Concurrent Multi-Agent Systems with Temporal Logic Objectives: Game Theoretic Analysis and Planning through Negotiation

Jie Fu * Herbert G. Tanner, † Jeffrey Heinz ‡

May 30, 2014

Abstract

The paper examines equilibrium behavior and negotiation protocol design for a class of multi-agent systems composed of multiple, non-cooperative, agents. The agents modeled as finite-state transition systems, are autonomous, and are interacting *concurrently* aiming at achieving individual tasks expressed as temporal logic formulae. Each agent has its own preferences over outcomes of its interaction with others. The agents' goals and preferences are neither perfectly aligned nor opposing. We reason about agent behaviors in such a system, by formulating a concurrent, multi-agent game with infinitely many stages. To enable the synthesis of strategies, we develop a negotiation protocol which ensures that under a proper design of preferences and tasks, the mutually accepted plan is a Pareto optimal Nash equilibrium.

Keywords: Multi-agent systems, temporal logic, game theory, negotiation protocols.

1 Introduction

We analyze the concurrent interaction of multiple heterogeneous dynamical systems, each trying to serve its own objective. These dynamical systems are modeled as finite-state transition systems, their objectives expressed as temporal logic formulae, and their interaction is encoded in a concurrent game. The questions we address is how to identify stable (in a Nash equilibrium sense) interaction behaviors, and then how to coordinate on a particular behavior, given the agents' individual objectives and preferences.

*J. Fu is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA. 19104 USA. e-mail: jief@seas.upenn.edu.

†H. Tanner is with the Department of Mechanical Engineering, University of Delaware, Newark, DE. 19716 USA. e-mail: btanner@udel.edu.

‡J. Heinz is with the Department of Linguistics and Cognitive Science, University of Delaware, Newark, DE. 19716 USA. e-mail: heinz@udel.edu.

Agent objectives are encoded in Linear Temporal Logic (LTL). Due to its expressiveness, LTL is widely used to specify many desired system properties such as safety, liveness, persistance, etc [5]. So far, (supervisory) control problems with temporal logic specifications, have been approached using primarily top-down, centralized synthesis methods; there is a global objective, and agents cooperate to achieve it. The solution usually involves a task decomposition: break the global task into subtasks, such that completion of these subtasks ensures the global one. Such a compositional architecture is found in concurrency theory [12], and decentralized supervisory control [4, 22]. When the global task is in the form of an LTL formula, methods have been developed [8] to break up the global specification into a set of control and communication policies for each agent to follow. Centralized controllers can also be synthesized in the face of modeling uncertainty [21]. Instances where agents have their own temporal logic specifications and treat each other as part of some reactive uncontrollable environment, have also been looked at [11].

In these approaches, the correctness of the entire system's behavior is determined by the correctness of the local subsystem controllers. However, if one of the local controllers fails, the performance and safety of the entire system is compromised. This fragility motivates us to consider coordination within a game theoretic framework. With available discrete abstractions from continuous or hybrid dynamical systems [5, 18, 19], algorithmic game theory [2] provides at the abstraction level a natural framework for the composition of systems. The notion of *rational synthesis* [10] poses the control problem for this class of multi-agent systems inside a non-zero-sum game theoretic framework (cf. [11]): each system is a player in a game, and the synchronous or asynchronous evolution of states in different systems are moves made by one or more of these players.

This paper exploits and extends recent game theoretic results [7, 10] for negotiation-based behavior planning in multi-agent systems with temporal logic control objectives. Existing results do not fit particularly well in the case considered here. First, with agents having independent objectives, either as a Boolean utility value [10] or a set of ranked objectives [7], implicit cooperation between agents is not encouraged. Furthermore, is that it is not clear how a single equilibrium is agreed upon without some type of negotiation and consensus building.

Consider the following example: three agents indexed 1, 2, and 3 are roaming in the four rooms A, B, C, and D, shown in Fig. 1. Every agent is assigned with a surveillance task that requires it to visit a sequence of rooms infinitely often. Their concurrent motion brings the possibility of interference: they can get into each other's way because only one agent can fit through a specific door. We see in Section 5 that if everyone just works for himself, there exists a stable outcome in which all agents accomplish their task. However, if agents 1 and 2 see agent 3 as a common adversary, depending on

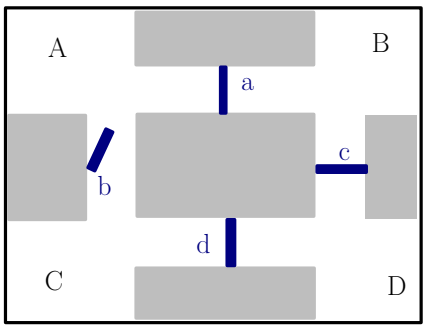

Figure 1: A partitioned rectangular environment in which three agents roam. Agents visit different rooms, indexed A, B, C, D, by passing through doors a, b, c and d, but only one at a time can go through a given door.

how their preferences over game outcomes are defined, they can implicitly cooperate to prevent agent 3 from succeeding. We show that in cases when agents are selfish, the right assignment of preferences and the implementation of a negotiation protocol allows them to reach an agreement on an equilibrium policy, with which each agent achieves its goal.

To do this, we use an incentive-centered design with a new definition of agents' utilities, that allows implicit cooperation to emerge as an equilibrum. We decide which interaction outcomes are stable (in a Nash sense) in this non-cooperative, concurrent game the agents are engaged in. Then we consider *cooperative games with temporal logic objectives* in which agents can form coalitions. A decision procedure for cooperative equilibiria is provided. To coordinate, agents communicate and negotiate a mutually accepted plan. Inspired by [13], we design a negotiation protocol which ensures that under a proper design of preferences and tasks for agents the mutually agreed plan is a Pareto optimal pure Nash equilibrium.

2 Preliminaries

2.1 Automata and Semiautomata

Let Σ be a fixed, finite alphabet. We denote Σ^* and Σ^ω the sets of *finite* and *infinite* sequences or words, respectively, over Σ. Elements in a sequence w are indexed $w^{(i)}$, where the index i runs from 0 to the *length of word*, denoted $|w|$, less one. The *empty word* is denoted λ and $|\lambda| = 0$. A word of infinite length is called an *ω-word*. A word v is a *prefix* of a word w if there exist $x \in \Sigma^*$ or $x \in \Sigma^\omega$ such that $w = vx$. For an integer $k \leq |w|$, $\mathsf{Pr}^{=k}(w)$ is the prefix of w of length k. Given a word w, we write $\mathsf{Occ}(w)$ for the set of symbols occurring in w and $\mathsf{Inf}(w)$ for the set of symbols occurring infinitely often in w. If w is a finite word, then $\mathsf{last}(w)$ denotes its last symbol, that is, the one for which $\mathsf{last}(w) = w^{(|w|-1)}$.

