Bachelor Thesis

How does a token based order compare to the asynchron order in multi-agent plan executions?

Felicitas Ritter

Examiner: Prof. Dr. B. Nebel

Advisers: Dr. Robert Mattmüller,
Thorsten Engesser

Albert-Ludwigs-University Freiburg

Faculty of Engineering

Department of Computer Science

Chair for Foundations of Artificial Intelligence

May 07th, 2019

Writing Period

07. 05. 2019 – 07. 08. 2019

Examiner

Prof. Dr. B. Nebel

Advisers

Dr. Robert Mattmüller, Thorsten Engesser

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature

Abstract

foo bar

Zusammenfassung

German version is only needed for an undergraduate thesis.

Contents

1 Introduction

How would a perfect execution order look?

(TODO: What is an execution order?) (TODO: Who are the agents?)

This thesis is trying to answer that question by looking at the different execution orders. In other pieces of writing **(TODO: which?)** about epistemic planning the execution order is ofthen unspecified, called an asynchronous execution order. This is easier to describe mathematically, but it also has some drawbacks. We are now going to have a look at the different execution orders, especially at the token based execution order.

There are four different execution modes that are possible:

1. Asynchronous

   The agents move in a seemingly random order.

2. Concurrent

   The agents can act at the same time.

3. token based

   The agents need a special token to be able to act, only one agent has the token at a time.

4. round robin

   The agents act one agent after the other, after the last agent is done they begin with the first agent again.

We have found that the token based orders are a subset of the asynchronous orders, because all token based orders can also be “a random” asynchronous order, but not all asynchronous orders are a token based order. Also all round robin orders can be token based orders by accident, but not all token based orders are a round robin order. Because of this we only looked at token based orders in this thesis.

The field of epistemic dynamic logic is relatively young. It started with the work from Plaza (1989) [1] about “Logics of public communications”. Epistemic planning is taking regular planning task and enriching it with knowledge and belief. In regular planning, the world is simplified, so that everyone knows everything there is to know about that particular microverse. But this is only a very simplified model of the real world. In the real world, agents often have only little knowledge and can therefore make only limited calculations and beliefs on that world. To model that, we use epistemic logic. To show that the agents each have different knowledge that can change over time, we describe the logic as being dynamic.

There are multiple ways this research could be used in the future, in the real world. One example is in autonomous driving. Two cars are standing across from each other on a bridge, only one car can pass. This problem could be solved with tokens.

“Many applications, such as robotic warehouses, must coordinate a large number of agents somultaneausly to carry out theis specific tasks.” This problem is usually described as the multi-agent path finding problem.

The thought of researching if a different order solves some problems is also done in multi agent path finding problems. Here, the problem gets more complex the more agents there are and it is very similar to the del approach.

(EXTEND: This work is structured as follows:)

2 Related Work

Give a brief overview of the work relevant for your thesis.

This thesis is building up from the work of Engesesser et al (2018) [2]. They investigated how a lazy agent, who had a preference against doing its own actions, compares to eager agents in planning problems. **(EXTEND: What did they find out?) (EXTEND: How is this different than my work? How does my work build up from their work?)**

DEl Papiere, Baltag und Moss implicit coordination

distributed systems, verteilte systeme $->$ vor allem Tokens

Tokens have also made an appearance in distributed systems **(TODO: cite)**. A distributed systems is a system where the components of the system are located apart from each other but they still have to communicate and coordinate their actions to archieve a common goal. This makes that field face some similar problems like the coordination of actions, the concurrency of the agents and the intransparency. The field also needs scalable solutions.

In this field **(TODO: specify the field again)**, there are a few load balancing algorithms that try to equally distribute a task amongst several agents. Ray et al. (2012) [3] analysed the different existing load balancing algorithms like token routing, round robin, randomized, Central queuing and Connection mechanism.
Another use case in this field is the restictional use of a mutual resource that can

only have a small number of users at a time. This can be archieved by using a token queue and a token semaphore, as from Makki et al. (1992) [4]

3 Background

Imagine the following game between two players, Anne and Bill. It involes a lever placed in between them. The lever is upright in the middle position and has five positions in total. Anne thinks the lever should be pulled two positions to her side and Bill thinks the lever should be pulled two positions to his side. This can be seen in figure 1.

