The Power of Waiting in Social Laws

Alexander Tuisov, Alexander Shleyfman, Erez Karpas
Technion, Israel
alexandt@campus.technion.ac.il, alesh@technion.ac.il, karpase@technion.ac.il

Abstract

Agents operating in the same environment as other agents could interfere with each other. One approach for coordination in such multi-agent systems is instituting a social law, which restricts possible behaviors of the agents. A social law which ensures that each agent is able to achieve its goal, regardless of what the other agents do, is called robust. Recent work has shown how to verify that a given social law, encoded in a MA-STRIPS formalism is robust, by compilation to classical planning. That work also introduced the notion of waitfor preconditions, which tell the agent to check if these preconditions hold before executing its next action. In this paper, we show that the waitfor mechanism strictly adds power to social laws. Specifically, we first show that, without any waitfor preconditions, a social law is robust iff there is no possible harmful interaction between the agents, and thus the MA-STRIPS problem can be decomposed into independent individual problems for each agent. Second, we show there exist multi-agent problems which are not decomposable, and therefore have no robust social law without waitfor preconditions, which do have a robust social law using waitfor. Finally, we prove that marking preconditions as waitfor preconditions never hurts the robustness of a multi-agent problem.

Introduction

Agents operating in a shared environment can interfere with each other. To ensure agents can plan without reasoning about others agents, and still be able reach their goals, a social law (Tenneholtz and Moses 1989) which restricts the allowed behaviors of the agents can be instituted. A social law is *robust* if it ensures each agent will achieve its goal.

Recently, social laws were studied in the context of multi-agent planning (Karpas, Shleyfman, and Tennenholtz 2017). In this setting, verifying if a given social law is robust was reduced to solving a centralized planning problem which attempts to find a counterexample. If the resulting planning problem is unsolvable, then the social law is robust. That paper introduced the notion of *waitfor* preconditions, which control the execution of agents' individual plans in the shared environment – that is, an agent will wait until the waitfor preconditions of its next action are fulfilled before trying to execute it in the environment. This simple mechanism allows for some synchronization between the agents via conditional execution, while still allowing each agent to solve a classical (non-conditional) planning task.

In this work we explore the power of the *waitfor* mechanism more closely. Our main motivation stems from the fact that for the physical agents some predicates are sensible and some are not. The agent cannot wait for the precondition it cannot sense, thus the two types of predicates should be treated differently. Thus, we introduce some basic observations on the nature of sensible propositions – the *waitfor* atoms.

First, we show that without waitfors, robustness verification is essentially reduced to checking if some notion of decomposability exists in the multi-agent planning task. Second, we show that there are multi-agent planning problems which only have a robust social law with waitfor preconditions, showing they are strictly more powerful than not using waitfors. Finally, we prove that adding *waitfors* preserves robustness, thus showing that agents should wait for any precondition they can.

Background

In this section we review some necessary background.

Multi-agent Problem Formalism

Here we primarily deal with problems represented in a modified version of the MA-STRIPS formalism (Brafman and Domshlak 2008), in which each agent has its own goal (Karpas, Shleyfman, and Tennenholtz 2017). Such a problem is given by $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$, where $\mathcal{F}$ is the *set of propositions* in the problem Π, $\{\mathcal{A}_i\}_{i=1}^n$ are the *sets of actions* available to *agents* numbered $[n] := \{1, \ldots, n\}$ respectively. Each *action* is a triplet $\langle \mathsf{pre}, \mathsf{add}, \mathsf{del} \rangle$, where all three components are subset of propositions. Given some action a, we sometimes refer to $(\mathsf{add}(a), \mathsf{del}(a))$ as $\mathsf{eff}(a)$ for brevity. The *initial state* $I \subseteq \mathcal{F}$ is a subset of propositions, where propositions in it are assumed to be true, and the rest $\mathcal{F} \setminus I$ are false. Finally, each agent $i \in [n]$ has its own *goal* $G_i \subseteq \mathcal{F}$.