A *deterministic* semiautomaton (SA) is a triple $A = (Q, \Sigma, T)$ where Q is a finite set of states, Σ

is a finite alphabet, and $T : Q \times \Sigma \to Q$ is the *transition function* which can be expanded recursively, i.e. $T(q, \sigma w) = T(T(q, \sigma), w)$, $\sigma \in \Sigma$, and $w \in \Sigma^*$. We write $T(q, \sigma) \downarrow$ to specify that a transition labeled σ is defined at q. A *run* in A on a word $w = w^{(0)}w^{(1)} \ldots \in \Sigma^*$ (or Σ^ω), is a sequence of states $\rho = \rho^{(0)}\rho^{(1)}\rho^{(2)} \ldots \in Q^*$ (or Q^ω) such that for each $0 \leq i \leq |\rho| - 1$, $\rho^{(i+1)} \in T(\rho^{(i)}, w^{(i)})$. In this case we say that ρ is *generated by* w. The transition function in A can be made *total* by adding a state called sink, such that for all states $q \in Q$ for which there exists a symbol $\sigma \in \Sigma$ that cannot trigger a transition from q, we define $T(q, \sigma) = \mathsf{sink}$, and let $T(\mathsf{sink}, \sigma) = \mathsf{sink}$, for all $\sigma \in \Sigma$. A deterministic Büchi automaton (DBA) is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, F)$ where (Q, Σ, T) is a deterministic SA, I is the *initial state*, and $F \subseteq Q$ is the *acceptance component*, and $\mathcal{A}$ accepts a word $w \in \Sigma^\omega$ iff the run ρ on w satisfies $\rho^{(0)} = I$ and $\mathsf{Inf}(\rho) \cap F \neq \varnothing$. The set of words accepted by the automaton $\mathcal{A}$ constitutes its *language* and is denoted $L(\mathcal{A})$. A DBA is *total* if its transition function is total.

2.2 Games and strategies

A deterministic *two-player turn-based zero-sum game* is a tuple $\mathcal{H} = (V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, T, q^{(0)}, \mathsf{WIN})$, where for $i = 1$ or 2, V_i is the set of states where player i makes a move and Σ_i is the set of available actions for player i. We assume $V_1 \cap V_2 = \varnothing = \Sigma_1 \cap \Sigma_2$ and let $V = V_1 \cup V_2$. The transition function in the game is $T : V_i \times \Sigma_i \to V_j$, with $q^{(0)}$ the initial state, and WIN the *winning condition*. A run in the game is an infinite sequence of states $\rho \in V^\omega$. For Büchi winning conditions, a run ρ is winning for player 1 if and only if $\mathsf{Inf}\,(\rho) \cap F \neq \varnothing$.

A strategy for player i is a function $\mathsf{S}_i : V^* V_i \to \Sigma_i$ such that for every wv with $w \in V^*$ and $v \in V_i$, if $\mathsf{S}_i(wv) = \sigma \in \Sigma_i$, then $T(v, \sigma) \downarrow$. Player i *follows* a strategy S_i if for any $\rho \in V^*$, player i takes action $\mathsf{S}_i(\rho)$. Player i has a *winning strategy* at state $v \in V_1 \cup V_2$, and we denote it WS_i, if the game that starts at v with player i following WS_i, results in victory. The set of states from which player i has a winning strategy is called *the winning region* for this player and is denoted Win_i. For Büchi games, one of the players has a winning strategy [14]. Details on how to compute winning strategies for player 1 in a two-player turn-based Büchi game are found in [20].

2.3 A specification language

We consider to use a fragment of LTL [1] to specify a set of desired system properties such as safety, liveness, persistence and stability. A formula in this LTL fragment is built from True, False, a finite set of atomic propositions $\mathcal{AP}$, and the Boolean and temporal connectives $\wedge, \vee, \neg$ and $\Box$ (always), $\Diamond$ (eventually). An LTL fragment formula φ can always be represented by a DBA with alphabet $2^{\mathcal{AP}}$. For semantics of temporal logic formulae, see [1, 9].

3 Büchi games

In the concurrent games considered, each agent is tasked with satisfying a temporal logic objective equivalently expressed as the language accepted by a DBA. Once we define player preference orderings over all possible outcomes of the game, we ask whether there are pure Nash equilibria in this game, and develop a decision process for these equilibria. We show that by allowing the agents to form coalitions, the emerging behavior is captured by another solution concept, called cooperative equilibrium. Decision procedures for cooperative equilibria are also provided.

3.1 The formulation of the multi-agent game

When agents interact, the actions of one have conditional effects over the others. We refer to the conditional effects of agents' interaction as *the world*, which is a formal system over a set of atomic propositions $\mathcal{AP}$ [16]. Atomic propositions and their negations are *literals*, combined in conjunction to form *sentences*. All sentences formed this way produce the set of *world states* $\mathcal{C}$. On this set, a formal model is created to capture all concurrent interactions agents. This model is a *semiautomaton* augmented with a *labeling function* that maps every state to a sentence which is true at that state. Formulae in an LTL fragment [1, 15] specify desired system properties such as reachability, safety and liveness, recurrence, etc. Each formula in this class is equivalently expressed as a DBA.

Let $\Pi = \{1, \ldots, N\}$ be an index set and 2^{Π} denote its power set, i.e., the set of all subsets of Π. Given a tuple $\boldsymbol{s} = (s_1, \ldots, s_N)$ denote $\boldsymbol{s}[i] = s_i$ the i-th entry of $\boldsymbol{s}$. An agent is modeled as a labeled semiautomaton (a variant of Kripke structure) $A_i = (Q_i, \Sigma_i, T_i, q_i^{(0)}, \mathsf{LB}_i)$, where $q_i^{(0)}$ is the initial state and $\mathsf{LB}_i : Q_i \to \mathcal{C}$ is the labeling function. The conditional effect of action $\sigma \in \Sigma_i$ is captured by its pre- and post-conditions: the pre-condition of σ, denoted $\textsc{Pre}(\sigma)$, is a sentence in $\mathcal{C}$ that has to be true in order for σ to be executed; the post-condition of σ, denoted $\textsc{Post}(\sigma)$, is a sentence in $\mathcal{C}$ that must be true once the action is completed. Whenever $T_i(q, \sigma) \downarrow$, $\mathsf{LB}_i(q) \implies \textsc{Pre}(\sigma)$; similarly, if there is a transition from q to q' on action σ, expressed as $q \xrightarrow{\sigma} q'$, it holds that $\mathsf{LB}_i(q') \implies \textsc{Post}(\sigma)$.

The interaction between agents is captured as follows.

Definition 1. *For a set* Π *of agents, each of which is modeled as* $A_i = (Q_i, \Sigma_i, T_i, q_i^{(0)}, \mathsf{LB}_i)$*, for* $i \in \Pi$*, their* concurrent product *is a tuple* $P = A_1 \circ A_2 \circ \ldots \circ A_N = \big(Q, \mathcal{ACT}, T, q^{(0)}, \mathsf{LB}\big)$ *where*

Q $\subseteq Q_1 \times Q_2 \times \ldots Q_N$ *is the set of states.*

$\mathcal{ACT}$ $= \Sigma_1 \times \ldots \times \Sigma_N$ *is the alphabet. Each* $\boldsymbol{a} = (a_1, a_2, \ldots, a_N) \in \mathcal{ACT}$ *is an action profile, encoding the actions played by all agents simultaneously.*

T $: Q \times \mathcal{ACT} \to Q$ *is the transition function: given* $q = (q_1, \ldots, q_N)$ *and* $\boldsymbol{a} = (a_1, \ldots, a_N) \in \mathcal{ACT}$*, we have*

$$T(q, \boldsymbol{a}) = T\big((q_1, \ldots, q_N), (a_1, \ldots, a_N)\big) = (q'_1, \ldots, q'_N)$$

provided that $\forall i \in \Pi$*, (i)* $q'_i = T_i(q_i, a_i)$*, and (ii)* $\wedge_{i\in\Pi}\mathsf{LB}_i(q_i) \implies \textsc{Pre}(a_i)$.

$q^{(0)}$ $= (q_1^{(0)}, \ldots, q_N^{(0)}) \in Q$ *is the initial state of the product.*

LB $: Q \to 2^{\mathcal{AP}}$ *is the labeling function. Given* $q = (q_1, \ldots, q_N) \in Q$*,* $\mathsf{LB}(q) = \{p \in \mathcal{AP} \mid \wedge_{i\in\Pi}\mathsf{LB}_i(q_i) \implies p\}$ *is a set of atomic propositions evaluated* True *at state* q.