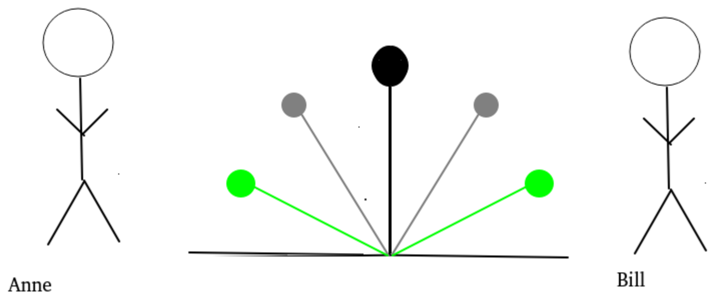


Figure 1: Visualization of the example

With the asynchronous execution order this game could be easily finished if Anne or Bill just pulled the lever two times in their direction. But just as easily this game could go on a really long time, if Anne and Bill pulled the lever alternating, once to the right and then to the left, then to the right again and so on. This could go on infinitely. With no set execution order, how can we prevent the game from going on infinitely?

3.1 Dynamic Epistemic Logic

In the following we will define and explain the core concepts of the Dynamic Epistemic Logic (DEL). DEL is a specific mathematical language used as the framework of this thesis.

The definitions are taken from the "Better eager than lazy" (2018) [2] paper, the "A gentle introduction to DEL" (2017) [5] and the book "Dynamic epistemic logic" from Ditmarsch [6].

Let $\mathcal{A}$ be a finite set of agents, from the example above this would be $\mathcal{A} = \{\text{Anne}, \text{Bill}\}$. Let $\mathcal{P}$ be a finite set of atomic propositions. Atomic propositions, like p or q, describe some affairs that can be true or false. The epistemic language $\mathcal{L}_{\text{KC}}$ is:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid C\varphi$$

with $p \in \mathcal{P}$ and $i \in \mathcal{A}$. $\top$ describes φ as being true, $\bot$ as false. $K_i\varphi$ reads as "Agent i knows φ". $C\varphi$ reads as "it is common knowledge that φ ". From the example before, the two agents, a (Anne) and b (Bill) have two goal positions, goal p (lever to the left) and q (lever to the right). Anne knows that one goal position is the left, but does not know the other: $K_a p \wedge \neg K_a q$. Bill knows that one goal position is the right, but does not know the other position: $\neg K_b p \wedge K_b q$.

Formulas are evaluated in epistemic models **(TODO: not defined)**

$$\mathcal{M} = (W, (\sim_i)_{i\in\mathcal{A}}, V)$$

with the domain W being a nonempty finite set of worlds, $\sim_i$ being an equivalence relation called the indistinguishably relation for agent $i \in \mathcal{A}$ and $V : P \to \mathcal{P}(W)$ assigning a valuation to each atomic proposition.

In the example above, Anne sees two worlds. One world where just the lever to the left is a goal, but also a world where another goal is the lever to the right. Those two worlds are indistinguishable for Anne. Let w_1 be the world where just the lever to the left is a goal and w_2 be the world where the goal is the lever to the left or to the right, then $w_1 \sim_a w_2$.

For $W_d \subseteq W$, the pair $(\mathcal{M}, W_d)$ is called an epistemic state (or simply a state) and the worlds of W_d are called designated worlds. A state is called global if $W_d = \{w\}$ for some world w (called the actual world). We then often write $(\mathcal{M}, w)$ instead of $(\mathcal{M}, \{w\})$. We use $S^{gl}(P, \mathcal{A})$ to denote the set of global states (or simply S^{gl} if P and $\mathcal{A}$ are clear from context). For any state $s = (\mathcal{M}, W_d)$ we let $Globals(s) = \{(\mathcal{M}, w) | w \in W_d\}$. A state $(\mathcal{M}, W_d)$ is called a local state for agent i if W_d is closed under $\sim_i$ (that is, if $w \in W_d$ and $w \sim_i v$, then $v \in W_d$). Given a state $s = (\mathcal{M}, W_d)$ the associated local state of agent i, denoted s^i, is $(\mathcal{M}, \{v | v \sim_i w \text{ and } w \in W_d\})$. Going from s to s^i amounts to a *perspective shift* to the local perspective of agent i.