The *centralized planning problem* is a STRIPS planning problem (Fikes and Nilsson 1971) denoted by $\Pi_c = \langle \mathcal{F}, \mathcal{A}_c, I, G_c \rangle$ where $\mathcal{A}_c = \bigcup_{i=1}^n \mathcal{A}_i$ and $G_c = \bigcup_{i=1}^n G_i$. The *projection* over the set of atoms $X \subseteq \mathcal{F}$ is also a STRIPS planning problem: $\Pi^X = \langle \mathcal{F}^X, \mathcal{A}^X, I^X, G^X \rangle$, where for every atom subset $E \in \{\mathcal{F}, I, G\}$ we apply the operator $E^X = E \cap X$, e.g., the set of atoms of Π^X is

$X = X \cap \mathcal{F}$, and the set of actions is defined as

$$\mathcal{A}^X = \{\langle \mathsf{pre}(a), \mathsf{del}(a) \cap X, \mathsf{add}(a) \cap X\rangle \mid \mathsf{pre}(a) \subseteq X\}.$$

The natural *individual projection* for agent i is denoted by $\Pi_i = \langle \mathcal{F}, \mathcal{A}_i, I, G_i\rangle$.

Let $\Pi = \langle \mathcal{F}, \mathcal{A}, I, G\rangle$ be a STRIPS planning task. We define a state to be $s \subseteq \mathcal{F}$. An action $a \in \mathcal{A}$ can be *applicable* in a state s if $\mathsf{pre}(a) \subseteq s$, and the *result of this application* is $s[\![a]\!] := (s \setminus \mathsf{del}(a)) \cup \mathsf{add}(a)$. The result of sequentially applying (if possible) the sequence of actions π to the state s is denoted by $s[\![\pi]\!]$. The sequence of actions $\pi = a_1..a_m$ is called a *plan* if $G \subseteq I[\![\pi]\!]$. The k-prefix of the plan π is the sequence of actions $a_1..a_k$, and denoted by $\pi|_k$, where $k \in [m]$. The zero prefix applying empty sequence of actions is annotated as $I[\![\pi|_0]\!] = I$. The set of all plans of a task Π is denoted by $\pi(\Pi)$. A plan π_i for a projection Π_i is called an *individual plan* of an agent i. The plan for Π_c is called a *joint plan*. To ease the notation below we define the following sets for each agent i, let $\psi : \mathcal{A} \to 2^{\mathcal{F}}$ be one of the following functions $\psi \in \{\mathsf{pre}, \mathsf{add}, \mathsf{del}\}$. We define $\psi(\mathcal{A}_i) = \bigcup_{a\in\mathcal{A}_i} \psi(a)$.

Execution Model

In their work, Karpas et al. (2017) introduced the concept of *waitfor*, which constitutes a basic form of synchronization that can eliminate some failures. Informally, *waitfor* is a set of action preconditions that postpones the action execution until all of them are fulfilled at the same time.

Formally, let $\{\pi_i\}_{i=1}^n$ be a set of individual agent plans that solve the projection Π_i for each agent $i \in [n]$. The set of *all interleaving executions* is denoted by $\pi^{[n]}(\Pi)$. We say that a planning task is *robust to rational* if $\emptyset \neq \pi^{[n]}(\Pi) \subseteq \pi(\Pi_c)$. On one hand this means that the success of the execution does not depend on the order in which the individual plans are combined, and on the other hand ensures that at least one solution exists. If an action a is to be executed in some state s, but is inapplicable in s, i.e., $\mathsf{pre}(a) \nsubseteq s$, we declare such a situation *failure* and terminate the execution.

Waitfor is a marking of a precondition of an action with the following semantics: during execution, if there is a unfulfilled *waitfor* precondition to the action that an agent is about to execute, the agent waits instead. Thus, for each action $a \in \mathcal{A}_c$ the set $\mathsf{pre}(a)$ is divided into $\mathsf{pre}_f(a)$ – regular preconditions, and $\mathsf{pre}_w(a)$ – the $waitfor$ preconditions that signal the scheduler to postpone the action. Alternatively, $\mathsf{pre}_w(a)$ can be described as a creating a conditional effects in the following matter: applying action $a = \langle \mathsf{pre}_w, \mathsf{pre}, \mathsf{eff}\rangle$ in a state s is equivalent to saying $\mathsf{eff}(a) = \mathsf{eff}$ if $\mathsf{pre}_w \subseteq s$, and $\mathsf{eff}(a) = \emptyset$ otherwise.