The concurrent product in Definition 1 describes the game *arena*, and expresses all possible interactions between agents. The actual concurrent game is played on this arena, with agents' objectives determining the game equilibria—see Remark 1. The objective of agent i is specified with an LTL fragment formula φ_i and can be expressed as an ω-regular language Ω_i over $2^{\mathcal{AP}}$, which is accepted by a *total* DBA $\mathcal{A}_i = \big(S_i, 2^{\mathcal{AP}}, T_i, I_i, F_i\big)$ with $\mathsf{sink} \in S_i$.

Let $\mathsf{Mov} : Q \times \Pi \to 2^{\Sigma}$, where $\Sigma = \bigcup_{i\in\Pi} \Sigma_i$, be a set-valued map which for state $q \in Q$ and agent $i \in \Pi$ outputs a set of actions available to agent i at q (cf. [6]). We write $\mathsf{Mov}(q, i) = \{\boldsymbol{a}[i] \in \Sigma_i \mid T(q, \boldsymbol{a}) \downarrow\}$, where T is the transition function in P.

A *play* $\mathsf{p} = q^{(0)}\boldsymbol{a}^{(0)}q^{(1)}\boldsymbol{a}^{(1)}q^{(2)}\boldsymbol{a}^{(2)}\ldots$ becomes an interleaving sequence of states and action profiles, such that for all $i \geq 0$, we have $T\big(q^{(i)}, \boldsymbol{a}^{(i)}\big) = q^{(i+1)}$. A *run* $\rho = q^{(0)}q^{(1)}\ldots$ is the projection of play p onto Q. A *deterministic strategy* for agent i is a map $\mathsf{S}_i : Q^* \to \Sigma_i$ such that $\forall\, \rho = q^{(1)}q^{(2)}\ldots \in Q^*$, $\mathsf{S}_i(\mathsf{Pr}^{=k}(\rho)) \in \mathsf{Mov}\big(q^{(k-1)}, i\big)$, for $1 \leq k$. A *deterministic strategy profile* $\boldsymbol{S} = (\mathsf{S}_1, \ldots, \mathsf{S}_N)$ is a tuple of strategies, with S_i being the strategy of agent i. The set of all strategy profiles is denoted $\mathcal{SP}$. In this paper, we consider only deterministic strategies. A run ρ is *compatible* with a strategy profile $\boldsymbol{S} = (\mathsf{S}_1, \ldots, \mathsf{S}_N)$ if it is produced when every agent i adheres to strategy S_i. All runs compatible with strategy profile $\boldsymbol{S}$ form the set of game *outcomes* for this strategy profile, denoted $\mathsf{Out}(q^{(0)}, \boldsymbol{S})$.

3.2 Preferences and equilibria

Assume that each agent obtains a Boolean payoff 1 if its objective is accomplished, and 0 otherwise. The payoff of agent i is given by a function $\mathrm{u}_i : Q \times \mathcal{SP} \to \{0, 1\}$ from the set of states Q and *strategy profiles* $\mathcal{SP}$ defined as $\mathrm{u}_i(q, \boldsymbol{S}) = 1$ if for all *runs* ρ in $\mathsf{Out}(q, \boldsymbol{S})$, we have $\mathsf{LB}(\rho) \in \Omega_i$. The payoff vector is the tuple made of the payoffs of all agents: $\mathbf{u}(q, \boldsymbol{S}) = \big(\mathrm{u}_1(q, \boldsymbol{S}), \ldots, \mathrm{u}_N(q, \boldsymbol{S})\big)$; we say that

strategy profile $\boldsymbol{S}$ yields the payoff vector $\mathbf{u}(q, \boldsymbol{S})$. The set of all possible payoff vectors is denoted $\mathcal{PV} = \bigcup_{\boldsymbol{S} \in \mathcal{SP}} \mathbf{u}(q^{(0)}, \boldsymbol{S})$.

A preference ordering for agent i is a partial order $\lesssim_i$ over $\mathcal{PV}$: for $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathcal{PV}$, if $\boldsymbol{u}_1 \lesssim_i \boldsymbol{u}_2$, then agent i either prefers a strategy profile $\boldsymbol{S}_2$ with which $\mathbf{u}\big(q^{(0)}, \boldsymbol{S}_2\big) = \boldsymbol{u}_2$, over a strategy profile $\boldsymbol{S}_1$ with which $\mathbf{u}\big(q^{(0)}, \boldsymbol{S}_1\big) = \boldsymbol{u}_1$, or is at least indifferent between $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. In the latter case we write $\boldsymbol{u}_1 \simeq_i \boldsymbol{u}_2$.

Definition 2 (cf. [10])**.** *A deterministic strategy profile $\boldsymbol{S}$ is a* pure Nash equilibrium *in a multi-agent non-cooperative concurrent game if any other strategy profile $\boldsymbol{S}'$ obtained by agent $i \in \Pi$* unilaterally *deviating from one action profile given by $\boldsymbol{S}$, results in $\mathbf{u}\big(q^{(0)}, \boldsymbol{S}'\big) \lesssim_i \mathbf{u}\big(q^{(0)}, \boldsymbol{S}\big)$.*

This paper considers only pure Nash equilibria, henceforth referred to as just equilibria.

3.3 Finding Equilibria

An equilibrium is related to the notion of best response; however, the notion itself does not directly specify some meaningful game outcome [17]. We pose the following question:

Problem 1. *For a payoff vector $\boldsymbol{u} \in \mathcal{PV} \subseteq \{0,1\}^N$, is there an equilibrium $\boldsymbol{S}$ such that $\mathbf{u}\big(q^{(0)}, \boldsymbol{S}\big) = \boldsymbol{u}$?*

Our answer is in line with existing methods [7, 10], but also differs. In the literature, each agent either has a single temporal logic specification [10], or a whole set of objectives ranked according to its own preference relation [7]. In either case, as long as an agent meets its specification it does not care what others are doing. Our definition of preference orderings among agents makes a difference in computing equilibria.

Definition 3 (cf. [3])**.** *Consider a set of semiautomata with designated initial states $A_i = (Q_i, \Sigma_i, T_i, q_i^{(0)})$, for $1 \leq i \leq n$. Their* synchronized product *is a tuple*

$$A_1 \ltimes A_2 \ltimes \ldots \ltimes A_n = \Big(\prod_{i=1}^{n} Q_i, \bigcup_{i=1}^{n} \Sigma_i, T, \big(q_1^{(0)}, \ldots, q_n^{(0)}\big) \Big)$$

where the transition relation T is defined as $T(q, \sigma) \triangleq (q'_1, \ldots, q'_n)$, for $q = (q_1, q_2, \ldots, q_n)$, where $q'_i = T_i(q_i, \sigma)$ if $T_i(q_i, \sigma) \downarrow$, and $q'_i = q_i$ otherwise.

