Published in IET Control Theory and Applications
Received on 30th May 2014
Revised on 9th September 2014
Accepted on 24th September 2014
doi: 10.1049/iet-cta.2014.0611

Special Issue on Co-operative Multi-Agent Systems with Engineering Applications

ISSN 1751-8644

Concurrent multi-agent systems with temporal logic objectives: game theoretic analysis and planning through negotiation

Jie Fu[1], *Herbert G. Tanner*[2], *Jeffrey Heinz*[3]

[1] *Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA*
[2] *Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA*
[3] *Department of Linguistics and Cognitive Science, University of Delaware, Newark, DE 19716, USA*
E-mail: jief@seas.upenn.edu.

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

1 Introduction

We analyse the concurrent interaction of multiple heterogeneous dynamical systems, each trying to serve its own objective. Instances of this interaction can be found in economics, social sciences, as well, of course, as in engineering. In the latter case, a large-scale example of such interaction can be thought to be the power grid, which includes collection of different types of producers and consumers, and interaction policies involve changes in production, consumption and pricing levels.

We model the interacting dynamical systems as finite-state transition systems, express their objectives in terms of temporal logic formulae and encode their interaction in the form of a concurrent game. Since temporal logic formulae result in games with infinite stages with outcomes being evaluated differently compared to classical finite games, the questions we address is how to identify stable (in a Nash equilibrium sense) interaction behaviours in these games, and then how to coordinate on a particular behaviour, given the agents' individual objectives and preferences.

Agent objectives are encoded in linear temporal logic (LTL). Owing to its expressiveness, LTL is widely used to specify many desired system properties such as safety, liveness, persistance etc. [1]. So far, (supervisory) control problems with temporal logic specifications, have been approached using primarily top-down, centralised 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 [2], and decentralised supervisory control [3, 4]. When the global task is in the form of an LTL formula, methods have been developed [5] to break up the global specification into a set of control and communication policies for each agent to follow. Centralised controllers can also be synthesised in the face of modelling uncertainty [6]. 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 [7].

In these approaches, the correctness of the entire system's behaviour 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 [1, 8, 9], algorithmic game theory [10] provides at the abstraction level a natural framework for the composition of systems. The notion of 'rational synthesis' [11] allows to frame the control problem for this class of multi-agent systems inside a non-zero-sum game theoretic framework (cf. [7]): 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 [11, 12] along a direction of negotiation-based behaviour planning in multi-agent systems with temporal logic control objectives. The formulation in this paper differs in the way agents' utilities are expressed and agents'

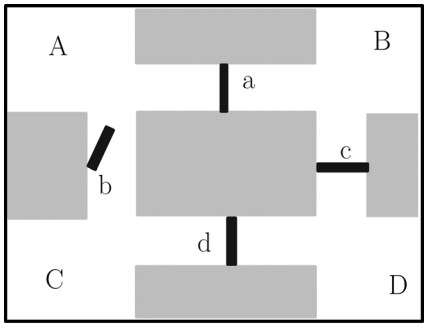

Fig. 1 *Partitioned rectangular environment in which three agents roam*

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

preferences are encoded. In existing results, with agents having independent objectives either as a Boolean utility value [11] or a set of ranked objectives [12], implicit cooperation between agents is not encouraged. Furthermore, 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 to a surveillance task that requires it to visit a sequence of rooms repeatedly. The agents' 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 an outcome in which all agents accomplish their task. However, if agents 1 and 2 see agent 3 as a common adversary, depending on 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-centred design with a new definition of agents' utilities, that allows implicit cooperation to emerge as an equilibrium. We decide which interaction outcomes are stable (in a Nash sense) in the non-cooperative, concurrent game the agents are engaged in. Here agents are cooperating implicitly in the sense that an agent will cooperate with another if the success of both makes each happier than succeeding alone.

We then model explicit cooperation (teaming up), using 'cooperative games with temporal logic objectives' in which agents can form coalitions. A decision procedure for cooperative equilibria is provided. To coordinate, agents communicate and negotiate a mutually accepted plan. Inspired by Murray and Gordon [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.

We present examples in Section 5 showing how these theoretical results allow us to model cases where every agent acts selfishly, where agents can choose to act selfishly or choose to cooperate explicitly in teams, or choose to form teams based on their preferred outcomes of 'other' agents' objectives.

2 Preliminaries

2.1 Automata and semi-automata

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' semi-automaton (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: $T(q, \lambda) = q$ and for all $\sigma \in \Sigma, w \in \Sigma^*$, $T(q, w\sigma) = T(T(q, w), \sigma)$. We write $T(q, \sigma) \downarrow$ to specify that a transition labelled 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 $\mathsf{sink} \notin Q$, 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 $q^{(0)}$ being the initial state, and WIN the 'winning condition', discussed below. A run in the game is an infinite sequence of states $\rho \in V^\omega$.

In this paper, we consider three types of winning conditions: 'Büchi', 'coBüchi' and 'Muller'. For a 'Büchi objective', a run ρ is winning for player 1 if and only if $\mathsf{Inf}(\rho) \cap F \neq \varnothing$. For a 'coBüchi objective', a run ρ is winning for player 1 if and only if $\mathsf{Inf}(\rho) \subseteq F$. The winning condition for a 'Muller objective' is given in the form of a finite set of subsets of states $\mathcal{F} = \{F_1, \ldots, F_n\}$ with $F_i \subseteq V$ for all $1 \leq i \leq n$. A run ρ is winning for player 1 if and only if $\mathsf{Inf}(\rho) \in \mathcal{F}$.

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

doi: 10.1049/iet-cta.2014.0611

strategy is called 'the winning region' for this player and is denoted Win_i. For two-player turn-based zero-sum games with Büchi, coBüchi or Muller objectives for player 1, 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 game with Büchi, coBüchi and Muller objectives are found in [15, 16].