Let $\pi_{\text{rw}} = a_1..a_m$ be an interleaving execution of the plans $\{\pi_i\}_{i=1}^n$. We say that π_{rw} respects *waitfor* preconditions if at the execution failure at some $k \in [m]$ it still holds that $\mathsf{pre}_w(a_k) \subseteq I[\![\pi_{\text{rw}}|_{k-1}]\!]$. We denote the set of all *executions that respect waitfors* of some problem Π as $\pi_{\text{rw}}^{[n]}(\Pi)$. It is important to note that the actual "decision to wait" is taken only during execution, and not during planning, since in the individual projection the agent is not aware of what the state of the world will be at any given point in the plan's execution. In this work we dive deeper into the implications of introducing a *waitfor* precondition into the system.

Social laws

Previous work (Karpas, Shleyfman, and Tennenholtz 2017; Tuisov and Karpas 2020) defined a social law in the context of MA-STRIPS problems in various ways. Here we adopt the most general definition:

Definition 1 (Social Law)**.** *A social law l is a transformation of an MA-STRIPS task $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n\rangle$ to a modified task $\Pi^l = \langle \mathcal{F}^l, \{\mathcal{A}_i^l\}_{i=1}^n, I^l, \{G_i^l\}_{i=1}^n\rangle$ such that:*

- *$\mathcal{F}^l$ is a set of propositions. $\mathcal{F} \subseteq \mathcal{F}^l$*
- *$\{\mathcal{A}_i^l\}_{i=1}^n$ is a set of actions for each agent i. $\forall a^l \in \mathcal{A}_i^l$ $\exists! a \in \mathcal{A}_i$:*
  - $\mathsf{pre}_f(a) \subseteq (\mathsf{pre}_f(a^l) \cup \mathsf{pre}_w(a^l))$
  - $\mathsf{pre}_w(a) \subseteq \mathsf{pre}_w(a^l)$
  - $\mathsf{add}(a) \subseteq \mathsf{add}(a^l)$
  - $\mathsf{del}(a) \subseteq \mathsf{del}(a^l)$
  - $(\mathsf{add}(a^l) \setminus \mathsf{add}(a)), (\mathsf{del}(a^l) \setminus \mathsf{del}(a)) \subseteq \mathcal{F}^l \setminus \mathcal{F}$
- *$I \subseteq I^l \subseteq \mathcal{F}^l$*
- *$\{G_i^l\}_{i=1}^n \subseteq \mathcal{F}^l$. $\forall i : G_i \subseteq G_i^l$*

In plain words, a social law can remove existing actions or add to their preconditions. It may also add propositions to the problem, and effects to the actions, but the added effects can only affect added propositions. This corresponds to the idea that the environment designer cannot add new capabilities to the agents (a city designer cannot make cars fly), but can add auxiliary variables. Lastly, it can add to the goals of agents, for example to motivate releasing the lock on some resource after using it.

Robustness of a Social Law

A crucial property to look for in a social law is its robustness, which is the guarantee that agents operating under the social law are guaranteed to achieve their goals with no conflicts. We rephrase the definition that appeared in Karpas et al. (2017) according to the notation introduced earlier:

Definition 2 (Rational Robustness)**.** *A social law l for multi-agent setting $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n\rangle$ is rationally robust with respect to waitfors iff: $\emptyset \neq \pi_{rw}^{[n]}(\Pi^l) \subseteq \pi^{[n]}(\Pi_c)$.*

In a *rationally robust* task agents can execute their plans in an arbitrary order, and still be guaranteed to arrive at their goals at the end of the execution.