Suspect players [7] are those who can potentially be held responsible for triggering a transition at state q which is unexpected in the sense that the players were to execute action profile $\boldsymbol{b}$ that would have brought them to state $T(q, \boldsymbol{b}) = q''$, but instead the game landed at state $q' \neq q''$. One of the

players, say $k \in \Pi$, unilaterally deviated from profile $\boldsymbol{b}$ and played σ instead of $\boldsymbol{b}[k]$, resulting in $T(q, \boldsymbol{a}) = q''$ where $\boldsymbol{a} = (b_1, \ldots, b_{k-1}, \sigma, b_{k+1}, \ldots, b_N)$. This new action profile $\boldsymbol{a}$ is denoted $\boldsymbol{b}[k \mapsto \sigma]$ to emphasize that it is produced from $\boldsymbol{b}$ by swapping $\boldsymbol{b}[k]$ with σ. For action profile $\boldsymbol{b}$, the suspect players triggering a transition from q to q' is

$$\mathsf{Susp}((q, q'), \boldsymbol{b}) = \{k \in \Pi \mid \exists\, \sigma \in \mathsf{Mov}(q, k), \boldsymbol{b}[k \mapsto \sigma] = \boldsymbol{a} \wedge T(q, \boldsymbol{a}) = q'\} \ .$$

To solve Problem 1, the concurrent arena $P = \big(Q, \mathcal{ACT}, T, q^{(0)}, \mathsf{LB}\big)$ is transformed to the arena of a two-player turn-based game with two fictitious players: player I and player II [7]. This arena is a semiautomaton $H = (V, \mathcal{ACT} \cup Q, T_h, v^{(0)})$, with components defined as follows:

- V $= V_I \cup V_{II}$ is the state space, with $V_I \subseteq Q \times 2^{\Pi}$, and $V_{II} \subseteq Q \times 2^{\Pi} \times \mathcal{ACT}$.
- $\mathcal{ACT} \cup Q$ is the alphabet, in which $\mathcal{ACT} = \Sigma_1 \times \cdots \times \Sigma_N$ are the moves for player I, and Q are the moves for player II.
- T_h is the transition relation defined as: given $v \in V$, either
  - (i) $v = (q, X) \in V_I$ and if for any $\boldsymbol{a} \in \mathcal{ACT}$ it is $T(q, \boldsymbol{a}) \downarrow$, then $T_h((q, X), \boldsymbol{a}) := (q, X, \boldsymbol{a}) \in V_{II}$; or
  - (ii) $v = (q, X, \boldsymbol{a}) \in V_{II}$ and if for any $q' \in Q$ it is $X' = X \cap \mathsf{Susp}\,((q, q'), \boldsymbol{a}) \neq \varnothing$, then $T_h(v, q') := (q', X') \in V_I$.
- $v^{(0)}$ $= (q^{(0)}, \Pi)$ is the initial state.

In this two-player game the players alternate: at each turn, one picks a state in the original concurrent game, and the other picks an action profile. The degree to which the objective of a particular player $i \in \Pi$ is satisfied in this process is being tracked by the objective automaton $\mathcal{A}_i$. We write $(\mathcal{A}_i, s_i^{(0)})$ to emphasize that the automaton $\mathcal{A}_i$ has initial state $s_i^{(0)} = T_i\big(I_i, \mathsf{LB}(q^{(0)})\big)$. The objectives $\{\Omega_i\}_\Pi$ are incorporated into the description of the two-player arena H using the synchronized product, to complete the associated two-player game:

$$\mathcal{H} = H \ltimes (\mathcal{A}_1, s_1^0) \ltimes \cdots \ltimes (\mathcal{A}_N, s_N^0) = \big(\hat{V}, \mathcal{ACT} \cup Q, \hat{T}, \hat{v}^{(0)}\big) \ , \tag{1}$$

with components are described as

$\hat{V}$ $= \hat{V}_I \cup \hat{V}_{II}$ where $\hat{V}_I = V_I \times S_1 \times \cdots \times S_N$, with S_i the states of $\mathcal{A}_i$, are the states where player I takes a transition. $\hat{V}_{II} = V_{II} \times S_1 \times \cdots \times S_N$ are the states where player II moves.

$\mathcal{ACT}, Q$ are the sets of actions for player I and II respectively.

$\hat{T}$ is the transition relation which for $\hat{v} = (v, s_1, \ldots, s_N)$,

- if $v \in V_I$ and $\sigma \in \mathcal{ACT}$, then $\hat{T}(\hat{v}, \sigma) := (v', s_1, \ldots, s_N)$ provided that $v' = T_h(v, \sigma)$;
- if, on the other hand, $v \in V_{II}$ and $\sigma \in Q$, then $\hat{T}(\hat{v}, \sigma) := (v', s'_1, \ldots, s'_N)$, provided that $v' = T_h(v, \sigma)$ and for each $i \in \Pi$ it is $s'_i = T_i(s_i, \mathsf{LB}(\sigma))$.

$\hat{v}^{(0)}$ is the initial state—player I moves first—which is a tuple $(v^{(0)}, s_1^{(0)}, \ldots, s_N^{(0)})$ where for each $i \in \Pi$, $s_i^{(0)} = T_i(I_i, \mathsf{LB}(\pi_1(v^{(0)}))) = T_i(I_i, \mathsf{LB}(q^{(0)}))$, with the understanding that the projection operator π_1 singles out the first component in $v^{(0)}$ and gives $\pi_1(v^{(0)}) = q^{(0)}$.

Remark 1. *The product in* (1) *is what we would use to construct the concurrent multi-agent game from its arena P and the agent objectives. For the two-player game $\mathcal{H}$, however, the agent objectives do not encode its winning condition; it is just a record keeping component to track how far each agent has gone to fulfilling its goal. The winning condition for $\mathcal{H}$ is included in* (2)*, which follows shortly.*

For a state $\hat{v} = (v, \boldsymbol{s})$ in the two-player game, v can be either in $\hat{V}_I$ or in $\hat{V}_{II}$. In the first case $(v, \boldsymbol{s}) = \big((q, X), \boldsymbol{s}\big)$ with $q \in Q$ and $X \in 2^{\Pi}$; in the second, $(v, \boldsymbol{s}) = \big((q, X, \boldsymbol{a}), \boldsymbol{s}\big)$ with $\boldsymbol{a} \in \mathcal{ACT}$. Define $\mathsf{Agt} := \pi_2 \circ \pi_1$ that maps a state $\hat{v}$ to the set of agents in $\hat{v}$, and $\mathsf{State} := \pi_1 \circ \pi_1$ that maps a state $\hat{v}$ to the state in Q in $\hat{v}$. Given $\rho = \hat{v}^{(0)}\hat{v}^{(1)} \ldots \in \hat{V}^\omega$, let $\mathsf{Agt}(\rho) = \mathsf{Agt}(\hat{v}^{(0)})\mathsf{Agt}(\hat{v}^{(1)}) \ldots$ and $\mathsf{State}(\rho) = \mathsf{State}(\hat{v}^{(0)})\mathsf{State}(\hat{v}^{(1)}) \ldots$. Player II *follows* player I on run $\rho = \hat{v}^{(0)}\hat{v}^{(1)} \ldots \in \hat{V}^\omega$ if for all $i \geq 0$, $\mathsf{Agt}(\hat{v}^{(i)}) = \mathsf{Agt}(\hat{v}^{(i+1)}) = \Pi$. State $\hat{v} = (v, s_1, s_2 \ldots, s_N)$ can be associated to a binary vector through a valuation function $\mathsf{Val} : \hat{V} \to \{0, 1\}^N$; $i = 2, \ldots, N+1$, $\mathsf{Val}(\hat{v})[i] = 1$ if $\hat{v}[i+1] \in F_i$, and $\mathsf{Val}(\hat{v})[i] = 0$ otherwise.

We can determine whether there exists an equilibrium associated with a given pay-off vector $\boldsymbol{u}$ in a multi-agent concurrent Büchi game, by solving a two-player turn-based Büchi game induced by $\boldsymbol{u}$.