2.3 Specification language

We consider a fragment of LTL [17] for specifying 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). Formulae in an LTL fragment [17, 18] specify desired system properties such as reachability, safety and liveness, recurrence etc., and each formula in this class is equivalently expressed as a DBA with alphabet $2^{\mathcal{AP}}$.

3 Multi-agent concurrent Büchi games

This section presents the main theoretical results in this paper. In Section 3.1, we define multiplayer concurrent games where each agent, or player, is tasked with satisfying a temporal logic objective equivalently expressed as the language accepted by a DBA. We also introduce the game 'arena', which is an automaton describing all the possible interactions of agents, given any state of the overall system. The notion of 'arena' is to be distinguished from the notion of 'game'. The former is like the chessboard and the rules explaining how pieces move whereas the latter is like a standard game of chess: in addition to the board and the rules governing how pieces move, the initial positions of the pieces are provided, as is the objective of capturing the other player's king.

In Section 3.2, we introduce 'payoff vectors' and players' preference orderings, which are necessary for defining pure Nash equilibria defined in this section. In Section 3.3, we formally define the problem we are addressing: Does a given concurrent multiplayer game have any pure Nash equilibria? Following [19], we show how to answer this question with a two-player game we construct. We build the two-player game in stages: first we transform the multiplayer arena to a two-player arena; then we combine this arena with the agents' objectives; and finally we include the different winning conditions. We show how the the existence of equilibria for different payoff vectors follows (in part) from the presence or absence of winning strategies for one of the players in this particular two-player game.

Section 3.4 extends the above analysis to handle the case where agents form teams and cooperate explicitly. Here the emerging behaviour of the agents is captured by another solution concept, called cooperative equilibrium. Decision procedures for cooperative equilibria are provided.

3.1 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 $\mathcal{AP}$ [20]. Atomic propositions and their negations are 'literals', combined in conjunction to form 'sentences'. All possible sentences form 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 'semi-automaton' augmented with a 'labelling function' that maps every state to a sentence which is true at that state.

Let $\Pi = \{1, \ldots, N\}$ be an index set and 2^{Π} denote its power set, that is, the set of all subsets of Π. Given a tuple $\vec{s} = (s_1, \ldots, s_N)$ denote $\vec{s}\,[i] = s_i$ the ith entry of $\vec{s}$. An agent is modelled as a labelled semi-automaton (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 labelling 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 modelled 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 $P = A_1 \circ A_2 \circ \cdots \circ A_N = \left(Q, \mathcal{ACT}, T, q^{(0)}, \mathsf{LB}\right)$ where

- Q $\subseteq Q_1 \times Q_2 \times \cdots \times Q_N$ is the set of states.
- $\mathcal{ACT}$ $= \Sigma_1 \times \cdots \times \Sigma_N$ is the alphabet. Each $\vec{a} = (a_1, a_2, \ldots, a_N) \in \mathcal{ACT}$ 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 $\vec{a} = (a_1, \ldots, a_N) \in \mathcal{ACT}$, we have $T(q, \vec{a}) = T((q_1, \ldots, q_N), (a_1, \ldots, a_N)) = (q'_1, \ldots, q'_N)$ 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 labelling 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. Essentially, the arena is a game without encoding agents' objectives.

The objective of agent i is specified with an LTL fragment formula φ_i and can be expressed as an ω-regular language $\Omega_i \subseteq (2^{\mathcal{AP}})^{\omega}$, which is accepted by a 'total' DBA $\mathcal{A}_i = \left(S_i, 2^{\mathcal{AP}}, T_i, I_i, F_i\right)$ 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. [19]). We write $\mathsf{Mov}(q, i) = \{\vec{a}[i] \in \Sigma_i \mid T(q, \vec{a})\downarrow\}$, where T is the transition function in P. A 'play' $\mathsf{p} = q^{(0)}\vec{a}^{(0)}q^{(1)}\vec{a}^{(1)}q^{(2)}\vec{a}^{(2)}\ldots$ is an interleaving sequence of states and action profiles, such that for all $i \geq 0$, we have $T(q^{(i)}, \vec{a}^{(i)}) = q^{(i+1)}$. 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 $\forall \rho = q^{(1)}q^{(2)}\ldots \in Q^*$, $\mathsf{S}_i(\mathsf{Pr}^{=k}(\rho)) \in \mathsf{Mov}(q^{(k-1)}, i)$, for $1 \leq k$. A 'deterministic strategy profile' $\vec{\mathsf{S}} = (\mathsf{S}_1, \ldots, \mathsf{S}_N)$ 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 $\vec{\mathsf{S}} = (\mathsf{S}_1, \ldots, \mathsf{S}_N)$ if it is produced when every agent

i adheres to strategy S_i. All runs compatible with strategy profile $\vec{\mathsf{S}}$ form the set of game 'outcomes' for this strategy profile, denoted $\mathsf{Out}(q^{(0)}, \vec{\mathsf{S}})$.

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 $\mathrm{u}_i : \mathcal{Q} \times \mathcal{SP} \to \{0, 1\}$ from the set of states $\mathcal{Q}$ and 'strategy profiles' $\mathcal{SP}$ defined as $\mathrm{u}_i(q, \vec{\mathsf{S}}) = 1$ if for all 'runs' ρ in $\mathsf{Out}(q, \vec{\mathsf{S}}) \subseteq \mathcal{Q}^\omega$, we have $\mathsf{LB}(\rho) \in \Omega_i$. The payoff vector is the tuple made of the payoffs of all agents: $\mathbf{u}(q, \vec{\mathsf{S}}) = (\mathrm{u}_1(q, \vec{\mathsf{S}}), \ldots, \mathrm{u}_N(q, \vec{\mathsf{S}}))$; we say that strategy profile $\vec{\mathsf{S}}$ yields the payoff vector $\mathbf{u}(q, \vec{\mathsf{S}})$. The set of all possible payoff vectors is denoted $\mathcal{PV} = \bigcup_{\vec{\mathsf{S}} \in \mathcal{SP}} \mathbf{u}(q^{(0)}, \vec{\mathsf{S}})$.