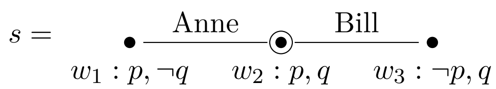
Let $(\mathcal{M}, W_d)$ be a state on $P, \mathcal{A}$ with $\mathcal{M} = (W, (\sim_i)_{i \in \mathcal{A}}, V)$. For $i \in \mathcal{A}$, $p \in P$ and

$\varphi, \psi \in \mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$, we define truth as follows:

$$\begin{array}{lll}
(\mathcal{M}, W_d) \models \varphi & \text{iff} & (\mathcal{M}, w) \models \varphi \text{ for all } w \in W_d \\
(\mathcal{M}, w) \models p & \text{iff} & w \in V(p) \\
(\mathcal{M}, w) \models \neg\varphi & \text{iff} & (\mathcal{M}, w) \not\models \varphi \\
(\mathcal{M}, w) \models \varphi \wedge \psi & \text{iff} & (\mathcal{M}, w) \models \varphi \text{ and } (\mathcal{M}, w) \models \psi \\
(\mathcal{M}, w) \models K_i\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim_i w \\
(\mathcal{M}, w) \models C\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim^* w \\
(\mathcal{M}, w) \models \top & & \text{always} \\
(\mathcal{M}, w) \models \bot & & \text{never}
\end{array}$$

where $\sim^*$ is the transitive closure of $\bigcup_{i \in \mathcal{A}} \sim_i$.

Using Anne and Bill as an Example, Anne cannot distinguish between p, q and $p, \neg q$. Bill cannot distinguish between p, q and $\neg p, q$. A graphic representation of this would be the global state $s = (\mathcal{M}, w_2)$ with the nodes representing the worlds and the edges representing the indistinguishably relation. The circle around a node represent designated worlds.



3.1.1 Epistemic Actions and Product Updates

An event model is $\mathcal{E} = \langle E, (\sim_i)_{i \in \mathcal{A}}, pre, \mathit{eff} \rangle$ where the domain E is a non-empty finite set of events; $\sim_i \subseteq E \times E$ is an equivalence relation called the indistinguishably relation for agent i; $pre : E \to \mathcal{L}_{KC}$ assigns a precondition to each event; and $\mathit{eff} : E \to \mathcal{L}_{KC}$ assigns a post condition, or effect to each event. For all $e \in E$, $\mathit{eff}(e)$ is a conjunction of literals, that means atomic propositions and their negations, including $\top$ and $\bot$.

For $E_d \subseteq E$, the pair $(\mathcal{E}, E_d)$ is called an epistemic action, or simply action and the events in E_d are called a local action for agent i when E_d is closed under $\sim_i$.

Each event of an action represents a different possible outcome. By using multiple events $e, e' \in E$ that are indistinguishable ($e \sim e'$), it is possible to model only partially observable actions.

If the event model has $E = \{e\}$, we will write $\mathcal{E} = \langle pre(e), \mathit{eff}(e)\rangle$.

The product update is used to specify the next state resulting from performing an action in a state. Let a state $s = (\mathcal{M}, W_d)$ and an action $a = (\mathcal{E}, E_d)$ be given with $\mathcal{M} = \langle W, (\sim_i)_{i\in\mathcal{A}}, V\rangle$ and $\mathcal{E} = \langle E, (\sim_i)_{i\in\mathcal{A}}, pre, \mathit{eff}\rangle$ then the product update of s with a is defined as $s \otimes a = ((W', (\sim_i')_{i\in\mathcal{A}}, W_d'))$ where :

- $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$;
- $\sim_i' = \{((w, e), (w', e')) \in W' \times W' \mid w \sim_i w' \text{ and } e \sim_i e'\}$;
- $V'(p) = \{(w, e) \in W' \mid \mathit{eff}(e) \models p \text{ or } (\mathcal{M}, w \models p \text{ and } \mathit{eff}(e) \not\models \neg p)\}$;
- $W_d' = \{(w, e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$.

$a = (\mathcal{E}, E_d)$ is applicable in $s = (\mathcal{M}, W_d)$ if for all $w \in W_d$ there is an event $e \in E_d$ so that $(\mathcal{M}, w) \models pre(e)$.