Proposition 1. *Consider a concurrent game played on arena P by players Π with objectives $\{\mathcal{A}_i \mid i \in \Pi\}$. If in the two-player turn-based Büchi game $\mathcal{H}(\boldsymbol{u})$ of the form*

$$\mathcal{H}(\boldsymbol{u}) = \Big(\hat{V}, \mathcal{ACT} \cup Q, \hat{T}, \hat{v}^{(0)}, F(\boldsymbol{u})\Big) \tag{2}$$

where $\boldsymbol{u} \in \mathcal{PV}$, $F(\boldsymbol{u}) = \{\hat{v} \in \hat{V} \mid \forall i \in \mathsf{Agt}(\hat{v}), \mathsf{Val}(\hat{v}) \lesssim_i \boldsymbol{u}\}$[1] *the following conditions are satisfied*

1. *player* I *wins;*
2. *there exists a run* $\rho \in \hat{V}^\omega$ *with* $\rho^{(0)} = \hat{v}^{(0)}$ *in which* $\forall i \geq 0$ *it is* $\rho^{(i)} \in \mathsf{Win}_I$, $\mathsf{Agt}(\rho^{(i)}) = \Pi$, $\mathsf{Inf}(\rho) \cap \{\hat{v} \in F(\boldsymbol{u}) \mid \mathsf{Val}(\hat{v}) = \boldsymbol{u}\} \neq \varnothing$ *and* $\mathsf{Inf}(\rho) \cap (\hat{V} \setminus F(\boldsymbol{u})) = \varnothing$,

then there exists a pure Nash equilibrium $\boldsymbol{S}$ *in the concurrent game, such that* $\mathbf{u}(q^{(0)}, \boldsymbol{S}) = \boldsymbol{u}$.

Proof. By condition *1*, states visited infinitely often are in $F(\boldsymbol{u})$; no unilateral deviation made by an agent $j \in \Pi$ produces a payoff vector better than $\boldsymbol{u}$. Condition *2* ensures the existence of an equilibrium $\boldsymbol{S}$ associated with $\boldsymbol{u}$: if w is an ω-word that generates a run ρ in $\mathcal{H}(\boldsymbol{u})$ which satisfies 1) $\rho^{(0)} = \hat{v}^{(0)}$ and $\forall k \geq 0$, $\rho^{(k)} \in \mathsf{Win}_I$, $\mathsf{Agt}(\rho^{(k)}) = \Pi$ (all rational agents adhere to this policy); 2) $\mathsf{Inf}(\rho) \cap (\hat{V} \setminus F(\boldsymbol{u})) = \varnothing$ and $(\exists\, \hat{v} \in \mathsf{Inf}(\rho))\,[\mathsf{Val}(\hat{v}) = \boldsymbol{u}]$, then the equilibrium $\boldsymbol{S}$ is just the projection of w on the set of action profiles. □

The computational complexity of solving two-player turn-based Büchi games is $\mathcal{O}(n(m+n))$, where n is the number of game states and m is the number of transitions in $\mathcal{H}(\boldsymbol{u})$ [20]. Constructing the two-player turn-based game $\mathcal{H}$ is polynomial in the size of P and the the specification automata $\mathcal{A}_i$ [6].

3.4 Cooperative equilibria

So far, agents have been cooperating implicitly: i cooperates with j if the success of both i and j makes i happier than succeeding alone. In this section, cooperation is considered explicitly, characterized as a concurrent deviation from a strategy profile for the purpose of collectively achieving better outcomes.

A *team* X is a subset of Π. A unilateral team deviation by team $X \in 2^\Pi$ from an action profile $\boldsymbol{a}$ is denoted $\boldsymbol{a}[X \mapsto \boldsymbol{\sigma}] = (a'_1, a'_2, \ldots, a'_N)$, where $\boldsymbol{\sigma} = (b_j)_{j \in X}$ is the tuple of actions of agents in X, ordered by their index; we have $a'_i \equiv a_i$ if $i \notin X$ and $a'_i \equiv b_i$ if $i \in X$. The set of teams is denoted $\mathsf{Teams} \subseteq 2^\Pi$. Nothing prevents agents from switching teams or breaking up—as long as any resulting teams are still in Teams.

Definition 4. *A strategy profile* $\boldsymbol{S}$ *is a* cooperative equilibrium *in a multi-agent non-cooperative game if for any team* $X \in \mathsf{Teams}$*, and for any strategy profile* $\boldsymbol{S}'$ *obtained from* $\boldsymbol{S}$ *by an unilateral team deviation of* X*, it holds that for all* $k \in X$, $\mathbf{u}(q^{(0)}, \boldsymbol{S}') \lesssim_k \mathbf{u}(q^{(0)}, \boldsymbol{S})$.

Now if $T(q, \boldsymbol{a}) = q'$ is defined, then for an action profile $\boldsymbol{b} \in \mathcal{ACT}$ the set of suspect teams

[1] This is the winning condition for $\mathcal{H}$.

triggering a transition from q to q' is

$$\mathsf{SuspTeams}((q,q'),\boldsymbol{b}) := \left\{X \in \mathsf{Teams} \mid (\forall i \in X)\left[(\exists \sigma_i \in \mathsf{Mov}(q,i))\left[\boldsymbol{b}[X \mapsto (\sigma_i)_{i\in X}] = \boldsymbol{a} \wedge T(q,\boldsymbol{a}) = q'\right]\right]\right\}.$$

The two-player turn-based arena H with team deviation is constructed as

$$H = \left(V, \mathcal{ACT} \cup Q, T_h, v^{(0)}\right)$$

where

V	$= V_I \cup V_{II}$ is the set of states, with $V_I \subseteq Q \times 2^{\mathsf{Teams}}$ and $V_{II} \subseteq Q \times 2^{\mathsf{Teams}} \times \mathcal{ACT}$.
$\mathcal{ACT} \cup Q$	in which $\mathcal{ACT} = \Sigma_1 \times \cdots \times \Sigma_N$ represents the available moves for player I, and Q the moves for player II.
T_h	is the transition relation defined as: given $v \in V$, either
	(i) $v = (q, \mathcal{X}) \in V_I$ where $\mathcal{X} \subseteq \mathsf{Teams}$ and for any $\boldsymbol{a} \in \mathcal{ACT}$ we have $T(q,\boldsymbol{a})\downarrow$, in which case $T_h((q,\mathcal{X}),\boldsymbol{a}) := (q,\mathcal{X},\boldsymbol{a}) \in V_{II}$; or
	(ii) $v = (q,\mathcal{X},\boldsymbol{a}) \in V_{II}$ and for any $q' \in Q$ we have $\mathsf{SuspTeams}((q,q'),\boldsymbol{a}) \cap \mathcal{X} \neq \varnothing$, in which case $T_h(v,q') := (q',\mathcal{X}') \in V_I$ where $\mathcal{X}' = \{ X \in \mathsf{Teams} \mid X \subseteq Y, Y \in \mathsf{SuspTeams}((q,q'),\boldsymbol{a}) \cap \mathcal{X} \}$.
$v^{(0)}$	$= (q^{(0)}, \mathsf{Teams})$ is the initial state.

In this game, opportunistic teams of agents play against each other if the interests of the teammates align. The analysis of equilibria is performed as in Section 3.3: First we construct the two-player turn based game $\mathcal{H}$ from the game arena H and the agents' objectives, and then we determine if an equilibrium with respect to a pay-off vector $\boldsymbol{u}$ exists based on Proposition 1. The definition of the winning condition is slightly different here: given $\boldsymbol{u} \in \{0,1\}^N$, $\mathcal{F}(\boldsymbol{u}) = \{\hat{v} \in \hat{V} \mid \forall X \in \mathsf{Teams}(\hat{v}), \forall i \in X, \mathsf{Val}(\hat{v}) \lesssim_i \boldsymbol{u}\}$, where $\mathsf{Teams} = \pi_2 \circ \pi_1$ maps a state $\hat{v}$ into a set of teams within $\hat{v}$. Then cases considered in Section 3.3 become special cases of the one considered here, where $\mathsf{Teams} = \{\{i\} \mid i \in \Pi\}$.

4 Negotiations

This section introduces a negotiation protocol which ensures that the agents agree upon which Nash equilibrium to implement through negotiation.

