Bachelor Thesis

How does a token based order compare to the asynchronous 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

August 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

In this thesis, we are going to use epistemic planning for the decision making process in multi-agent planning with distributed knowledge. In multi-agent domains, each agent is independent with the actions of other agents causing changes not instigated by the agent. We have used dynamic epistemic logic to model tokens to avoid problems with deadlocks and infinite executions in implicit coordination. Through tokens, an agent can decide the next active agent. It is possible to solve planning tasks with joint goals without the agents having to negotiate about and commit to a centralized joint policy at plan time. We then analyzed our model by solving infinite executions and deadlocks.

Zusammenfassung

In dieser Arbeit verwenden wir epistemische Planung für den Entscheidungsprozess in der Multi-Agenten Planung mit verteiltem Wissen. In Multi-Agenten-Domänen sind die Agenten unabhängig von den Aktionen anderer Agenten, die Änderungen verursachen, die nicht vom Agenten eingeleitet wurden. Wir haben die dynamische epistemische Logik für die Modellierung von Tokens verwendet, um Probleme mit gegenseitigen Blockaden und unendlichen Ausführungen bei der impliziten Koordination zu vermeiden. Durch Tokens kann ein Agent den nächsten aktiven Agenten bestimmen. Es ist möglich, Planungsaufgaben mit gemeinsamen Zielen zu lösen, ohne dass die Agenten zur Planzeit über eine zentralisierte gemeinsame Strategie verhandeln und sich dazu verpflichten müssen. Wir haben dann unser Modell evaluiert, indem wir unendliche Ausführungen und Deadlocks gelöst haben.

Contents

1 Introduction

Implicit coordination is a technique for planning and coordination of multi-agent systems in which the agents try to collaboratively reach a joint goal. The knowledge and abilities needed to reach a goal can be distributed among the agents. With no centralized coordination instance, the agents need to plan individually and decentrally execute their plans.

The field of dynamic epistemic logic is relatively young. It started with the work from Plaza [1] about "Logics of public communications". Epistemic planning investigates techniques and tools of automated planning with formal semantics of communicative actions and knowledge representation. 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.

With the asynchronous execution order, every agent can intervene in the execution order at any time. This can potentially harm the execution and prevent the agents from reaching a goal. For example, if two agents have opposing goals with no knowledge about the other agents' goal and the common goal is to reach one of the individual goals, then one agent would always undo the actions of the other agent.

To prevent this from happening, we want to exclude all agents except for the acting agent from taking action. The acting agent should also be able to specify the next acting agent. To model this, we used a token that the acting agent can pass on to the next agent. Only the agent with the token is allowed to participate.

This thesis is structured as follows: In section 2, we are going to give a brief overview of the related work to this thesis and highlight how this thesis is built on that work. In section 3 we will introduce the formal framework, dynamic epistemic logic (DEL) with modeling beliefs, states and actions. This logic is explained using a specific example, introduced in the beginning. In Section 4, we are going to formally introduce tokens to an existing planning problem and show how this approach solves infinite executions and deadlocks. In Section 5 we have the Conclusion of this thesis and the future work still nedded to be done.

2 Related Work

This thesis is built on the work of Bolander et al[2]. They investigated how agent types impact the success of implicitly coordinated plans. They introduced three agent types, lazy agents that have a preference against their own actions, eager agents, who have a preference for their own actions and optimally eager agents that try to come up with optimal policies that reach the goal in the fewest number of steps. The problem with lazy agents is that they can produce deadlocks by expecting the other agents to act. This does not happen with eager agents, but eager agents can also potentially be over-eager and produce infinite executions by unintentionally working against each other. To solve this, they introduced optimally eager agents. But optimally eager agents are only guaranteed to prevent infinite executions if there is uniform observability. They had a sequential action execution, but if two agents want to act at the same time, then the first agent who acts is not predefined. This could lead to the agents unintentionally sabotaging each other.

The formalism for epistemic planning in this thesis follows the work of Bolander et al.[3]. They introduced the internal perspective of an agents view of the world to model partial observability in planning. The view of one agent can differ from other agents if the agent has different knowledge.

