# On-line Coordination Among Discrete-Event Agents

# (Extended Abstract)

Manh Tung Pham and Kiam Tian Seow
Division of Computing Systems, School of Computer Engineering
Nanyang Technological University
Republic of Singapore 639798
pham0028@ntu.edu.sg, asktseow@ntu.edu.sg

## ABSTRACT

This paper addresses a novel coordination problem for distributed agents in a discrete-event setting. We introduce and study a predicate coordination problem as the problem of distributed agents interacting and communicating between themselves to satisfy (the invariance of) a global predicate specifying an inter-agent constraint. We then develop an optimal coordination policy by which the agents can coordinate to satisfy the predicate constraint. To implement the optimal policy, we develop two on-line coordination strategies including one that can achieve significant savings in communication bandwidth.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents, Multiagent Systems

## General Terms

Algorithms, Design, Theory

## Keywords

Multiagent Coordination, Discrete-Event Modeling

## 1. BACKGROUND & NOTATION

We use the following notations from language and automata theory [2]: For an automaton $A = \left(X^A, \Sigma^A, \delta^A, x_0^A\right)$, $(\Sigma^A)^*$ denotes the set of all finite strings over $\Sigma^A$, and $L(A)$ denotes the (prefix-closed) language generated by $A$; for two strings $s$ and $s'$ in $(\Sigma^A)^*$, $s' \leq s$ denotes that $s'$ is a prefix of $s$; for $x \in X^A$, $\sigma \in \Sigma^A$, $\delta^A(\sigma, x)!$ denotes that $\delta^A(\sigma, x)$ is defined, and $\Sigma^A(x)$ denotes the set of events $\sigma \in \Sigma^A$ where $\delta^A(\sigma, x)!$; $Reach(A)$ denotes the reachable automaton obtained from $A$ by deleting all unreachable states of $A$; and $A = A_1 \parallel A_2$ denotes that automaton $A$ is the synchronous product of the two automata $A_1$ and $A_2$.

## 2. PROBLEM FORMULATION

Consider a system of two discrete-event agents modeled by the respective automata $A_i = (X^{A_i}, \Sigma^{A_i}, \delta^{A_i}, x_0^{A_i})$ $(i \in \{1, 2\})$, where $\Sigma^{A_1} \cap \Sigma^{A_2} = \emptyset$. The event set $\Sigma^{A_i}$ of agent $A_i$ is partitioned into the controllable set $\Sigma_c^{A_i}$ and the uncontrollable set $\Sigma_{uc}^{A_i}$.

**Cite as:** On-line Coordination Among Discrete-Event Agents, (Extended Abstract), Manh Tung Pham, Kiam Tian Seow, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1223–1224


Let $A = A_1 \parallel A_2$ with $\Sigma_c^A = \Sigma_c^{A_1} \cup \Sigma_c^{A_2}$ and $\Sigma_{uc}^A = \Sigma_{uc}^{A_1} \cup \Sigma_{uc}^{A_2}$. A (safety) inter-agent constraint can then be specified by an automaton $C$ with $\Sigma^C = \Sigma^A$ where $L(C)$ is interpreted as the desired behavior that one wishes to impose on the system $A$ [1]. In this paper, we focus on a $C$ which is a nonempty sub-automaton [2] of $A$. Such an automaton $C$ can be equivalently represented by a predicate defined on the set $X^A$. In essence, a predicate $P$ defined on $X^A$ is a function $P : X^A \to \{0, 1\}$. For automaton $C$, the equivalent predicate $P_c$ is defined as follows:

$$(\forall x \in X^A) P_c(x) = 1 \text{ iff } x \in X^C.$$

For a state $x \in X^A$, we say $x$ satisfies $P_c$, and write $x \models P_c$, if $P_c(x) = 1$. For two predicates $P_1$ and $P_2$ defined on $X^A$, we say that $P_1$ is not less restrictive than $P_2$, denoted by $P_1 \preceq P_2$, if $(\forall x \in X^A)(x \models P_1 \Rightarrow x \models P_2)$.