For two strategy profiles $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, we say $\boldsymbol{S}_1$ *Pareto dominates* $\boldsymbol{S}_2$ if for all agents $i \in \Pi$, it

is $\mathbf{u}(q^{(0)}, \boldsymbol{S}_1) \gtrsim_i \mathbf{u}(q^{(0)}, \boldsymbol{S}_2)$ and for at least one agent $k \in \Pi$, we have $\mathbf{u}(q^{(0)}, \boldsymbol{S}_1) >_k \mathbf{u}(q^{(0)}, \boldsymbol{S}_2)$. A strategy profile $\boldsymbol{S}$ that is not Pareto dominated by any other strategy profile is called *Pareto optimal*.

Let $\mathcal{U} \subseteq \mathcal{PV}$ be the set of payoff vectors for all equilibria. A utility function $\mu_i : \mathcal{U} \to \mathbb{Z}$ maps payoff vectors to integer utilities, and measures for a given payoff vector $\boldsymbol{u}$, the number of payoff vectors that the agent prefers over $\boldsymbol{u}$. For instance, if $\boldsymbol{u}$ is the worst payoff vector for agent i, then $\mu_i(\boldsymbol{u}) = 0$; if the agent's utility for some payoff vector $\boldsymbol{u}'$ is, say ℓ, and $\boldsymbol{u} >_i \boldsymbol{u}'$ while no other $\boldsymbol{u}'' \neq \boldsymbol{u}$ exists with $\boldsymbol{u}'' >_i \boldsymbol{u}'$, then $\mu_i(\boldsymbol{u}) = \ell + 1$.

We thus get an ordering of equilibrium strategies depending on their associated utilities. Consider utility vectors in the form $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)$, and denote $\mathcal{M}$ the set of all utility vectors that can be obtained for equilibria in the game.

Even if agents see clearly the best payoff, they still need to pick a single *same* equilibrium to implement, otherwise the payoff will not be realized. In our negotiation protocol, the agent who initiated the successful proposal on that particular utility vector arbitrarily picks their policy—a reward for making a good proposal.

Negotiations cannot go forever. If after a given number of negotiation rounds no consensus is reached, agents are forced to default to an egregious utility vector $\boldsymbol{\mu}_{\mathsf{dis}}$—subscript marks disagreement—associated with a given strategy profile. Once this becomes common knowledge, the negotiation proceeds as follows. A random agent from Π proposes the utility vector $\boldsymbol{\mu}$ it prefers in the sense that $\boldsymbol{\mu}[i] = \max_{\boldsymbol{u} \in \mathcal{U}} \boldsymbol{\mu}_i(\boldsymbol{u}) \geq \boldsymbol{\mu}_{\mathsf{dis}}[i]$. One by one, other agents either accept this proposal as is, or suggest an improvement without lowering the utility of anyone else who has accepted, or outright reject it by proposing a new one. A negotiation round is completed when a new proposal is made.

At any such time, the set of agents X that have accepted it is always nonempty—the proposing agent is always included. Before agent j rejects a proposal $\boldsymbol{\mu}$, it must look at who has already accepted it and compute the set of alternative proposals $\boldsymbol{\mu}'$ that will be accepted by all those agents. If the set is nonempty, j selects the best for itself improved proposal from there. If the set of alternative proposals is empty, j compares the current proposal $\boldsymbol{\mu}$ to the disagreement utility vector $\boldsymbol{\mu}_{\mathsf{dis}}$. If $\boldsymbol{\mu}[i] > \boldsymbol{\mu}_{\mathsf{dis}}[i]$, then j must compromise and accept $\boldsymbol{\mu}$. If $\boldsymbol{\mu}[i] \leq \boldsymbol{\mu}_{\mathsf{dis}}$, j has really nothing to lose by outright rejecting and suggesting a new proposal that serves its best interest, one that satisfies $\boldsymbol{\mu}'[j] = \max_{\boldsymbol{u} \in \mathcal{U}} \mu_j(\boldsymbol{u})$. A new round of negotiation starts. Algorithm 1 describes this negotiation protocol.

If an agreement is reached, then the implemented strategy profile is ensured to be a Pareto optimal equilibrium.

Lemma 1. *Given the set $\mathcal{M}$ of utility vectors for the pure Nash equilibria in a concurrent multi-agent game, if at least one utility vector which corresponds to a (set of) Pareto optimal equilibria exists, and*

Algorithm 1: `Negotiation`

Input: A set $\mathcal{M}$ of utility vectors where each $\boldsymbol{\mu} \in \mathcal{M}$ is obtained from a payoff vector that corresponds to an equilibrium; the initial state $q^{(0)}$; a disagreement utility vector $\boldsymbol{\mu}_{dis}$, and an upper bound *limit* on the number of negotiation rounds.

Output: A single equilibrium $\boldsymbol{S}$ agents have agreed upon.

begin

$proposer \leftarrow \texttt{Random}(\Pi)$; $done \leftarrow \mathsf{False}$; $\boldsymbol{\mu}^* \leftarrow \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}[proposer]$; $prop \leftarrow \boldsymbol{\mu}^*$; $k \leftarrow 1$; $X = \{proposer\}$;

while $\neg done$ *and* $k \leq limit$ **do**

$done \leftarrow \mathsf{True}$;

for $j \in \Pi \setminus X$ **do**

$X, newprop, accept, k \leftarrow \texttt{AcceptOrReject}\ (j, X, prop, \mathcal{M}, \boldsymbol{\mu}_{\mathsf{dis}}, k)$;

if $accept = \mathsf{False}$ **then**

$proposer \leftarrow j$, $prop \leftarrow newprop$, $done \leftarrow \mathsf{False}$

if $accept = \mathsf{True}$ **then**

if $prop \neq newprop$ **then**

$prop \leftarrow newprop$; $proposer \leftarrow j$; $done \leftarrow \mathsf{False}$.

if $done = \mathsf{True}$ **then**

return $\boldsymbol{S} = \texttt{Choose}(proposer, prop)$

`/* Agreement reached, and proposer selects an equilibrium corresponding to the agreed utility vector. */`

`/* Agreement not reached before limit rounds. */`

return $\boldsymbol{S} = \texttt{Choose}(proposer, \boldsymbol{\mu}_{\mathsf{dis}})$

the disagreement utility $\boldsymbol{\mu}_{\mathsf{dis}}$ *is chosen such that* $\boldsymbol{\mu}_{\mathsf{dis}}[i] < \boldsymbol{\mu}[i]\ \forall\ \boldsymbol{\mu} \in \mathcal{M}$ *and* $i \in \Pi$*, then Algorithm 1 will ensure agreement on a utility vector which corresponds to a (set of) Pareto optimal equilibria.*

Proof. Let the utility vector in the current proposal be $\boldsymbol{\mu}$. If it is not Pareto optimal, there is another utility vector $\boldsymbol{\mu}'$ such that for at least some agent i, $\boldsymbol{\mu}'[i] > \boldsymbol{\mu}[i]$ while for any other $k \in \Pi \setminus \{i\}$ it is $\boldsymbol{\mu}'[k] \geq \boldsymbol{\mu}[k]$. When agent i gets its turn, it will propose $\boldsymbol{\mu}'$ as an improvement to $\boldsymbol{\mu}$—based on Algorithm 2, any utility vector other than $\boldsymbol{\mu}_{\mathsf{dis}}$ agents will either accept as it is, or improve. Thus, the cardinality of the set of agents who have accepted the current proposal is monotonically increasing. The disagreement utility $\boldsymbol{\mu}_{\mathsf{dis}}$, being worse than any other utility vector, will never be proposed. Therefore, in at most as many steps as the number of agents, consensus will have been reached. In the final round, the set Y only includes utility vectors that correspond to Pareto optimal equilibria, unless it is empty. The last agent who decides whether to accept or to reject the current proposal, has to pick an alternative from Y. If Y is empty, this last agent cannot better itself without making someone else worse off. Backstepping to the one-before-last agent, we see that if this penultimate agent had improved on an existing proposal, the resulting utility vector must have been a Pareto optimal. Backward induction completes the reasoning: the agreed utility vector corresponds to a Pareto optimal equilibrium. □

Algorithm 2: `AcceptOrChoose`

Input: An agent j, a set X of agents, the proposal *prop*, the set $\mathcal{M}$ of utility vectors for negotiation, the disagreement utility vector $\boldsymbol{\mu}_{dis}$ and the negotiation round k.