3.2 Planning tasks

A planning task $\Pi = \langle s_0, A, \omega, \gamma\rangle$ consists of a global state s_0 called the *initial state*; a finite set of actions A; an owner function $\omega : A \to \mathcal{A}$; and a *goal formula* $\gamma \in \mathcal{L}_{KC}$. We require that each $a \in A$ is local for $\omega(a)$. **(TODO: define local)**

Consider the planning task from the beginning. For simplicity, in this example there is only one player and the lever can only be pulled once. The planning task $\langle s_0, \{a_1\}, \omega, p\rangle$ consists of the initial state $s_0 = \underset{\neg p}{\odot}$ with the lever being in the upright position. The action $a_1 = \underset{e_1 : \langle \top, p\rangle}{\odot}$ has the owner $\omega(a_1) = 1$ (player 1). Everything is fully observable for the agent. The intuitive solution should prescribe the action a_1 to agent 1, pulling the lever to the right.

$$\underset{w_1 : \neg p}{\odot} \otimes \underset{e_1 : \langle \top, p\rangle}{\odot} = \underset{(w_1, e_1) : p}{\odot}$$

A policy π for $\Pi = \langle s_0, A, \omega, \gamma\rangle$ is a partial mapping $\pi : S^{gl} \hookrightarrow \mathcal{P}(A)$ such that:

1. Applicability

   We require actions to be applicable in all states they are assigned to:
   for all $a \in S^{gl}, a \in \pi(s) : a$ is applicable in s.

2. Uniformity

   If the policy π prescribes some action a to agent i in state s and agent i cannot distinguish s from some other state t, then π has to prescribe the same action a for i in t as well:
   for all $s, t \in S^{gl}$ such that $s^{\omega(a)} = t^{\omega(a)}, a \in \pi(s) : a \in \pi(t)$

3. Determinism

   We require π to be unambiguous for all agents in the sense that in each state s where an agent i is supposed to act according to π, π will always prescribe the same action for agent i.

The properties uniformity and applicability together imply knowledge of preconditions, the property that in each state, an agent who is supposed to perform a particular action must also know that the action is applicable in that state.

We also must allow policies to sometimes prescribe multiple actions of different owners to the same state. This is because the set of indistinguishable states can differ between the agents. To characterize the different outcomes of agents acting according to a common policy, we define the notion of policy executions.

An execution of a policy π from a global state s_0 is a maximal (finite or infinite) sequence of alternating global states and actions $(s_0, a_1, s_1, a_2, s_2, ...)$, such that for all $m \leq 0$

1. $a_{m+1} \in \pi(s_m)$ and

2. $s_{m+1} \in Globals(s_m \otimes a_{m+1})$

An execution is called successful for a planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, if it is a finite execution $(s_0, a_1, s_1, ..., a_n, s_n)$ such that $s_n \models \gamma$.

(TODO: Geht das hier etwas schöner?) We now want to restrict our focus to policies that are guaranteed to achieve the goal after a finite number of steps. More formally, all of their executions must be successful. As in nondeterministic planning, such policies are called strong (Cimatti et al. 2003 [7]) **(TODO: Quelle lesen)**.

For a planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, a policy π is called strong if $s_0 \in \mathrm{Dom}(\pi) \cup \{s \in S^{gl} \mid s \models \gamma\}$ and for each $s \in \mathrm{Dom}(\pi)$, any extension of π from s is successful for Π. A planning task is called solvable if a strong policy for Π exists. For $i \in \mathcal{A}$, we call a policy i-strong if it is strong and $Globals(s_0^i) \subseteq \mathrm{Dom}(\pi) \cup \{s \in S^{gl} \mid s \models \gamma\}$.

When a policy is i-strong it means that the policy is strong and defined on all the global states that agent i cannot indistinguish between. It follows directly from the definition that any execution of an i-strong policy from any of those initially indistinguishable states will be successful. So if agent i comes up with an i-strong policy, agent i knows the policy to be successful.

Sometimes the agents cannot coordinate their plans but rather have to come up with plans individually. These plans can differ a lot, the agents could have different reasoning capabilities, have non-uniform knowledge of the initial state and of action outcomes. For this reason we will define a policy profile for a planning task Π to be a family $(\pi_i)_{i \in \mathcal{A}}$ where each π_i is a policy for Π. We assume actions to be instantaneous and executed asynchronously. This leads to the following generalization:

An execution of a policy profile $(\pi_i)_{i \in \mathcal{A}}$ is a maximal (finite or infinite) sequence of alternating global states and actions $(s_0, a_1, s_1, ...)$, such that for all $m \leq 0$,

1. $a_{m+1} \in \pi_i(s_m)$ where $i = \omega(a_{m+1})$

   Note here the source of nondeterminism as a result from the possibility of multiple policies prescribing actions for their respective agents.