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

Definition 2 cf. [11]*:* A deterministic strategy profile $\vec{\mathsf{S}}$ is a 'pure Nash equilibrium' in a multi-agent non-cooperative concurrent game if any other strategy profile $\vec{\mathsf{S}}'$ obtained by agent $i \in \Pi$ 'unilaterally' deviating from one action profile given by $\vec{\mathsf{S}}$, results in $\mathbf{u}(q^{(0)}, \vec{\mathsf{S}}') \lesssim_i \mathbf{u}(q^{(0)}, \vec{\mathsf{S}})$.

This paper considers only pure Nash equilibria, henceforth just referred to as 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 [21]. We pose the following question:

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

In the literature, each agent either has a single temporal logic specification [11], or a whole set of objectives ranked according to its own preference relation [12]. 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. [22]*:* Consider a set of semi-automata with designated initial states $A_i = (Q_i, \Sigma_i, T_i, q_i^{(0)})$, for $1 \leq i \leq n$. Their 'synchronised product' is a tuple

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

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 [12] 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 $\vec{b}$ that would have brought them to state $T(q, \vec{b}) = q''$, but instead the game landed at state $q' \neq q''$. One of the players, say $k \in \Pi$, unilaterally deviated from profile $\vec{b}$ and played σ instead of $\vec{b}[k]$, resulting in $T(q, \vec{a}) = q''$ where $\vec{a} = (b_1, \ldots, b_{k-1}, \sigma, b_{k+1}, \ldots, b_N)$. This new action profile $\vec{a}$ is denoted $\vec{b}[k \mapsto \sigma]$ to emphasise that it is produced from $\vec{b}$ by swapping $\vec{b}[k]$ with σ. For action profile $\vec{b}$, the suspect players triggering a transition from q to q' is

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

To solve Problem 1, the concurrent arena $P = (Q, \mathcal{ACT}, T, q^{(0)}, \mathsf{LB})$ is transformed to the arena of a two-player turn-based game with two fictitious players: player I and player II [12]. This arena is a semi-automaton $H = (V, \mathcal{ACT} \cup Q, T_h, v^{(0)})$, with components defined as follows:

- V $= V_{\mathrm{I}} \cup V_{\mathrm{II}}$ is the state space, with $V_I \subseteq Q \times 2^{\Pi}$, and $V_{\mathrm{II}} \subseteq Q \times 2^{\Pi} \times \mathcal{ACT}$.
- $\mathcal{ACT} \cup Q$ 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) $v = (q, X) \in V_{\mathrm{I}}$ and if for any $\vec{a} \in \mathcal{ACT}$ it is $T(q, \vec{a}) \downarrow$, then $T_h((q, X), \vec{a}) := (q, X, \vec{a}) \in V_{\mathrm{II}}$; or

  (ii) $v = (q, X, \vec{a}) \in V_{\mathrm{II}}$ and if for any $q' \in Q$ it is $X' = X \cap \mathsf{Susp}((q, q'), \vec{a}) \neq \varnothing$, then $T_h(v, q') := (q', X') \in V_I$.
- $v^{(0)}$ $= (q^{(0)}, \Pi)$ is the initial state.

Here, 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 emphasise that the automaton $\mathcal{A}_i$ has initial state $s_i^{(0)} = T_i(I_i, \mathsf{LB}(q^{(0)}))$. The objectives $\{\Omega_i\}_\Pi$ are incorporated into the description of the two-player arena H using the synchronised product (Definition 3)

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

with components are described as follows:

- $\hat{V}$ $= \hat{V}_{\mathrm{I}} \cup \hat{V}_{\mathrm{II}}$ where $\hat{V}_{\mathrm{I}} = V_{\mathrm{I}} \times S_1 \times \cdots \times S_N$, with S_i the states of $\mathcal{A}_i$, are the states where player I takes a transition. $\hat{V}_{\mathrm{II}} = V_{\mathrm{II}} \times S_1 \times \cdots \times S_N$ 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 $v' = T_h(v, \sigma)$;

doi: 10.1049/iet-cta.2014.0611

• 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 $\pi_1(v^{(0)}) = q^{(0)}$.

The object $\mathcal{H}$ given by the synchronised product above in (1) is neither an arena (because it contains objectives) nor a game (because the winning conditions are absent). Winning conditions for $\mathcal{H}$ will be added soon below in (2) and (3) to complete the two-player game.

For a state $\hat{v} = (v, \vec{s})$ in the two-player game, v can be either in $\hat{V}_{\mathrm{I}}$ or in $\hat{V}_{\mathrm{II}}$. In the first case $(v,\vec{s}) = ((q,X),\vec{s})$ with $q \in Q$ and $X \in 2^{\Pi}$; in the second, $(v,\vec{s}) = ((q,X,\vec{a}),\vec{s})$ with $\vec{a} \in \mathcal{ACT}$. 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 $i \geq 0$, $\mathsf{Agt}(\hat{v}^{(i)}) = \mathsf{Agt}(\hat{v}^{(i+1)}) = \Pi$.

Let us denote the set $\hat{F}_i = \{\hat{v} = (v, s_1, s_2, \ldots, s_N) \in \hat{V} \mid s_i \in F_i\}$. Informally, $\hat{F}_i$ is a set of states, in each of which the specification state of the DBA $\mathcal{A}_i$ is in F_i. State $\hat{v} = (v, s_1, s_2, \ldots, s_N)$ can be associated with 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 $\hat{v}[i+1] \in \hat{F}_i$, and $\mathsf{Val}(\hat{v})[i] = 0$ otherwise. We extend the value function over sets of states as follows: for a given set $S \subseteq \hat{V}$, $\mathsf{Val}(S)$ is defined if and only if for any $\hat{v}, \hat{v}' \in S$, $\mathsf{Agt}(\hat{v}) = \mathsf{Agt}(\hat{v}')$. Given a set S for which $\mathsf{Val}(S)$ is defined, let $\mathsf{Agt}(S) = \mathsf{Agt}(\hat{v})$, for any $\hat{v} \in S$ and define $\mathsf{Val}(S)[i] = 1$ if $S \cap \hat{F}_i \neq \varnothing$, and $\mathsf{Val}(S)[i] = 0$ otherwise.