Output: A set X of agents who accept the proposal, the current proposal *newprop*, a Boolean value *accept* indicates whether agent j accepts or rejects and the updated negotiation round k.

begin

  $Y \leftarrow \emptyset$; $accept = \mathsf{False}$;

  for $\boldsymbol{\mu} \in \mathcal{M}$ **do**

    if *For each* $\ell \in X$, $\boldsymbol{\mu}[\ell] \geq prop[\ell]$ *and* $\boldsymbol{\mu}[j] > prop[j]$ **then**

      $Y \leftarrow Y \cup \{\boldsymbol{\mu}\}$.

  if $Y \neq \emptyset$ **then**

    $X \leftarrow X \cup \{j\}$; $accept = \mathsf{True}$; $\boldsymbol{\mu}^* \leftarrow \arg\max_{\boldsymbol{\mu} \in Y}(\boldsymbol{\mu}[j])$; $newprop \leftarrow \boldsymbol{\mu}^*$;

    return X, *newprop*, *accept*, k;

  else

    if $prop[j] > \boldsymbol{\mu}_{dis}[j]$ **then**

      $X \leftarrow X \cup \{j\}$; $accept = \mathsf{True}$; ;

      return X, *prop*, *accept*, k;

    else

      $\boldsymbol{\mu}^* \leftarrow \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}[j]$; $accept = \mathsf{False}$; $k \leftarrow k + 1$; $newprop \leftarrow \boldsymbol{\mu}^*$; $X \leftarrow \{j\}$;

      return $X, newprop, accept, k$;

5 Example

Recall the example of Section 1, in which three agents need to visit different rooms in the environment of Fig. 1. Each agent may move to an adjacent room through the connecting door. Symbols a, b, c, and d represent the actions of crossing the corresponding door, and an additional symbol ϵ expresses inaction: staying in place. The set of actions for agent $i \in \Pi$ is thus $\Sigma_i = \{a, b, c, d, \epsilon\}$.

A fragment of the arena P is shown in Fig. 2, A transition of the form e.g. $ABC \xrightarrow{a,c,\epsilon} BDC$ means that agents 1, 2, and 3 are in rooms A, B and C, respectively, and that 1 crosses door a, 2 crosses door c, and 3 stays put. The agents will then arrive at rooms B, D and C. Figure 3 shows a fragment of the two-player game arena H, obtained from P.

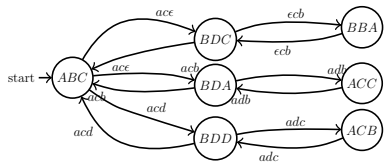

Figure 2: A fragment of the multi-agent arena $P = \left(Q, \mathcal{ACT}, T, q^{(0)}\right)$. A state (i, j, k) is represented as ijk—agent 1 is in room i, agent 2 in room j and agent 3 in room k. The initial state $q^{(0)} = (A, B, C)$. For all $(a_i)_\Pi \in \mathcal{ACT}$, and $i \neq j \in \Pi$, if $a_i, a_j \neq \epsilon$, then $a_i \neq a_j$ captures the constraint that two agents cannot pass through the same door simultaneously. $\mathcal{AP} = \{\alpha(i, m) :$ the robot i is in room $m, i \in \{1, 2, 3\}; m \in \{A, B, C, D\}\}$. The set of propositions evaluated true at $q \in Q$ indicates the current locations of agents.

Agent objectives are Büchi: infinitely often, agent 1 should visit A and B, agent 2 should visit C

and D, and agent 3 must visit rooms A, B, and D:

$$\varphi_1 : \Box(\Diamond(A \wedge \Diamond B)) \qquad \varphi_2 : \Box(\Diamond(C \wedge \Diamond D)) \qquad \varphi_3 : \Box(\Diamond(A \wedge \Diamond(B \wedge \Diamond D))) \ .$$

For each pay-off vector $\boldsymbol{u}$, computing a winning strategy in the two-player game $\mathcal{H}(\boldsymbol{u})$ takes an average of 38 seconds, with a Python implementation on a desktop with Intel(R) Core(TM) $i5$ processor and 16 GB RAM.

Let us see now how the design of preference orderings and the formation of teams affects the equilbria in the game.

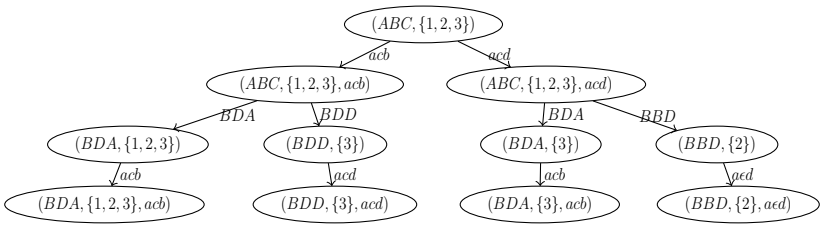

Figure 3: A fragment of the two-player turn-based game arena H. The semantics of a state in V_I (e.g., $(ABC, \{1,2,3\}, acb)$) is that agents are in the rooms marked by the first component (i.e., 1 in A, 2 in B and 3 in C), the agents suspect for triggering the transition there are the ones in the second component (i.e., all of them), and agents are supposed to execute the actions specified in the third component (i.e., 1 go through a, 2 go through c and 3 go through b). The semantics of a state in V_{II} (say, $(BDD, \{3\})$), is that agents are now where the first component says (i.e., 1 in B, 2 in D and 3 also in D) and that for this state to have been reached, the agents in the second component (i.e., 3) are suspect of triggering the transition: the action profile that was actually implemented to reach that particular state in V_{II} is acd. By comparing acd with acb, it is clear that 3 deviates.

5.1 Equilibrium analysis

Case 1: No teams—everyone for themselves Agents selfishly focus on achieving their own objectives, which means that their preference relations are

$$\boldsymbol{u} \lesssim_i \boldsymbol{u}' \text{ with } i \in \Pi \text{ if } \boldsymbol{u}[i] = 0 \text{ and } \boldsymbol{u}'[i] = 1 \ , \quad \boldsymbol{u} \simeq_i \boldsymbol{u}' \text{ if } \boldsymbol{u}[i] = \boldsymbol{u}'[i] \ .$$

In this case, the two-player game $\mathcal{H}$, has payoff vector set $\mathcal{PV} = \{0,1\}^3$. Analyzing all payoff vectors in $\mathcal{PV}$, reveals that there exists an equilibrium for every one of them (see first row of Table 1).

Case 2: Selfish individuals in teams. Let $\mathsf{Teams} = \big\{\{1\}, \{2\}, \{1,2\}, \{3\}\big\}$; agents 1 and 2 can now cooperate in an ad-hoc way, but also deviate unilaterally as individuals. In this case we also have a cooperative equilibrium to realize every payoff vector (see second row of Table 1).