The coordination problem now becomes that of $A_1$ and $A_2$ appropriately exchanging information and enabling their events so that none of the states in the set $X^A - X^C$ is ever visited during their execution. The coordinating actions for a pair of agents are governed by a coordination policy, formally defined as follows.

Definition 1. *A coordination policy $\pi_{<A_1,A_2>}$ is a pair of agent policies $< \pi_{A_1}, \pi_{A_2} >$, where $\pi_{A_i}$ for agent $A_i$ is a mapping from a state $x \in X^A$ to an event subset of $\Sigma^{A_i}$, such that $(\forall x \in x^A)\pi_{A_i}(x) \supseteq \Sigma_{uc}^{A_i} \cap \Sigma^A(x)$.*

Using coordination policy $\pi_{<A_1,A_2>}$, agent $A_i$, upon observing the system state $x \in X^A$, enables every event $\sigma \in \pi_{A_i}(x)$, and disables all other events. The condition $(\forall x \in x^A)\pi_{A_i}(x) \supseteq \Sigma_{uc}^{A_i} \cap \Sigma^A(x)$ characterizes the fact that uncontrollable events can never be disabled.

Definition 2. *The system of agents $A_1$ and $A_2$ interacting using a coordination policy $\pi_{<A_1,A_2>}$ is a system represented by an automaton $A_\pi = Reach(X^A, \Sigma^A, \delta_\pi^A, x_0^A)$, where $(\forall \sigma \in \Sigma^{A_i})(\forall x \in X^A)$ $\delta_\pi^A(\sigma, x) = \delta^A(\sigma, x)$ if $\delta^A(\sigma, x)!$ and $\sigma \in \pi_{A_i}(x)$, and is undefined otherwise.*

Definition 3. *A predicate $P$ defined on $X^A$ is said to be coordinable if, for every $x \in X^A$ satisfying $P$, the following conditions are satisfied: (1) $(\exists s \in (\Sigma^A)^*)[\delta^A(s, x_0^A) = x$ and $(\forall w \leq s)\delta^A(w, x_0^A) \models P]$, and (2) $(\forall \sigma \in \Sigma_{uc}^A)[\delta^A(\sigma, x)! \Rightarrow \delta^A(\sigma, x) \models P]$.*

Theorem 1. *Let $C$ be a nonempty sub-automaton of $A$. Then there exists a coordination policy $\pi_{<A_1,A_2>}$ such that $A_\pi = C$ iff $P_c$ is coordinable.*

It can be shown that the set of coordinable predicates that are not less restrictive than $P_c$ is nonempty and closed under arbitrary predicate disjunctions, and so its supremal element, denoted by $P_c^{sup}$, exists. Let $C^{sup}$ denote the equivalent automaton of $P_c^{sup}$. The predicate coordination problem can now be formally stated as follows.

PROBLEM 1. *Given a predicate constraint $P_c$ defined on the system state space $X^A$, construct the (unique) optimal coordination policy $\pi_{<A_1,A_2>}$ such that $A_\pi = C^{sup}$.*

DEFINITION 4. *A state $x \in X^A$ is said to be $P_c$-safe if $(\forall s \in (\Sigma^A_{uc})^*)[\delta^A(s,x)! \Rightarrow \delta^A(s,x) \models P_c]$.*

THEOREM 2. *Let $C$ be a nonempty sub-automaton of $A$. Assume that $x_0^A$ is $P_c$-safe. Let $\pi_{<A_1,A_2>}$ be a coordination policy with $\pi_{A_i}$ given as: $(\forall x \in X^A)(\forall \sigma \in \Sigma^{A_i})[\sigma \in \pi_{A_i}(x)$ iff $\delta^A(\sigma,x)!$ and $\delta^A(\sigma,x)$ is $P_c$-safe$]$. Then $A_\pi = C^{sup}$ (i.e., $\pi_{<A_1,A_2>}$ is the optimal solution policy of Problem 1).*

By Theorem 2, a procedure called *ComputeEnabledEventSet* can be used by agent $A_i$ ($i \in \{1,2\}$) to determine the set of events to enable next each time it observes a new system state $x$ as follows: $A_i$ begins by initializing $\pi_{A_i}(x)$ with the set of all uncontrollable events in $\Sigma^{A_i}_{uc} \cap \Sigma^A(x)$. Next, it simply checks if $\delta^A(\sigma,x)$ is $P_c$-safe for each controllable event $\sigma \in \Sigma^{A_i}_c \cap \Sigma^A(x)$, and adds $\sigma$ to the set $\pi_{A_i}(x)$ of enabled events if the check returns a positive result.

