properly handled. The "move-ship-AUV" event requires "at-auv=cell" before it finishes its application. The "move-AUV" action sets "at-auv=cell" at the end of its application. Moreover, "move-AUV" requires "free-cell=true" while "move-ship-AUV" sets it as false. Assuming that both finish their application at the same time, they are in conflict. "Move-AUV" shall prevail, setting "at-auv=cell" while "move-ship-AUV" becomes suspended. At the next step, "move-ship-AUV" can be resumed and its effects are free to take place. Note that the finishing "move-ship-AUV" action shall prevail over any of starting "move-AUV" actions from the same cell. Whereas the behaviour of simultaneous application of "move-ship-AUV' and "move-AUV" results in desired effects (the AUV is destroyed by the ship), the process will take one more time unit. Similar observations can be made in Dark Dungeon in which the hero might "collide" with monsters.

Suspending application of actions and events might be desirable as, for example, an UAV might wait for a location to become free. However, it can easily happen that multiple UAVs will be waiting for a single location to become free. Therefore, additional conflicts might arise (and are not limited to action-event conflicts).

Discussion and Conclusion

Planning and acting in dynamic environments poses numerous challenges. One of the challenges, we discuss in this paper, concerns modelling of interaction between actions and events. We have considered a two-actor model, an agent that applies actions and an environment that applies events. An actor cannot control how the other actor will act, on the other hand, we assume that each actor has full observability of the environment and full knowledge of the actions and events that might be applied. The key issue regarding the action and event application is dealing with action/event conflicts.

In the "classical" variant, where actions and events do not explicitly take time into consideration, action and event application might alternate as in a two-player game such as Chess. Such a model of the interaction between the agent and the environment is rather simplistic and might be feasible only for some domains (e.g. the AUV domain). On the other hand, alternating application of sets of independent actions and sets of independent events (i.e., Model 1) does not require additional information to be provided alongside a domain model. Model 4 provides more realistic interpretation of the interaction between the agent and the environment and is thus feasible for a larger set of domains. On the other hand, additional information has to be provided on top of domain models. The relation of action/event compatibility is stronger than the relation of independence. However, there is a degree of freedom in specifying which pairs of actions/events following only the necessary condition (and not the sufficient condition) should be considered as compatible. The action choice function for pairs – action, event – that are not compatible has to be specified as well (if the action choice function follows the specified restrictions can be verifies automatically). Also, one has to specify a replacing function. Whereas in the example domains providing such information might be rather straightforward, in some other and more complicated domains much more engineering effort might be necessary.

In the temporal variant, actions and events might get into conflicts with each other. Whereas in cases of "at start" conflicts it is possible to resolve them by leveraging the action choice function such as in the classical variant, conflicts that arise during action/event application are more complicated to handle. Specifying the conflict resolution function seems to require much more engineering effort than specifying the action choice function. Model 5 provides an interpretation of the interaction between the agent and the environment with some limitations. While some limitations are rather small (e.g. requiring one more time unit to deal with AUV/ship or hero/monster conflict), other limitations seem to be larger (e.g. more UAVs waiting for a location to become free).

As a rule of thumb, it is advisable to encode the domain model in such a way that domain requirements are met (or the discrepancies are reasonably small) while needing the simplest possible model of actor interaction. That means, in consequence, minimising engineering effort for the model while keeping its required expressivity. On top of that, we believe (albeit do not discuss in detail) that techniques which can be used for reasoning might be simpler and computationally cheaper for simpler actor interaction models.

In future, we plan to further investigate conditions which temporal domain models have to meet to prevent undesirable action/event suspending. Also, we plan to study how existing techniques such as dangerous state reasoning (Chrpa, Gemrot, and Pilát 2017) or robust plan generation (Chrpa, Gemrot, and Pilát 2020) have to be adapted for different models of actor interaction.

Acknowledgements This research was funded by AFOSR award FA9550-18-1-0097 and by the Czech Science Foundation (project no. 18-07252S).

References

Applegate, C.; Elsaesser, C.; and Sanborn, J. C. 1990. An architecture for adversarial planning. *IEEE Trans. Systems, Man, and Cybernetics* 20(1):186–194.

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artif. Intell.* 90(1-2):281–300.

Bowling, M.; Jensen, R.; and Veloso, M. 2003. A formalization of equilibria for multiagent planning. In *IJCAI*, 1460–1462.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 28–35.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2009. Planning games. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Chapman, D. 1987. Planning for conjunctive goals. *Artif. Intell.* 32(3):333–377.

Chrpa, L.; Pinto, J.; Ribeiro, M. A.; Py, F.; de Sousa, J. B.; and Rajan, K. 2015. On mixed-initiative planning and control for autonomous underwater vehicles. In *IROS*, 1685–1690.

Chrpa, L.; Gemrot, J.; and Pilát, M. 2017. Towards a safer planning and execution concept. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, 972–976.

Chrpa, L.; Gemrot, J.; and Pilát, M. 2020. Planning and acting with non-deterministic events: Navigating between safe states. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 9802–9809.

Chrpa, L.; Pilát, M.; and Gemrot, J. 2019. Compiling planning problems with non-deterministic events into FOND planning. In *Joint Proceedings of the RCRA International Workshop and of the RCRA Incontri e Confronti Workshop co-located with the 18th International Conference of the Italian Association for Artificial Intelligence (AIIA 2019), Rende, Italy, November 19-20, 2019*.

Chrpa, L.; Rytíř, P.; and Horčík, R. 2020. Planning against adversary in zero-sum games: Heuristics for selecting and ordering critical actions. In *Proceedings of the Thirteenth International Symposium on Combinatorial Search, SOCS 2020, Online Conference [Vienna, Austria], 26-28 May 2020*, 20–28.

Dean, T., and Wellman, M. 1990. *Planning and Control*. Morgan Kaufmann Publishers.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20:61–124.

Jonsson, A., and Rovatsos, M. 2011. Scaling up multiagent planning: A best-response approach. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS*.

Jordán, J., and Onaindia, E. 2015. Game-theoretic approach for non-cooperative planning. In *the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 1357–1363.

Kolobov, A.; Mausam; and Weld, D. S. 2012. LRTDP versus UCT for online probabilistic planning. In *AAAI*.

Nir, R., and Karpas, E. 2019. Automated verification of social laws for continuous time multi-robot systems. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 7683–7690.

Patra, S.; Ghallab, M.; Nau, D. S.; and Traverso, P. 2019. Acting and planning using operational models. In *The 33rd AAAI Conference on Artificial Intelligence*, 7691–7698.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 4808–4814.

Rintanen, J. 2015. Discretization of temporal models with application to planning with SMT. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, 3349–3355.

Rytíř, P.; Chrpa, L.; and Bošanský, B. 2019. Using classical planning in adversarial problems. In *the IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 1327–1332.