Case 3: Implicit teaming against agent 3. Now we do not explicitly define possible teams; we allow agents to choose by themselves how to team up based on their preference relations. We select preference relations to radicalize agents 1 and 2: they now prefer failure to letting agent 3 get his way. For agent 1 we have:

$$(0,0,1) \lesssim_1 (0,1,1) \lesssim_1 (1,0,1) \lesssim_1 (1,1,1) \lesssim_1 (0,0,0) \lesssim_1 (0,1,0) \lesssim_1 (1,0,0) \lesssim_1 (1,1,0) \ , \quad (3)$$

which reads "ideally I want myself and agent 2 to achieve our goals but not agent 3, and if I cannot have that I would rather win alone; if this is not possible I can let agent 2 win, but under no circumstances do I let 3 get his way—in case 3 wins, my preferences coincide with the case where he loses." Similarly for agent 2,

$$(0,0,1) \lesssim_2 (1,0,1) \lesssim_2 (0,1,1) \lesssim_2 (1,1,1) \lesssim_2 (0,0,0) \lesssim_2 (1,0,0) \lesssim_2 (0,1,0) \lesssim_2 (1,1,0) \ . \quad (4)$$

Agent 3 plays as in case 1. Now we see that there is no equilibrium corresponding to payoff vectors $(0,0,1)$ or $(0,1,1)$; the opportunistic alliance of agents 1 and 2, will have them both sacrifice for seeing agent 3 fail.

Table 1: Nash equilibria for all payoff vectors in concurrent game $\mathcal{G}$ with Büchi objectives

	$\mathcal{PV}$							
	(0,0,0)	(0,0,1)	(0,1,0)	(1,0,0)	(1,1,0)	(0,1,1)	(1,0,1)	(1,1,1)
case 1	✓	✓	✓	✓	✓	✓	✓	✓
case 2	✓	✓	✓	✓	✓	✓	✓	✓
case 3	✓	✗	✓	✓	✓	✗	✓	✓

5.2 Agreeing on strategies through negotiation

Negotiation is needed if coordination is to be decentralized. Here we will not consider team deviations, and thus focus on cases 1 and 3.

For case 1, the condition in Lemma 1 is satisfied because for each agent, there exists a utility vector which is strictly preferred by this agent to the one corresponding to the payoff vector $(0,0,0)$. Thus, if we choose the disagreement utility vector be the one that corresponds to $(0,0,0)$, there is a unique outcome of negotiation which corresponds to payoff vector $(1,1,1)$, for which all agents accomplish their goals.

For case 3, we see that for agents 1 and 2 the least preferable equilibrium payoff vector is $(1,0,1)$, while for agent 3 it is any of the form $\{(x,y,0) \mid x,y \in \{1,0\}\}$. The condition in Lemma 1 is no longer

satisfied. In fact, if the disagreement utility vector is chosen among those that correspond to any payoff vector in $\{(0,0,0),(0,1,0),(1,0,0),(1,1,0),(1,1,1)\}$, agents will default to the disagreement equilibrium policy. If, however, the disagreement utility vector is the one associated with payoff vector $(1,0,1)$, the negotiation steers the agents toward their (best) payoff vector $(1,1,1)$. To see this, suppose agent 1 first proposes $(1,1,0)$; agent 2 will agree and accept. However, agent 3 finds this no better than the disagreement utility vector, and will therefore reject and propose $(1,0,1)$. Agent 1 gets its turn, and finds a set Y of utility vectors using Algorithm 2, that correspond to payoff vectors $\{(1,1,1)\}$. Now agent 1 suggests an improvement in the form of a utility vector that corresponds to a payoff $(1,1,1)$. Agent 2 takes his turn and computes a set Y that turns out to be empty: that tells it that the current proposal cannot be improved further, while being strictly better than the disagreement policy—agent 2 has to agree. The negotiation terminates.

6 Conclusion and future work

In an instance of a multi-agent coordination problem where agents act concurrently and have their own objectives and preferences over the outcomes of the interaction between them, discrete planning and control synthesis can be performed within a game theoretic framework. Effective coordination policies take the form of Nash equilibria. This paper reports on methods for identifying these behaviors, particularly when agent objectives are expressed in temporal logic. Allowing agents to team up if they see that doing so yields a better outcome, gives rise to new type of cooperative equilibria, which can be treated within the same proposed framework.

If coordination is to be achieved in a decentralized way, agents need a way of selecting a single, common equilibrium to implement; mixing individual behaviors from different game equilibria may not give another equilibrium. This agreement can be achieved through negotiation, and the algorithm reported here ensures that the equilibrium behavior that agents reach consensus over is Pareto optimal, assuming that certain conditions on the agents' objectives, preferences and the disagreement policy in the negotiation are satisfied.

The framework described cannot currently capture mixed Nash equilibria. The algorithms for computing the game equilibria are of polynomial time complexity, but this obviously cannot overcome the challenge posed by the curse of dimensionality. Additional measures for curbing the increase in the dimension of the product systems are being investigated, trying to take advantage of the fact that agent coupling may occur over subsets of their operational space.

References

[1] R. Alur and S. La Torre. Deterministic generators and games for ltl fragments. In *Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on*, pages 291–300, 2001.

[2] Krzysztof R Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.

[3] André Arnold. Synchronized products of transition systems and their analysis. In Jörg Desel and Manuel Silva, editors, *Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 26–27. Springer Berlin Heidelberg, 1998.

[4] Mohamed-Amine Belhaj Seboui, Nejib Ben Hadj-Alouane, Gwenael Delaval, Eric Rutten, and Moez Yeddes. A decentralized supervisory control approach for distributed adaptive systems. In *Proceedings of the Fourth international conference on Verification and Evaluation of Computer and Communication Systems*, pages 13–23. British Computer Society, 2010.

[5] Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George Pappas. Symbolic planning and control of robot motion. *IEEE Robotics Automation Magazine*, 14(1):61–70, 2007.

[6] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Nash equilibria in concurrent games with Büchi objectives. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 13, pages 375–386. Schloss Dagstuhl, 2011.

[7] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Concurrent games with ordered objectives. In Lars Birkedal, editor, *Foundations of Software Science and Computational Structures*, volume 7213 of *Lecture Notes in Computer Science*, pages 301–315. Springer Berlin Heidelberg, 2012.

[8] Yushan Chen, Xu Chu Ding, and Calin Belta. Synthesis of distributed control and communication schemes synthesis of distributed control and communication schemes from global ltl specifications. In *IEEE Conference on Decision and Control*, pages 2718–2723, Orlando FL, 2011.

[9] E Allen Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, 2:995–1072, 1990.

[10] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 190–204, 2010.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[12] Madhavan Mukund. *From Global Specifications to Distributed Implementations*, pages 19—34. Kluwer Academic Publishers, 2002.

[13] Chris Murray and Geoffrey Gordon. *Multi-robot negotiation: approximating the set of subgame perfect equilibria in general-sum stochastic games.* Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

[14] Dominique Perrin and Jean Éric Pin. *Infinite words: automata, semigroups, logic and games.* Elsevier, 2004.

[15] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. In *Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

[16] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001.

[17] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2009.

[18] Paulo Tabuada. Approximate simulation relations and finite abstractions of quantized control systems. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 529–542. Springer-Verlag, 2007.

[19] Herbert Tanner, Jie Fu, Chetan Rawal, Jorge Piovesan, and Chaouki Abdallah. Finite abstractions for hybrid systems with stable continuous dynamics. *Discrete Event Dynamic Systems*, 22:83–99, 2012.

[20] Wolfgang Thomas. Infinite games and verification. In *Proceedings of the 14th International Conference on Computer Aided Verification*, pages 58–64, London, UK, 2002. Springer-Verlag.

[21] Alphan Ulusoy, Stephen L. Smith, Xu Chu Ding, and Calin Belta. Robust multi-robot optimal path planning with temporal logic constraints. *International Journal of Robotics Research*, submitted, 2012.

[22] T-S Yoo and Stéphane Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems*, 12(3):335–377, 2002.