2. $s_{m+1} \in Globals(s_m \otimes a_{m+1})$

   Here the source of nondeterminism is from the possibility of nondeterministic action outcomes.

The subjective cost of a policy profile π_i is the length of the sequence of alternating global states and actions.

If all agents have one strong policy in common which all of them follow, then at execution time, the goal is guaranteed to be eventually reached. If, however, each agent acts on its individual strong policy, then the incompatibility of the individual policies may prevent the agents from reaching the goal, even tough each individual policy is strong.

4 The tokenized approach

In the example from the beginning, the problem was that both agents a and b wanted to pull the lever to their own goal which results in infinite executions. This problem can be eliminated by introducing a token. With a token, only the player that has the token gets to execute an action. If the agent is done with their own actions, then they can pass the token on to the next player.

The easiest way to introduce the token is seemingly random. Some undefined agent has the token in the beginning. The advantage is that you do not have to define anything in the beginning, but this is also a disadvantage, since this token is defined, but the agent that has it in the beginning is not. Here, giving the token to the next player would have to be added to the set of actions.

There are three other ways to introduce a token to this game. **(TODO: Du nimmst bisher keine Referenz zum Kapitel davor. Wuerde das bedeuten, dass man das "Token nehmen" und "Token abgeben" als Aktion in den Planungsalgorithmus einfuehren muss. Dann wuerde ich das hier schreiben.)**

1. “Table token” - the token is lying on a table and one of the agents can take the token in the beginning.
   The disadvantage is that maybe no agent would take the token.

2. “give token” - in the specification of the game it is also specified which agent will have the token in the beginning.

The problem with this introduction is that in every new game, this has to be written in the definition of the game. In order for the game to be efficient, it should be given to a player who has found a plan.

3. "random token" - the token is given to a random agent in the beginning. If that agent can not preform any action it can pass the token to a player who can.
One disadvantage would be that an agent who knows nothing and has no action will prevent the game from ever reaching a goal state.

4.1 Infinite executions to finite executions

We are now going to describe a function that takes a regular planning task and tokenize that task. The goal with the tokens is that only one player gets to make a move at a time.

Given a regular planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, the function *tokenize* will transform the regular planning task into a tokenized planning task so that for all $a \in A$: if $a = \langle pre, eff \rangle$, then $tokenize(a) = \langle pre \wedge hasToken^{\omega(a)}, eff \rangle$.
Further we define an action $giveToken^{ij} = \langle hasToken^i, \neg hasToken^i \wedge hasToken^j \rangle$ for all $i, j \in \mathcal{A}$ with $i \neq j$.
Then $A^{Token} = \{tokenize(a) | a \in A\} \cup \{giveToken^{ij} | i, j \in \mathcal{A}, i \neq j\}$.
Moreover: $\omega^{Token}(tokenize(a)) = \omega(a)$ for all $a \in A$, and $\omega^{Token}(giveToken^{ij}) = i$ for all $i, j \in \mathcal{A}$.
$s_0^{Token} = s_0 \cup \{\neg hasToken^i | i \in \mathcal{A} \backslash j\} \cup hasToken^j$
With j depending on the way the token should be introduced to the token:

1. Table Token: The first agent to take the token off the table.

2. give Token: a specified agent to be determined by the game maker

3. random Token: a random agent

Then $\Pi^{\text{Token}} = \langle s_0, A^{\text{Token}}, \omega^{\text{Token}}, \gamma \rangle$.

The action *giveToken* can have different costs in this case. In searching for the optimal plan the search tree can have a smaller width, but therefore will probably have a deeper depth. This could be researched in the future. **(TODO: Wie?)**

(EXTEND: überleitender Satz)

Proposition 1. *Under the condition of optimal plans one can prevent the appearance of infinite executions in solvable games with asynchronous execution order with the introduction of a token based execution order.*

proof sketch. The game is finished when the agent that has the token reaches a goal state. When an agent that has a plan gets handed the token, that player has two possible actions:

1. keep the token and execute an action, which will lower the subjective costs of the remaining policy profile or

2. give the token to the next player. in order to pass on the token the player that has found the plan would have already performed a perspective shift for the next player, in order to check if the next player could also find that plan and calculated the subjective cost of the remaining plan. The player that receives the token will always execute that plan or find a better plan and execute the better plan.