In recent work, Nebel et al.[4] investigated how implicitly coordinated plans without communications can succeed. For this they used multi-agent path finding in which agents have to move to different destinations in a collision free manner. They first

assumed that the destinations of the agents were common knowledge and the agents could not communicate. If agents are eager and have conservative replanning or plan optimally, the plans would succeed. Then they dropped the assumption and researched how no communication worked with unknown destinations. They found that if the agents are eager and replan conservatively, they could also succeed.

Tokens have also made an appearance in distributed systems for example in the work from Loucks et al.[5]. A distributed system 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 achieve a common goal. This makes that field face some similar problems like the coordination of actions, the concurrency of the agents and the in-transparency. The field also needs scalable solutions.

In the field of distributed systems, there are a few load balancing algorithms that try to equally distribute a task among several agents. Ray et al.[6] analyzed 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 restrictional use of a mutual resource that can only have a small number of users at a time. This can be achieved by u**(TODO: richtig zitieren)**sing a token queue and a token semaphore, as from Makki et al.[7]

In contrast to this thesis' approach, their tokens are used to restrict the access to a limited resource. Their agents do not plan with other agents and handing the token to the next player is usually done by a waiting queue. One focus of this thesis is that the agent who has the token gets to decide which agent will have the token next. This is not wanted in distributed systems because they want all agents to have the same rights of access to a resource with no agent being strategically excluded.

3 Background

Imagine the following task between two players, Anne and Bill. Between Anne and Bill is a lever 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. Anne does not know Bill has a goal on his side, and Bill does not know Anne has a goal on her side. This can be seen in figure 1.

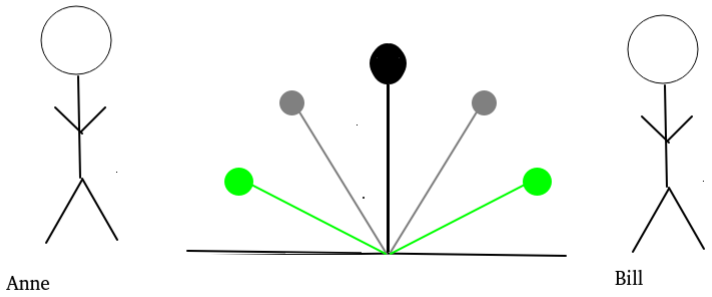


Figure 1: Between Anne and Bill is a lever, it is upright in the black position with two grey positions to the sides. The green positions mark the goal positions of Anne and Bill.

With the asynchronous execution order this task could be easily finished if Anne or Bill just pulled the lever two times in their direction. But just as easily this task

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 task from going on infinitely? To answer this question we first need to introduce the formal framework.

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 work of Bolander et al. [2] and [8] and the book "Dynamic epistemic logic" from Ditmarsch [9].

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: $\neg K_b p \wedge K_b q$.

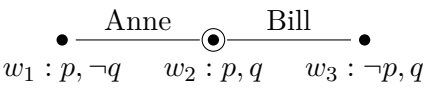
Formulas are evaluated in epistemic models

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

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 $V : \mathcal{P} \to \mathcal{P}(W)$ assigning a valuation to each atomic proposition.

Using Anne and Bill as an example, Anne only knows p is true. She does not know if $p \wedge q$ or $p \wedge \neg q$, so she sees two worlds. Those two worlds are indistinguishable for Anne. Bill only knows q is true, he does not know if $p \wedge q$ or $\neg p \wedge q$. One world where just the lever to the left is a goal, a world where another goal is the lever to the right and a world where just the lever to the right is a goal. Let w_1 be the world where just the lever to the left is a goal, w_2 be the world where the goal is the lever to the left or to the right and w_3 be the world where the goal is the lever to the right. Then $w_1 \sim_{\text{Anne}} w_2$ and $w_2 \sim_{\text{Bill}} w_3$. 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. In all graphic representations in this thesis we usually omit reflexive edges and edges that can be implied by transivity for better readability.

$s =$



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.

In the example above, Anne has the local state $s^{\text{Anne}} = (\mathcal{M}, \{w_1, w_2\})$ and Bill has the local state $s^{\text{Bill}} = (\mathcal{M}, \{w_2, w_3\})$.