# 3. ON-LINE COORDINATION

We now present two on-line strategies to implement the optimal coordination policy given in Theorem 2.

## 3.1 With Full Communication

The first on-line strategy is called OnlineCoAgent-ComFull, and follows directly from Theorem 2. Using the strategy, the agents start by exchanging their initial states; and upon entering a new state, an agent would immediately send its updated local state to the other agent. Each time the agents have individually updated the system state, they would apply procedure *ComputeEnabledEventSet* to determine their next set of events to enable.

## 3.2 With Reduced Communication

The second coordination strategy attempts to reduce communication bandwidth. It uses the concept of an agent's (coordination) view to implement the optimal policy given in Theorem 2. The local view of agent $A_1$ is represented by the tuple $(x_1, x_2^{r_1}, x_1^{s_2})$, where $x_1$ is its current state, $x_2^{r_1}$ is $A_1$'s view of $A_2$'s current state and is the most recent state information $A_1$ received from agent $A_2$, and $x_1^{s_2}$ is the most recent state information that $A_1$ sent to $A_2$. The local view of agent $A_2$ is similarly represented by the tuple $(x_2, x_1^{r_2}, x_2^{s_1})$. Note that since inter-agent communication is assumed instantaneous, $x_1^{s_2} = x_1^{r_2}$ and $x_2^{s_1} = x_2^{r_1}$.

DEFINITION 5. *Given $x_1 \in X^{A_1}$, two states $x_2, x_2' \in X^{A_2}$ are said to be equivalent with respect to $x_1$ (on $P_c$), and denoted by $x_2 \equiv_{x_1} x_2'$ (mod $P_c$), if $(\forall \sigma \in \Sigma^{A_1}(x_1) \cap \Sigma^{A_1}_c)$ $[(\delta^A(\sigma,x_1),x_2)$ is $P_c$-safe iff $(\delta^A(\sigma,x_1),x_2')$ is $P_c$-safe$]$. The notion $x_1 \equiv_{x_2} x_1'$ for $x_1, x_1' \in X^{A_1}$ and $x_2 \in X^{A_2}$ is defined similarly.*

For economy of notation, we will often omit 'mod $P_c$' and simply write $x_i \equiv_{x_j} x_i'$ in place of $x_i \equiv_{x_j} x_i'$ (mod $P_c$).

DEFINITION 6. *For two states $x_1, x_1' \in X^{A_1}$, $\equiv_{x_1}$ is said to be finer than $\equiv_{x_1'}$, and denoted by $\equiv_{x_1} \preceq \equiv_{x_1'}$, if $(\forall x_2, x_2' \in X^{A_2})(x_2 \equiv_{x_1} x_2') \Rightarrow (x_2 \equiv_{x_1'} x_2')$. The notion $\equiv_{x_2} \preceq \equiv_{x_2'}$ for $x_2, x_2' \in X^{A_2}$ is defined similarly.*

DEFINITION 7. *Two agents $A_1$ and $A_2$, with their respective local views $(x_1, x_2^{r_1}, x_1^{s_2})$ and $(x_2, x_1^{r_2}, x_2^{s_1})$, are said to be coordination-ready (for $P_c$) if $x_1^{s_2} \equiv_{x_2} x_1$ and $x_2^{s_1} \equiv_{x_1} x_2$.*

Thus, to implement the optimal solution policy $\pi_{<A_1,A_2>}$ for which $A_\pi = C^{sup}$, the agents, following every event execution, would need to re-establish coordination-readiness prior to determining their next set of enabled events. Note that always re-establishing state synchronization as with OnlineCoAgent-ComFull is the most conservative way that trivially re-establishes coordination-readiness. Checking for coordination-readiness first, with $x_i^{s_j} \equiv_{x_j} x_i$ by agent $A_i$, might reduce $A_i$ to communicating its current local state to the other agent $A_j$ only when the check fails. However, such direct checking clearly requires agent $A_i$ to also know the current local state $x_j$ of agent $A_j$, which is not always possible. This necessitates a stronger notion called co-stability, whose conditions can be mutually checked by the agents.