Lemma 1: Given a run $\rho \in \hat{V}^{\omega}$, if $\mathsf{Inf}(\rho) = S$, then for any $\hat{v}, \hat{v}' \in S$, it holds that $\mathsf{Agt}(\hat{v}) = \mathsf{Agt}(\hat{v}')$, and so $\mathsf{Val}(S)$ is defined.

Proof: For run $\rho = \hat{v}_0\hat{v}_1 \ldots$ and for any $i \in \mathbb{N}$, the intersection of suspect players yields $\mathsf{Agt}(\hat{v}_i) \subseteq \mathsf{Agt}(\hat{v}_{i+1})$. Since $\mathsf{Inf}(\rho) = S$, then there must exist $i,j \in \mathbb{N}$, $i \leq j$, $\hat{v}_i = \hat{v}_j$, and thus there is no reduction in the set of agents within the subsequence of ρ from $\hat{v}_i$ to $\hat{v}_j$. Since there are infinitely many such i,j, there will be no reduction in the set of agents on the subsequence of ρ which visits every state in S infinite often. □

Proposition 1: Consider a concurrent game played on arena P by players Π with objectives $\{\mathcal{A}_i \mid i \in \Pi\}$. Each player $i \in \Pi$ has a preference ordering over $\mathcal{PV}$. If in the two-player turn-based game $\mathcal{H}(\vec{u})$ with 'Muller' objectives

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

where $\vec{u} \in \mathcal{PV}$, $\mathcal{F}(\vec{u}) = \{S \subseteq \hat{V} \mid \forall i \in \mathsf{Agt}(S), \mathsf{Val}(S) \lesssim_i \vec{u}\}$, 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 $\rho^{(i)} \in \mathsf{Win}_I$, $\mathsf{Agt}(\rho^{(i)}) = \Pi$, and $\mathsf{Val}(\mathsf{Inf}(\rho)) = \vec{u}$.

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

Proof: For Muller objectives, by definition, a run ρ is winning for player 1 if and only $\mathsf{Inf}(\rho) \in \mathcal{F}$. When player 1 applies its winning strategy, runs are forced to visit every state in S infinitely often, for some $S \in \mathcal{F}$. For any agent $i \in \mathsf{Agt}(S)$, the payoff vector associated with such a run, $\mathsf{Val}(S)$, cannot be preferable to payoff vector $\vec{u}$. Condition 2 ensures the existence of an equilibrium $\vec{\mathsf{S}}$ with the payoff vector $\vec{u}$, in the form of an infinite sequence of action profiles: if w is an ω-word that generates a run ρ in $\mathcal{H}(\vec{u})$ satisfying condition 2, then equilibrium $\vec{\mathsf{S}}$ is obtained by the projection of w on the set of action profiles. □

For a game with a Muller objective, there are existing tools and methods [16] for synthesising the winning strategy for player 1, whenever it exists. The challenge is that the set of subsets that defines the winning condition for Muller objectives can be exponentially large in the size of the two-player turn-based game.

In what follows, we study a particular case in which it is not necessary to solve Muller games in order to determine pure Nash equilibria. When all agents are selfish, whether there exists an equilibrium associated with a given pay-off vector $\vec{u}$ in the multi-agent concurrent 'Büchi' game can be determined by solving the two-player turn-based game with a 'coBüchi' objective induced by $\vec{u}$. Furthermore, as we explain below, this case avoids the exponential growth that can occur in the general case with Muller objectives.

Proposition 2: Consider a concurrent game played on arena P by players Π with objectives $\{\mathcal{A}_i \mid i \in \Pi\}$. Each player $i \in \Pi$ has a 'selfish' preference ordering over $\mathcal{PV}$. If in the two-player turn-based 'coBüchi' game $\mathcal{H}(\vec{u})$ of the form

$$\mathcal{H}(\vec{u}) = (\hat{V}, \mathcal{ACT} \cup Q, \hat{T}, \hat{v}^{(0)}, F(\vec{u})) \tag{3}$$

where $\vec{u} \in \mathcal{PV}$, $F(\vec{u}) = \{\hat{v} \in \hat{V} \mid \forall i \in \mathsf{Agt}(\hat{v}), \mathsf{Val}(\hat{v}) \lesssim_i \vec{u}\}$, 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 $\rho^{(i)} \in \mathsf{Win}_I$, $\mathsf{Agt}(\rho^{(i)}) = \Pi$, and $\mathsf{Val}(\mathsf{Inf}(\rho)) = \vec{u}$.

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

Proof: By condition 1, states visited infinitely often are in $F(\vec{u})$. Consider an arbitrary set $S \subseteq F(\vec{u})$ of states that is infinitely often visited. By the definition of selfish preference ordering $\lesssim_i$ and that of $F(\vec{u})$, for any agent $i \in \mathsf{Agt}(S)$, if $\vec{u}[i] = 1$, then either $S \cap \hat{F}_i \neq \varnothing$ or $S \cap \hat{F}_i = \varnothing$ and the payoff vector $\vec{u}'$ received by visiting S infinitely often is such that $\vec{u}'[i] = 1$ or $\vec{u}'[i] = 0$. In either case, $\vec{u}'[i] \leq \vec{u}[i]$ and thus $\vec{u}' \lesssim_i \vec{u}$. That is, agent i will not prefer the outcome with payoff vector $\vec{u}'$ which is triggered by his deviation over the outcome with payoff vector $\vec{u}$. Similarly, when $\vec{u}[i] = 0$, we have $S \cap \hat{F}_i = \varnothing$. By the same token, payoff vector $\vec{u}'$ received by agent i by visiting S infinitely often will not be preferable over payoff vector $\vec{u}$. Since the set S is arbitrary chosen, it follows that no agent can produce by deviating an outcome with payoff vector preferable to $\vec{u}$. Condition 2 ensures the existence of an equilibrium $\vec{\mathsf{S}}$, which is an infinite sequence of action profiles associated with $\vec{u}$: if w is an

ω-word that generates a run ρ in $\mathcal{H}(\vec{u})$, then the equilibrium $\vec{\mathsf{S}}$ 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 $\mathcal{H}(\vec{u})$ [15]. Constructing the two-player turn-based game $\mathcal{H}$ is polynomial in the size of P and the the specification automata $\mathcal{A}_i$ [19].