Moreover, *waitfor* preconditions are only accounted for during the execution, not during planning done by agents. Thus, the set of all possible individual plans (and their interleavings) remains the same with no regard to the presence of *waitfor*s. Formally:

Theorem 1. *A rationally robust MA-STRIPS task Π is also rationally robust with respect to waitfors, i.e.:*

$$\emptyset \neq \pi^{[n]}(\Pi) \subseteq \pi(\Pi_c) \implies \emptyset \neq \pi^{[n]}(\Pi) = \pi_{rw}^{[n]}(\Pi).$$

Proof. The direction $\pi_{\text{rw}}^{[n]}(\Pi) \subseteq \pi^{[n]}(\Pi)$ is true by definition. Let $\pi \in \pi^{[n]}(\Pi) \subseteq \pi(\Pi_c)$, then π is a valid plan for

Π_c. Let a_k be the k's action in the plan π, then it holds that $\mathsf{pre}(a_k) \subseteq I[\![\pi|_{k-1}]\!]$, which, implies that $\mathsf{pre}_w(a_k) \subseteq I[\![\pi|_{k-1}]\!]$. Thus, $\pi \in \pi_{\text{rw}}^{[n]}(\Pi)$. □

Robustness and Single-agent Decomposition

To understand the true meaning and applicability of waitfors, we first need to discuss what happens if we do not allow them. For the rest of this section assume no waitfor preconditions exist. In what follows we develop a useful criterion that implies a problem's robustness and does not require to invoke a centralized planner.

Decompositions

The main idea behind developing such a criterion is decomposing the multi-agent task into a set of minimally interacting single-agent tasks. If this can be done, and some additional criteria are met – the task is robust. Here we lay out the relevant definitions:

Given an MA-STRIPS problem $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$, let *decomposition* $\{\mathcal{F}_i\}_{i=0}^n$ be a partition of $\mathcal{F}$ into $n+1$ subsets. The semantics are that $\mathcal{F}_i$ is a set of propositions associated with agent i, and $\mathcal{F}_0$ are the global propositions – for example, the existence of a road usable by all agents.

Given a decomposition, define $D_i = \mathcal{F}_i \cup \mathcal{F}_0$ – the *relevant set* of propositions for agent i. This allows us to define the individual projections for each agent based on their relevant sets as $\Pi_i^{D_i} = \langle \mathcal{F}^{D_i}, \mathcal{A}^{D_i}, I^{D_i}, G^{D_i} \rangle$. We denote by Π^D the set of projections $\{\Pi_i^{D_i}\}_{i=1}^n$. The set of all interleaving plans of Π^D is denoted by $\pi^{[n]}(\Pi^D)$. We say that Π^D is rationally robust if $\pi^{[n]}(\Pi^D) \subseteq \pi(\Pi_c)$. Note that natural projections admit the decomposition $\mathcal{F}_0 = \mathcal{F}$ and $\mathcal{F}_i = \emptyset$ for each $i \in [n]$.

Since $\mathcal{A}_i^{D_i} \subseteq \mathcal{A}_i$ for each $i \in [n]$, a decomposition constitutes a social law in terms of action restrictions. Thus, from all possible decompositions we expose one that has an intrinsic connection to rational robustness:

Definition 3 (Delete decomposition)**.** *Let* $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$ *be an MA-STRIPS problem.* $\{\mathcal{F}_i\}_{i=0}^n$ *is a delete decomposition of* Π *if:*

- $\forall i \in [n] : \mathcal{F}_i = \mathsf{del}(\mathcal{A}_i)$.
- $\mathcal{F}_0 = \mathcal{F} \setminus \bigcup_{i=1}^n \mathsf{del}(\mathcal{A}_i)$.
- $\forall i, j \in [n] \cup \{0\}, i \neq j : \mathcal{F}_i \cap \mathcal{F}_j = \emptyset$

In a delete decomposition D_i includes only propositions that no one but the agent i can delete. Such a decomposition is not guaranteed to exist for every problem, and we label the task *delete-decomposable* if it does have a delete decomposition. We proceed to show that the delete-decomposability of a task, along with some technical requirements, guarantees that the problem in question is rationally robust. To formalize this claim and prove it, we require some additional definitions first:

- **Useless actions** – actions that do not appear in any individual plan. An action that is not useless is called useful.
- **Useless propositions** – propositions that do not appear as preconditions for any useful action.