OnlineCoAgent-ComReduce($A_1$)

**begin**

Communicate the initial state $x_0^{A_1}$ to $A_2$;

Upon receiving local state $x_2$ from $A_2$

Update the view of $A_2$'s state: $x_2^{r_1} \leftarrow x_2$;
**if** $x_1^{s_2} \not\equiv_{x_2^{r_1}} x_1$ **then**
  Communicate $x_1$ to $A_2$; Update $x_1^{s_2} \leftarrow x_1$;
Apply $ComputeEnabledEventSet((x_1, x_2^{r_1}))$ to determine the next set of enabled events;

Upon executing event $\sigma \in \Sigma^{A_1}$

Update current state: $x_1 \leftarrow \delta^{A_1}(\sigma, x_1)$;
**if** $\equiv_{x_1^{s_2}} \not\preceq \equiv_{x_1}$ *or* $x_1^{s_2} \not\equiv_{x_2^{r_1}} x_1$ **then**
  Communicate $x_1$ to $A_2$; Update $x_1^{s_2} \leftarrow x_1$;
Apply $ComputeEnabledEventSet((x_1, x_2^{r_1}))$ to determine the next set of events to enable;

**end**

**Figure 1:** OnlineCoAgent-ComReduce($A_i$). For definiteness of description, the strategy instance for $A_1$ is shown; that for $A_2$ is the same except that its reciprocal agent is $A_1$.

DEFINITION 8. *Two agents $A_1$ and $A_2$ with their respective local views $(x_1, x_2^{r_1}, x_1^{s_2})$ and $(x_2, x_1^{r_2}, x_2^{s_1})$, are said to be co-stable (for $P_c$) if (1) $x_1^{s_2} \equiv_{x_2^{r_1}} x_1$, (2) $x_2^{s_1} \equiv_{x_1^{r_2}} x_2$, (3) $\equiv_{x_1^{s_2}} \preceq \equiv_{x_1}$, and (4) $\equiv_{x_2^{s_1}} \preceq \equiv_{x_2}$.*

PROPOSITION 1. *Whenever agents $A_1$ and $A_2$ are co-stable (for $P_c$), they are coordination-ready (for $P_c$).*

Importantly, the co-stability conditions could be mutually checked by the two agents $A_1$ and $A_2$: Conditions (1) and (3), which only require information access to $x_1$, $x_2^{r_1}$ and $x_1^{s_2}$, can be checked by agent $A_1$; Similarly, Conditions (2) and (4) can be checked by agent $A_2$.

Thus, to re-establish coordination-readiness following an event execution, the agents can check the co-stability conditions, and when necessary, interact by communicating their local state to re-establish co-stability using a new strategy called OnlineCoAgent-ComReduce (Fig. 1). Importantly, unlike OnlineCoAgent-ComFull, OnlineCoAgent-ComReduce does not require two coordinating agents always updating each other with their current local state, and hence can potentially reduce inter-agent communication. Experimental results, described in detail elsewhere, show that compared to OnlineCoAgent-ComFull, OnlineCoAgent-ComReduce can achieve significant bandwidth reduction.

# 4. CONCLUSIONS

This paper has presented new coordination results formalizing how discrete-event agents can interact and communicate in an on-line fashion to guarantee the invariance of a predicate specifying an inter-agent constraint.

# 5. REFERENCES

[1] K. T. Seow, C. Ma, and M. Yokoo. Multiagent planning as control synthesis. In *AAMAS'04*, 972–979, New York, 2004.
[2] C. G. Cassandras and S. Lafortune. Introduction to Discrete Event Systems. Springer, 2008.