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, characterised 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 $\vec{a}$ is denoted $\vec{a}[X \mapsto \vec{\sigma}\,] = (a'_1, a'_2, \ldots, a'_N)$, where $\vec{\sigma} = (b_j)_{j\in X}$ 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 $\vec{\mathsf{S}}$ is a 'cooperative equilibrium' in a multi-agent non-cooperative game if for any team $X \in \mathsf{Teams}$, and for any strategy profile $\vec{\mathsf{S}}'$ obtained from $\vec{\mathsf{S}}$ by a unilateral team deviation of X, it holds that for all $k \in X$, $\mathbf{u}(q^{(0)}, \vec{\mathsf{S}}') \lesssim_k \mathbf{u}(q^{(0)}, \vec{\mathsf{S}})$.

If $T(q, \vec{a}\,) = q'$ is defined, then for an action profile $\vec{b} \in \mathcal{ACT}$ the set of suspect teams triggering a transition from q to q' is

$$\begin{aligned}&\mathsf{SuspTeams}((q,q'),\vec{b})\\ &\quad := \{X \in \mathsf{Teams} \mid (\forall i \in X)[(\exists \sigma_i \in \mathsf{Mov}(q,i))\\ &\qquad \times [\vec{b}[X \mapsto (\sigma_i)_{i\in X}] = \vec{a} \wedge T(q,\vec{a}) = q']]\}\end{aligned}$$

The two-player turn-based arena H where teams can deviate is constructed as

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

where

- V = $V_{\mathrm{I}} \cup V_{\mathrm{II}}$ is the set of states, with $V_{\mathrm{I}} \subseteq Q \times 2^{\mathsf{Teams}}$ and $V_{\mathrm{II}} \subseteq Q \times 2^{\mathsf{Teams}} \times \mathcal{ACT}$.
- $\mathcal{ACT} \cup Q$ 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) $v = (q, \mathcal{X}) \in V_{\mathrm{I}}$ where $\mathcal{X} \subseteq \mathsf{Teams}$ and for any $\vec{a} \in \mathcal{ACT}$ we have $T(q, \vec{a}\,) \downarrow$, in which case $T_h((q,\mathcal{X}),\vec{a}) := (q, \mathcal{X}, \vec{a}) \in V_{\mathrm{II}}$; or
  (ii) $v = (q, \mathcal{X}, \vec{a}) \in V_{\mathrm{II}}$ and for any $q' \in Q$ we have $\mathsf{SuspTeams}((q,q'),\vec{a}) \cap \mathcal{X} \neq \varnothing$, in which case $T_h(v, q') := (q', \mathcal{X}') \in V_{\mathrm{I}}$ where $\mathcal{X}' = \{X \in \mathsf{Teams} \mid X \subseteq Y, Y \in \mathsf{SuspTeams}((q,q'),\vec{a}) \cap \mathcal{X}\}$.

- $v^{(0)}$ = $(q^{(0)}, \mathsf{Teams})$ is the initial state.

In this game, opportunistic teams of agents play against each other if the interests of the teammates align. The sychronisation product between H and the set of DBAs $(\mathcal{A}_i, s_i^0)$ for $i \in \Pi$ is a two-player turn-based game $\mathcal{H} = H \ltimes (\mathcal{A}_1, s_1^0) \ltimes \cdots \ltimes (\mathcal{A}_N, s_N^0) = (\hat{V}, \mathcal{ACT} \cup Q, \hat{T}, \hat{v}^{(0)})$. For any state $\hat{v} \in \hat{V}$, let $\mathsf{Teams}(\hat{v}) = \pi_2(\pi_1(\hat{v}))$, which includes the set of teams in the state $\hat{v}$. A result similar to Lemma 1 can be proven to show that for a run $\rho \in \hat{V}^{\omega}$, for any $\hat{v}, \hat{v}' \in \mathsf{Inf}(\rho)$, $\mathsf{Teams}(\hat{v}) = \mathsf{Teams}(\hat{v}')$.

The analysis of equilibria is performed as in Section 3.3: we determine if an equilibrium with respect to a payoff vector $\vec{u}$ exists based on Propositions 1 and 2. The definition of the winning condition is slightly different here: in Proposition 1, given $\vec{u} \in \{0,1\}^N$, $\mathcal{F}(\vec{u}) = \{S \subseteq \hat{V} \mid \forall X \in \mathsf{Teams}(S), \forall i \in X, \mathsf{Val}(S) \lesssim_i \vec{u}\}$ where $\mathsf{Teams}(S) = \mathsf{Teams}(\hat{v})$ for any $\hat{v} \in S$ and $\mathsf{Val}(S)$ is defined if and only if $\mathsf{Teams}(\hat{v}) = \mathsf{Teams}(\hat{v}')$ for any $\hat{v}, \hat{v}' \in S$. In Proposition 2, given $\vec{u} \in \{0,1\}^N$, $F(\vec{u}) = \{\hat{v} \in \hat{V} \mid \forall X \in \mathsf{Teams}(\hat{v}), \forall i \in X, \mathsf{Val}(\hat{v}) \lesssim_i \vec{u}\}$. 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.