Every agent that will get the token in the plan will decrease the subjective cost of the policy profile, and because every agent decreases the subjective cost, the game is executable in infinite executions. □

4.2 The prevention of deadlocks

A deadlock for a policy profile $(\pi_i)_{i \in \mathcal{A}}$ is a global state such that:

1. s is not a goal state

   Something still needs to be done

2. $s \in \mathrm{Dom}(\pi_i)$ for some $i \in \mathcal{A}$

   Someone wants something to be done

3. $\omega(a) \neq i$ for all $i \in \mathcal{A}$ and $a \in \pi_i(s)$

   Nothing will be done because of incompatible individual policies

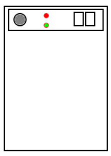


Figure 2: Dishwasher

This can be seen in the following example: The two agents from before, Anne and Bill have to empty out the dishwasher. They each have a preference against doing this, since they both spend time (costs) to do this. In this example you can see that each agent expects the other agent to act, therefore no agent will act.

The formalization of this example would be: Consider the following planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$ with $s_0 = fullDishwasher$, $\mathcal{A} = \{1, 2\}$,
$A = \{emptyDishwasher_1, emptyDishwasher_2\}$ where
$emptyDishwasher_1 = \langle fullDishwasher, \neg fullDishwasher \rangle$,
$\omega(emptyDishwasher_1) = 1$,
$emptyDishwasher_2 = \langle fullDishwasher, \neg fullDishwasher \rangle$ and
$\omega(emptyDishwasher_2) = 2$. The goal formula is $\gamma = \neg fullDishwasher$.
The individual goal policies in this example are: $\pi_1 = (s_0, emptyDishwasher_2, \gamma)$ and $\pi_2 = (s_0, emptyDishwasher_1, \gamma)$ which means no agent will empty the dishwasher because every agent expects the other agent to empty the dishwasher.

Up to this point, the token has given the player that has it the right to perform an action, a right that none of the other players have. But tokens could also force a player to perform an action.

Consider the definition from before with some changes marked in color:
Given a regular planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, the function *tokenize-force* will transform the regular planning task into a tokenized planning task so that for all $a \in A$:
if $a = \langle pre, \mathit{eff} \rangle$, then $tokenize{-}force(a) = \langle pre \wedge hasToken^{\omega(a)}, \mathit{eff} \wedge doneAction^{\omega(a)} \rangle$
Former we define an Action $giveToken^{ij} = \langle hasToken^i \wedge doneAction^i, \neg hasToken^i \wedge hasToken^j \wedge \neg doneAction^i \rangle$ for all $i, j \in \mathcal{A}$
Then $A^{Token} = \{tokenize(a) | a \in A\} \cup \{giveToken^{ij} | i, j \in \mathcal{A}, i \neq j\}$
Moreover: $\omega^{Token}(tokenize{-}force(a)) = \omega(a)$ for all $a \in A$, and $\omega^{Token}(giveToken^{ij}) = i$ for all $i, j \in \mathcal{A}$.
$s_0^{Token} = s_0 \cup \{\neg hasToken^i | i \in \mathcal{A} \backslash j\} \cup hasToken^j \cup \{\neg doneAction^i | i \in \mathcal{A}\}$ with $j \in \mathcal{A}, |j| = 1$
With j depending on the way the token should be introduced to the token:

1. Table Token: The first agent to take the token off the table.

2. give Token: a specified agent to be determined by the game maker

3. random Token: a random agent

Then $\Pi^{\text{Token}} = \langle s_0^{Token}, A^{\text{Token}}, \omega^{\text{Token}}, \gamma \rangle$

In this case, the giveToken action has to cost something, otherwise the agents would just give each other the token back and forth

The changes imply that each agent can only pass on the token when that agent has performed an action.

This new token version also solves the infinite execution problem.