Now we have the tool set to formally establish the connection between the delete-decomposability and the rational robustness using the following pair of theorems:

Theorem 2 (Decomposability condition)**.** *Consider an MA-STRIPS problem* $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$*. If* Π *has no waitfor preconditions, is rationally robust, has no useless actions and no useless propositions, then* Π *is delete-decomposable.*

Proof. We need to prove the existence of a delete decomposition $\{\mathcal{F}\}_{i=0}^n$, where each $\mathcal{F}_i = \mathsf{del}(\mathcal{A}_i)$ for each $i > 0$, i.e. it is required to show that $\{\mathcal{F}\}_{i=0}^n$ is a partition. $\mathcal{F}_0$ is always disjoint from the rest of the $\mathcal{F}_i$'s by definition, so it is sufficient to show that $\forall i, j \in [n], i \neq j : \mathcal{F}_i \cap \mathcal{F}_j = \emptyset$

Assume by contradiction that there exists some $f \in \mathcal{F}$ and a pair of agents i, j such that $f \in \mathcal{F}_i \cap \mathcal{F}_j$. Thus, there are two actions $a_i \in \mathcal{A}_i, a_j \in \mathcal{A}_j$ such that $f \in \mathsf{del}(a_i) \cap \mathsf{del}(a_j)$. Given that f is not useless there is some useful action a' such that $f \in \mathsf{pre}(a')$.

Let $a' \in \mathcal{A}_z$. Since there are no useless actions, a' belongs to some plan of agent z, denoted π_z. Let π_i and π_j be the plans of agents i and j respectively, that include the actions a_i and a_j respectively. Assume, WLOG, that $i \neq z$. There is an interleaving execution where a_i appears immediately before a', resulting in failure in the execution. This contradicts the robustness of Π. □

The second theorem is an attempt to establish the link in the opposite direction – what is needed for a delete-decomposable task to be rationally robust. Here the conditions are much stricter, since i's goals and the preconditions of i's actions should be in i's control too for the rational robustness to emerge. Formally:

Theorem 3 (Robustness condition)**.** *Consider a delete-decomposable problem with no waitfor preconditions* $\Pi = \langle \mathcal{F}, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$*, with the relevant sets* $\{D_i\}_{i=1}^n$*. If* $\forall i \in [n] : \mathsf{pre}(\mathcal{A}_i) \cup G_i \subseteq D_i$*, and for each i individual task* $\Pi_i = \langle \mathcal{F}, \mathcal{A}_i, I, G_i \rangle$ *is solvable,* Π *is robust.*

Proof. Assume by contradiction Π is not robust. Thus, $\pi^{[n]}(\Pi) \not\subseteq \pi(\Pi_c)$, i.e. execution of some interleaving of individual plans leads to failure or G_c is not achieved.

Goal achievement – for each agent i we know that π_i is a solution, which means if we apply π_i to I, we achieve G_i. Since we assumed $G_i \subseteq D_i$, no element of it can be deleted by action not in $\mathcal{A}_i$. Thus no action from $\mathcal{A}_{j \neq i}$ will have any effect on G_i, which will still be achieved after executing last action in π_i. This reasoning can be applied to any agent, thus the execution achieves G_c.

Possibility of execution – Assume by contradiction that there exists some $\pi = a_1..a_m$ – an interleaving of individual plans for Π leading to a failure. Let a_k ($k \in [m]$) be the first action that can not be applied. Let $\pi|_{k-1}$ be the $k-1$ prefix of the plan, so $s_{k-1} := s_0[\![\pi|_{k-1}]\!]$. Since a_k is not applicable, it holds that $\mathsf{pre}(a_k) \not\subseteq s_{k_1}$, i.e., there is $f \in \mathsf{pre}(a_k)$ such that $f \notin s_{k-1}$. Such f can be missing from s^{k-1} for three reasons:

1. f was not a part of any state before s_{k-1}, i.e. $f \notin I$, and was never added.