For two strategy profiles $\vec{\mathsf{S}}_1$ and $\vec{\mathsf{S}}_2$, we say $\vec{\mathsf{S}}_1$ 'Pareto dominates' $\vec{\mathsf{S}}_2$ if for all agents $i \in \Pi$, it is $\mathbf{u}(q^{(0)}, \vec{\mathsf{S}}_1) \gtrsim_i \mathbf{u}(q^{(0)}, \vec{\mathsf{S}}_2)$ and for at least one agent $k \in \Pi$, we have $\mathbf{u}(q^{(0)}, \vec{\mathsf{S}}_1) >_k \mathbf{u}(q^{(0)}, \vec{\mathsf{S}}_2)$. A strategy profile $\vec{\mathsf{S}}$ 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 $\vec{u}$, the number of payoff vectors that the agent prefers over $\vec{u}$. For instance, if $\vec{u}$ is the worst payoff vector for agent i, then $\mu_i(\vec{u}\,) = 0$; if the agent's utility for some payoff vector $\vec{u}\,'$ is, say ℓ, and $\vec{u} >_i \vec{u}\,'$ while no other $\vec{u}\,'' \neq \vec{u}$ exists with $\vec{u}\,'' >_i \vec{u}\,'$, then $\mu_i(\vec{u}\,) = \ell + 1$.

We thus get an ordering of equilibrium strategies depending on their associated utilities. Consider utility vectors in the form $\vec{\mu} = (\mu_1, \ldots, \mu_N)$, 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 that payoff will not be realised. In our negotiation protocol, the agent who initiated the successful proposal on that particular utility vector arbitrarily picks their policy.

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 $\vec{\mu}_{\mathsf{dis}}$ – 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 $\vec{\mu}$ it prefers in the sense that $\vec{\mu}[i] = \max_{\vec{u}\in\mathcal{U}} \vec{\mu}_i(\vec{u}\,) \geq \vec{\mu}_{\mathsf{dis}}[i]$. 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 non-empty – the proposing agent is always

doi: 10.1049/iet-cta.2014.0611

Algorithm 1

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

$proposer \leftarrow$ `Random`(Π); $done \leftarrow$ 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**

$done \leftarrow$ True;

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

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

if $accept =$ False **then**

$proposer \leftarrow j$, $prop \leftarrow newprop$, $done \leftarrow$ False

if $accept =$ True **then**

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

$prop \leftarrow newprop$; $proposer \leftarrow j$; $done \leftarrow$ False.

if $done =$ True **then**

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

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

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

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

Fig. 2 *Algorithm 1: negotiation*

Algorithm 2

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

$Y \leftarrow \emptyset$; $accept =$ 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**

$X \leftarrow X \cup \{j\}$; $accept =$ 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**

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

return X, $prop$, $accept$, k;

else

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

return $X, newprop, accept, k$;

Fig. 3 *Algorithm 2: AcceptOrChoose*

included. Before agent j rejects a proposal $\vec{\mu}$, it must look at who has already accepted it and compute the set of alternative proposals $\vec{\mu}\,'$ that will be accepted by all those agents. If the set is non-empty, j selects the best for itself improved proposal from there. If the set of alternative proposals is empty, j compares the current proposal $\vec{\mu}$ to the disagreement utility vector $\vec{\mu}_{\text{dis}}$. If $\vec{\mu}[j] > \vec{\mu}_{\text{dis}}[j]$, then j must compromise and accept $\vec{\mu}$. If $\vec{\mu}[j] \leq \vec{\mu}_{\text{dis}}[j]$, j has really nothing to lose by outright rejecting and suggesting a new proposal that serves its best interest, one that satisfies $\vec{\mu}\,'[j] = \max_{\vec{u}\in\mathcal{U}} \mu_j(\vec{u}\,)$. A new round of negotiation starts. Algorithm 1 (see Fig. 2) describes this negotiation protocol.

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

Lemma 2: 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 the disagreement utility $\vec{\mu}_{\text{dis}}$ is chosen such that $\vec{\mu}_{\text{dis}}[i] < \vec{\mu}\,[i]$ $\forall \vec{\mu} \in \mathcal{M}$ and $i \in \Pi$, then Algorithm 1 (see Fig. 2) ensures agreement on a utility vector which corresponds to a (set of) Pareto optimal equilibria.

Proof: Let the utility vector in the current proposal be $\vec{\mu}$. If it is not Pareto optimal, there is another utility vector $\vec{\mu}\,'$ such that for at least some agent i, $\vec{\mu}\,'[i] > \vec{\mu}\,[i]$ while for any other $k \in \Pi \setminus \{i\}$ it is $\vec{\mu}\,'[k] \geq \vec{\mu}[k]$. When agent i gets its turn, it will propose $\vec{\mu}\,'$ as an improvement to $\vec{\mu}$ – based on Algorithm 2 (see Fig. 3), agents will either accept as is, or improve, any utility vector other than $\vec{\mu}_{\text{dis}}$. Thus, the cardinality of the set of agents who have accepted the current proposal is monotonically increasing. The disagreement utility $\vec{\mu}_{\text{dis}}$, 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. □

5 Examples

In this section we illustrate the method with two robotic motion planning examples. The first example is the one mentioned in Section 1. The second example studies a similar scenario with more than three agents and we discuss potential problems and solutions for extending the equilibrium analysis for games with large numbers of agents.

5.1 Case study I

Recall the example in 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. 4. A transition of the form for example $ABC \xrightarrow{a,c,\epsilon} BDC$ 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

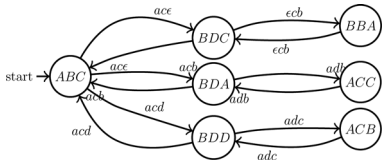

Fig. 4 *Fragment of the multi-agent arena $P = (Q, \mathcal{ACT}, T, q^{(0)})$*
A state (i,j,k) is represented as ijk – agent 1 is in room i, agent 2 in room j and agent 3 in room k. The initial state $q^{(0)} = (A, B, C)$. For all $(a_i)_\Pi \in \mathcal{ACT}$, and $i \neq j \in \Pi$, if $a_i, a_j \neq \epsilon$, 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.

agents will then arrive at rooms B, D and C. Fig. 5 shows a fragment of the two-player game arena H, obtained from P.