Proposition 2. *Under the condition of optimal plans one can prevent the appearance of infinite executions in solvable games with asynchronous execution order with the introduction of a force token based execution order.*

proof sketch. The game is finished when the agent that has the token reaches a goal state. When an agent that has a plan gets handed the token, that player first has to perform an action, thereby lowering the subjective cost, and then that player has two possible actions:

1. keep the token and execute another action, which will lower the subjective costs of the remaining policy profile or

2. give the token to the next player. in order to pass on the token the player that has found the plan would have already performed a perspective shift for the next player, in order to check if the next player could also find that plan and calculated the subjective cost of the remaining plan. The player that receives the token will always execute that plan or find a better plan and execute the better. plan.

Every agent that will get the token in the plan will decrease the subjective cost of the policy profile, and because every agent decreases the subjective cost, the game is executable in infinite executions. □

Proposition 3. *Under the condition of optimal plans one can prevent the appearance of deadlocks in solvable games with asynchronous execution order with the introduction of a force token based execution order with the exception of the table token introduction of the token.*

proof sketch. Before any player can give away the token, that player has to perform an action. If a player that has found a plan gets the token, that agent first has to do an action. This contradicts the definition of a deadlock. □

The table token introduction of the token does not prevent deadlocks because in this case, no agent would take the token off the table, which causes a deadlock.

Proposition 4. *Under the condition of optimal plans one can prevent the appearance of deadlocks in solvable games with asynchronous execution order with the introduction of a token based execution order with the exception of the table token introduction of the token, as long as the giveToken action costs at least 1.*

proof sketch. In order to have a policy profile with minimal costs, the agent will have to minimize the amount of giveToken actions in the policy profile. Therefore the agent will always perform an action themself instead of passing the token away, if possible. Because the agents have a common individual policy of keeping the subjective costs minimal, this contradicts the definition of a deadlock. □

A deadlock is different from a dead end. In a dead end, none of the agents' policies prescribe an action, not even for another agent. In the definition (2) above it becomes obvious that these are two different things. **(TODO: is this really needed? It looks unmotivated)**

5 Conclusion

We looked at how the two different token orders solved the two main problems: Deadlocks and infinite executions. We discussed that there are three different ways to introduce a token and all except the takle token solve both problems. The table token does not solve the deadlock problem because there is the possibility that no agent takes the token off the table. This is illustrated in the following table:

	table token	random token	give token
empower token	solves infinite executions doesn't solve deadlocks	solves infinite executions solves deadlocks	solves infinite executions solves deadlocks
force action token	solves infinite executions doesn't solve deadlocks	solves infinite executions solves deadlocks	solves infinite executions solves deadlocks

This means that there are positive results with the introduction of a token based order. In every case, if an agent that has found a plan gets the token, the goal will be reached in a finite number of steps.

Eager agents können weggelassen werden weil man mit Kosten gleich 1 die deadlocks und die infinite executions in den meisten fällen lösen kann. Mit den force tokens können die lazy agenten zu sachen gezwungen werden, es ist also egal ob die agents eager, overeager oder lazy sind.

For future work, the way the token changes the searching of a plan should be researched. The searchtree of a plan might get a smaller width but also a deeper depth.
There could also be future research in researching the fairness of the tokens. How do

the tokens get passed from player to player and if that is fair.
Another field of research for the future could be impatient players and the table token execution order. An impatient player will wait for the other player to take the token first, but only for a limited amout of time.

ToDo Counters

To Dos: 12; 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Parts to extend: 4; 1, 2, 3, 4

Draft parts: 0;

Bibliography

[1] J. Plaza, "Logics of public announcements," in *Proceedings 4th International Symposium on Methodologies for Intelligent Systems*, 1989.

[2] T. Bolander, T. Engesser, R. Mattmüller, and B. Nebel, "Better eager than lazy? how agent types impact the successfulness of implicit coordination," in *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018.

[3] S. Ray and A. De Sarkar, "Execution analysis of load balancing algorithms in cloud computing environment," *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, vol. 2, no. 5, pp. 1–13, 2012.

[4] K. Makki, P. Banta, K. Been, N. Pissinou, and E. Park, "A token based distributed k mutual exclusion algorithm," in *[1992] Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing*, pp. 408–411, IEEE, 1992.

[5] T. Bolander, "A gentle introduction to epistemic planning: The del approach," *arXiv preprint arXiv:1703.02192*, 2017.

[6] H. v. Ditmarsch, W. van der Hoek, and B. Kooi, *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st ed., 2007.

[7] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "Weak, strong, and strong cyclic planning via symbolic model checking," *Artificial Intelligence*, vol. 147, 08 2001.