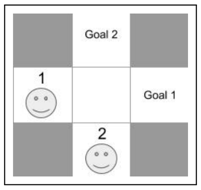

Figure 1: Crossroads Example (smileys denote agents).

2. f appeared in some state before s_{k-1}, but was last deleted by the same agent i that performs action a_k.
3. f appeared in some state before s_{k-1}, but was last deleted by some agent $j \neq i$.

We now show that all three cases are impossible:

Case 1 and Case 2: Recall that π is an interleaving of individual solutions, i.e. the relative order of actions in π_i is preserved in π. Moreover, we know that π_i is a solution, thus it is executable. This means that after executing the action immediately preceding a_k in π_i, f must be true, hence cases 1 and 2 are impossible.

Case 3: Recall that Π is assumed to be decomposable and $\mathsf{pre}(\mathcal{A}_i) \subseteq D_i$. This means $f \in D_i$. By definition, all propositions that can be deleted by j are in $\mathcal{F}_j$, and also by definition, $\mathcal{F}_j \cap D_i = \emptyset$, i.e., f can not be deleted by any $j \neq i$.

We have shown that execution of any arbitrary π is always possible and always achieves G_c, thus Π is robust. $\square$

Notice that Theorem 3 only requires verifying that *individual* problems are solvable, thus invoking no centralized planner. Checking decomposability and basic set operations on preconditions and goals all require polynomial time. This differs radically from the robustness verifying procedure given by Karpas et al. (2017), which, although complete, requires a centralized planning procedure.

The Power of Waitfor

So far, we limited our discussion to the setting where there are no *waitfor* preconditions for actions. Introducing *waitfors* changes the picture. For example the two theorems proven above are no longer valid. However, in what follows we uncover some deeper connections between robustness and *waitfor*s, as well as their usability and limitations.

Usefulness of waitfors We first present a short and informal example to convey the usefulness of waitfors. Consider a rectangular grid where agents can move to one of 4 adjacent tiles, given this tile is not occupied by another agent. In a layout shown in Figure 1, assume agent 1 can only move left or right, and agent 2 can only move up or down. The problem with no waitfor preconditions is not rationally robust, while if we mark the "not occupied" precondition as waitfor, the problem becomes robust.

Consequences of introducing new waitfors As was shown above, marking preconditions as waitfor can be beneficial, i.e. turn a non-robust problem into a robust one. In what follows we claim that marking preconditions as waitfor does no harm, i.e., it cannot turn a robust problem to a non-robust one. This claim can be considered a generalization of Thm. 1. Formally:

Theorem 4. *Consider a MA-STRIPS task* $\Pi = \langle F, \{\mathcal{A}_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$. *Let* Π^l *be a social law that is identical to* Π *with respect to atoms, initial state and goals. The actions of* Π^l *are defined as:* $\forall i \in [n], a \in \mathcal{A}_i, \exists! a^l \in \mathcal{A}_i^l$ *such that*

1. $\mathsf{add}(a_i) = \mathsf{add}(a_i^l)$, $\mathsf{del}(a_i) = \mathsf{del}(a_i^l)$;
2. $\mathsf{pre}(a_i) = \mathsf{pre}(a_i^l)$, $\mathsf{pre}_w(a_i) \subseteq \mathsf{pre}_w(a_i^l)$.

Then, $\pi_{rw}^{[n]}(\Pi) \subseteq \pi(\Pi_c) \implies \pi_{rw}^{[n]}(\Pi) = \pi_{rw}^{[n]}(\Pi^l)$.

In what follows, we assume the actions $\mathcal{A}_c$ and $\mathcal{A}_c^l$ to be identical name-wise for the sake of set of plans comparison.