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:

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

For each payoff vector $\vec{u}$, computing a winning strategy in the two-player game $\mathcal{H}(\vec{u})$ takes an average of 38 s, with a Python implementation on a desktop with Intel(R) Core(TM) *i*5 processor and 16 GB RAM.

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

5.1.1 Equilibrium analysis:

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

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

In this case, the two-player game $\mathcal{H}$, has payoff vector set $\mathcal{PV} = \{0,1\}^3$. Analysing 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 $\mathsf{Teams} = \{\{1\},\{2\},\{1,2\},\{3\}\}$; 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 realise 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 radicalise agents 1 and 2: they now prefer failure to letting agent 3 get his way. For agent 1 we have

$$(0,0,1) \lesssim_1 (0,1,1) \lesssim_1 (1,0,1) \lesssim_1 (1,1,1) \lesssim_1 (0,0,0)$$
$$\lesssim_1 (0,1,0) \lesssim_1 (1,0,0) \lesssim_1 (1,1,0) \quad (4)$$

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

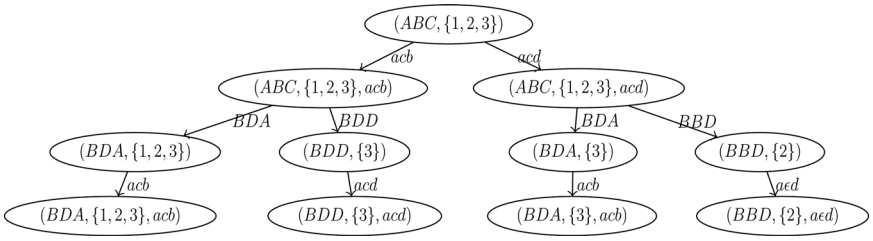

Fig. 5 *Fragment of the two-player turn-based game arena H*
The semantics of a state in V_I [e.g. $(ABC, \{1,2,3\}, acb)$] is that agents are in the rooms marked by the first component (i.e. 1 in A, 2 in B and 3 in C), the agents suspect for triggering the transition there are the ones in the second 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.

doi: 10.1049/iet-cta.2014.0611

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	✓	✗	✓	✓	✓	✗	✓	✓

case 3 wins, my preferences coincide with the case where he loses.' Similarly for agent 2

$$(0,0,1) \lesssim_{\mathcal{Z}2} (1,0,1) \lesssim_{\mathcal{Z}2} (0,1,1) \lesssim_{\mathcal{Z}2} (1,1,1) \lesssim_{\mathcal{Z}2} (0,0,0) \lesssim_{\mathcal{Z}2} (1,0,0) \lesssim_{\mathcal{Z}2} (0,1,0) \lesssim_{\mathcal{Z}2} (1,1,0) \quad (5)$$

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.

5.1.2 Agreeing on strategies through negotiation: Negotiation is needed if coordination is to be decentralised. Here we will not consider team deviations, and thus focus on cases 1 and 3.

For case 1, the condition in Lemma 2 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 $(1,0,1)$, whereas for agent 3 it is any of the form $\{(x,y,0) \mid x,y \in \{1,0\}\}$. The condition in Lemma 2 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 towards 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 (see Fig. 3), 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.

5.2 Case study II – more players

Consider four agents aiming at visiting infinitely often different regions in the environment in Fig. 6. The rules in this game are as follows.

It is assumed that agents R1, R2 and R3 can only move east or west, whereas agent R4 can also move north and south, but is confined to regions below the northwest-to-southeast diagonal. None of these agents is allowed to stay in the same region for more than one turn. Only one agent can go through a given region boundary at any time, although they can share a region during a single turn. The temporal logic specifications for the agents are, respectively, $\square\lozenge D$, $\square\lozenge B$, $\square\lozenge C$ and $\square\lozenge(D \wedge \lozenge C)$. Specification $\square\lozenge X$ translates to region X being visited infinitely often; $\square\lozenge(X \wedge \lozenge Y)$ requires that regions X and Y need to be visited infinitely often, and in this order. We assume that all agents are selfish, that is, have selfish preference orderings over all outcomes, and no teaming between agents is allowed. In this case, if there exist equilibria that allow the agents to accomplish their objectives, then the negotiation is guaranteed to yield the utility vector that corresponds to this set of equilibria.

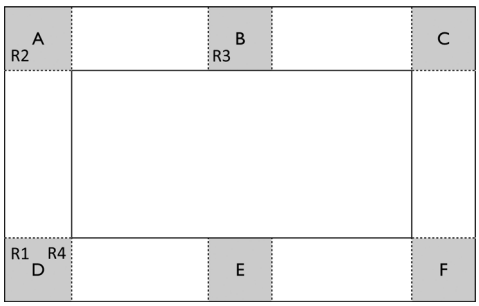

Fig. 6 *Partitioned rectangular environment in which four agents, R1, R2, R3 and R4, roam*

The locations of agents at the start of the game is D, A, B and D. A dashed line marks a region boundary that can be crossed by the agents.

Analysing all payoff vectors in $\mathcal{PV} = \{0,1\}^4$, we find that there are no equilibria to realise the following four payoffs: $(0,0,0,0)$, $(0,0,0,1)$, $(1,0,0,0)$ and $(1,0,0,1)$.

Since the computation of the two-player turn-based game requires the analysis of all possible deviations, from all game states, for all agents, scaling up the number of agents can come at a considerable computational cost.

However, there are reasons to be optimistic that the results here will be able to scale up, at least in certain cases. First, the factorisation of the games that our constructions provide (via the concurrent and synchronised products) structure the two-player game in a certain way. In certain class of situations, this structure may allow the exhaustive analysis to be computed efficiently directly from the factors (the agents, their objectives etc.).

Second, if no teaming-up is allowed, we can break-up the analysis on an agent-by-agent basis. By the definition of equilibrium, if no winning strategy exists for player 1 in the resulting two-player turn-based game, then no equilibrium is associated with the given payoff vector. Consequently, solving the two-player turn-based game where exactly one agent deviates is relatively cheap, computationally. Meanwhile, since concurrent interaction only occurs when the dynamics of agents are tightly coupled, for systems with a large number of states and agents, one can possibly reduce

the equilibrium analysis for the overall game into a fragment of the game that involves the dynamically coupled agents only.

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 behaviours, 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 decentralised way, agents need a way of selecting a single, common equilibrium to implement; mixing individual behaviours 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 behaviour 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 cannot overcome the challenge posed by the curse of dimensionality. However, the factorisation of the games that our constructions provide may allow for the curse to be overcome when the factorisation reveals that the games are structured in a way that can be exploited efficiently. These and other measures for curbing the increase in the dimension of the product systems – such as taking advantage of the fact that agent coupling may occur over subsets of their operational space – are currently being investigated.

7 Acknowledgment

The authors were supported by the National Science Foundation under award #1035577.

8 References

1 Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., Pappas, G.: 'Symbolic planning and control of robot motion', *IEEE Robot. Autom. Mag.*, 2007, **14**, (1), pp. 61–70
2 Mukund, M.: 'From global specifications to distributed implementations' (Kluwer Academic Publishers, 2002), pp. 19–34
3 Seboui, M.-A.B., Hadj-Alouane, N.B., Delaval, G., Rutten, E., Yeddes, M.: 'A decentralized supervisory control approach for distributed adaptive systems'. Proc. of the Fourth Int. Conf. on Verification and Evaluation of Computer and Communication Systems, British Computer Society, 2010, pp. 13–23
4 Yoo, T.-S., Lafortune, S.: 'A general architecture for decentralized supervisory control of discrete-event systems', *Discrete Event Dyn. Syst.*, 2002, **12**, (3), pp. 335–377
5 Chen, Y., Ding, X.C., Belta, C.: 'Synthesis of distributed control and communication schemes synthesis of distributed control and communication schemes from global LTL specifications'. IEEE Conf. on Decision and Control, Orlando, FL, 2011, pp. 2718–2723
6 Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C.: 'Robust multi-robot optimal path planning with temporal logic constraints'. IEEE Int. Conf. on Robotics and Automation, May 2012, pp. 4693–4698
7 Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: 'Temporal-logic-based reactive mission and motion planning'. *IEEE Trans. Robot.*, 2009 , **25**, (6), pp. 1370–1381
8 Tabuada, P.: 'Approximate simulation relations and finite abstractions of quantized control systems', in Bemporad, A., Bicchi, A., Buttazzo, G. (Eds.): 'Hybrid systems: computation and control, (*LNCS*, **4416**) (Springer-Verlag, 2007), pp. 529–542
9 Tanner, H., Fu, J., Rawal, C., Piovesan, J., Abdallah, C.: 'Finite abstractions for hybrid systems with stable continuous dynamics', *Discrete Event Dyn. Syst.*, 2012, **22**, pp. 83–99
10 Apt, K.R., Grädel, E.: 'Lectures in game theory for computer scientists' (Cambridge University Press, 2011)
11 Fisman, D., Kupferman, O., Lustig, Y.: 'Rational synthesis', in Esparza, J., Majumdar, R. (Eds.): 'Tools and algorithms for the construction and analysis of systems' (Cyprus, Springer Berlin Heidelberg, 2010), pp. 190–204
12 Bouyer, P., Brenguier, R., Markey, N., Ummels, M.: 'Concurrent games with ordered objectives', in Birkedal, L. (Ed.): 'Foundations of software science and computational structures' (*LNCS*, **7213**), (Springer, Berlin, Heidelberg, 2012), pp. 301–315
13 Murray, C., Gordon, G.: '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 Perrin, D., Pin, J.É.: 'Infinite words: automata, semigroups, logic and games' (Elsevier, 2004)
15 Thomas, W.: 'Infinite games and verification'. Proc. 14th Int. Conf. on Computer Aided Verification (Springer-Verlag, London, UK, 2002), pp. 58–64
16 Neider, D., Rabinovich, R., Zimmermann, M.: 'Down the Borel hierarchy: solving muller games via safety games', *Theor. Comput. Sci.*, 2014, **560**, (3,4), pp. 219–234
17 Alur, R., La Torre, S.:'Deterministic generators and games for ltl fragments'. 2001 Proc. 16th Annual IEEE Symp. on Logic in Computer Science, 2001, pp. 291–300
18 Piterman, N., Pnueli, A., Sa'ar, Y.: 'Synthesis of reactive (1) designs', in Allen Emerson, E., Namjoshi, Kedar S., (Eds.): 'Verification, model checking, and abstract interpretation' (Springer, 2006), pp. 364–380
19 Bouyer, P., Brenguier, R., Markey, N., Ummels, M.: 'Nash equilibria in concurrent games with Büchi objectives'. IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science', 2011, vol. 13, pp. 375–386
20 Reiter, R.: 'Knowledge in action: logical foundations for specifying and implementing dynamical systems' (MIT Press, 2001)
21 Shoham, Y., Leyton-Brown, K.: 'Multiagent systems – algorithmic, game-theoretic, and logical foundations' (Cambridge University Press, 2009)
22 Arnold, A.: 'Synchronized products of transition systems and their analysis', in Desel, J., Silva, M. (Eds.): 'Application and theory of Petri nets' (*LNCS*, **1420**) (Springer, Berlin, Heidelberg, 1998), pp. 26–27

doi: 10.1049/iet-cta.2014.0611