Lemma 1. *Let* Π *and* Π^l *to be as described in Thm. 4. Then,* $\pi_{rw}^{[n]}(\Pi^l) \subseteq \pi_{rw}^{[n]}(\Pi)$.

Proof. Let $\pi^l \in \pi_{\text{rw}}^{[n]}(\Pi^l)$. If π^l is a plan, we have that $\pi \in \pi_{\text{rw}}^{[n]}(\Pi)$, since for each $a^l \in \pi^l$ we have $\mathsf{pre}(a^l) = \mathsf{pre}(a)$. Otherwise, assume in contradiction that $\pi \notin \pi_{\text{rw}}^{[n]}(\Pi)$. Then, there is a failure at the execution of action number k alongside π, denote it a_k. The failure to execute a_k means that there is an atom $f \in \mathsf{pre}_w(a_k)$, such that $f \notin I[\![\pi|_{k-1}]\!]$ but by definition of l we have that $f \in \mathsf{pre}_w(a_k) \subseteq \mathsf{pre}_w(a_k^l)$. This contradicts that $\pi^l \in \pi_{\text{rw}}^{[n]}(\Pi^l)$. $\square$

Now we give the proof of Thm. 4.

Proof. The direction $\pi_{\text{rw}}^{[n]}(\Pi^l) \subseteq \pi_{\text{rw}}^{[n]}(\Pi)$ is due to Lem. 1. We shall prove inclusion in other direction. Let $\pi^l \in \pi_{\text{rw}}^{[n]}(\Pi) \subseteq \pi(\Pi_c)$. Since π is a plan for Π_c, there are no conflicts in π. Thus, $\pi_{\text{rw}}^{[n]}(\Pi) \subseteq \pi_{\text{rw}}^{[n]}(\Pi^l)$. $\square$

Lastly, we establish equivalence between the set of joint plans for Π and Π^l:

Lemma 2. $\pi(\Pi_c) = \pi(\Pi_c^l)$

Proof. Note that Π_c and Π_c^l being a classical planning problems are agnostic to waitfors. And without this distinction, $\Pi_c^l = \Pi_c$ by definition. $\square$

An immediate corollary of Thm. 4 and Lem. 2 is that declaring any precondition of any action as *waitfor* will not invalidate the robustness of the task.

Corollary 1. $\pi_{rw}^{[n]}(\Pi) \subseteq \pi(\Pi_c) \implies \pi_{rw}^{[n]}(\Pi^l) \subseteq \pi(\Pi_c^l)$.

This means there is nothing to gain from a robustness perspective from leaving some precondition non-*waitfor*. This observation may be useful for developing social law synthesis tools, as the search space for a robust social law may be restricted to all-*waitfor* tasks only.

Conclusion

We have discussed the applicability and usefulness of *waitfor* preconditions in social laws. We have established a strong connection between *delete-decomposability* of a task and its robustness in the absence of *waitfor* preconditions. We have also shown that marking precondition as *waitfor* cannot destroy robustness, but might achieve it – and is thus always beneficial. Thus, the main limitation is access to sensors which enable implementing *waitfor* preconditions on the agent.

References

Brafman, R. I.; and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS 2008 - Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 28–35. ISBN 9781577353867. URL www.aaai.org.

Fikes, R.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artif. Intell.* 2(3/4): 189–208. doi:10.1016/0004-3702(71)90010-5. URL http://dx.doi.org/10.1016/0004-3702(71)90010-5.

Karpas, E.; Shleyfman, A.; and Tennenholtz, M. 2017. Automated verification of social law robustness in strips. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 163–171. ISBN 9781577357896. ISSN 23340843.

Tenneholtz, N.; and Moses, Y. 1989. On Cooperation in a Multi-Entity Model. In *International Joint Conference on Artificial Intelligence*, 918–936. URL https://www.ijcai.org/Proceedings/89-2/Papers/011.pdf.

Tuisov, A.; and Karpas, E. 2020. Automated verification of social law robustness for reactive agents. *ECAI* 325: 2386–2393. ISSN 09226389. doi:10.3233/FAIA200369.