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

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

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

From the example above, in $(\mathcal{M}, w_2)$, w_2 is the designated world. In this designated world, Anne still does not know if q is true or not. $(\mathcal{M}, w_2) \models \neg K_a q \wedge \neg K_a \neg q$. Performing a perspective sift on $s = (\mathcal{M}, w_2)$ for Anne is $(\mathcal{M}, w_2)^{\text{Anne}} = (\mathcal{M}, \{w_1, w_2\})$. This perspective shift gives us $(\mathcal{M}, \{w_1, w_2\}) \not\models q$ and $(\mathcal{M}, \{w_1, w_2\}) \not\models \neg q$. This way, we can verify the knowledge from the global world for an agent by performing a perspective shift for that agent.

3.1.1 Epistemic Actions and Product Updates

An event model is a 4-tuple $\mathcal{E} = \langle E, (\sim_i)_{i\in\mathcal{A}}, pre, \mathit{eff}\rangle$ 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 :

- each world is paired up with all applicable events
  $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$;

- new worlds are indistinguishable if the old worlds and the events are indistinguishable
  $\sim'_i = \{((w, e), (w', e')) \in W' \times W' \mid w \sim_i w' \text{ and } e \sim_i e'\}$;

- propositions become true if the proposition occurred as a positive literal in the effect of the event or the proposition was satisfied in the world before and did not appear negative in the effect
  $V'(p) = \{(w, e) \in W' \mid \mathit{eff}(e) \models p \text{ or } (\mathcal{M}, w \models p \text{ and } \mathit{eff}(e) \not\models \neg p)\}$;

- worlds are designated if both predecessor world and event are designated.
  $W'_d = \{(w, e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$.

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

Let us imagine a planning task between Lea and Max. Lea puts two cards face down in front of Max. The goal is that Max picks up the queen. The problem here is that Max does not know where the queen is.
Lea has put the queen to the right, so qR.

$$s = \underset{w_1 : \neg qR}{\bullet} \overset{\text{Max}}{\text{———}} \underset{w_2 : qR}{\circledcirc}$$

There are two types of actions that could be performed to solve this problem:

1. Announcement actions

   Lea could tell Max where she put the queen. $\omega(a_{\text{announcement}}) = \text{Lea}$

$$\underset{w_1 : \neg qR}{\bullet} \overset{\text{Max}}{\text{———}} \underset{w_2 : qR}{\circledcirc} \quad \otimes \quad \underset{e_1 : \langle qR, \top \rangle}{\circledcirc} \quad = \quad \underset{(w_2, e_1) : qR}{\circledcirc}$$

2. Sensing actions

   Max could look at the left card, then he knows where the queen is. $\omega(a_{\text{sense}}) =$ Max

$$\underset{w_1 : \neg qR}{\bullet} \overset{\text{Max}}{\text{———}} \underset{w_2 : qR}{\circledcirc} \quad \otimes \quad \underset{e_1 : \langle qR, \top \rangle e_2 : \langle \neg qR, \top \rangle}{\bullet \qquad \circledcirc} \quad = \quad \underset{w_1 : \neg qR}{\bullet} \qquad \underset{w_2 : qR}{\circledcirc}$$

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* $\gamma \in \mathcal{L}_{KC}$.

Consider a version of the lever problem from before. 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 $s_0 = \underset{\neg p}{\circledcirc}$ with the lever being in the upright position. The action $a_1 = \underset{e_1 : \langle \top, p \rangle}{\circledcirc}$ 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.

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

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 $s^{\omega(a)} = t^{\omega(a)}, a \in \pi(s) : a \in \pi(t)$

   A planning task with the agents Lea and Max, initial state Lea puts two cards face down on the table, one of them is a queen, the other a king. Max does not know if the queen is on the left or the right side. The goal is that Max

turns over the queen. Max has two actions, he can either turn over the left card or the right card. The policy in which Max turns over the left card is a valid policy because even though he does not know if that card is the queen, the action is prescribed in both cases.

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

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 $s_n \models \gamma$.

In the lever example, a policy could be that, starting in the position with the lever being in the middle, Bill pulls the lever to the right and then to the right again. This is a policy that satisfies all properties and where the last state satisfies the goal

formula. The execution of the policy is finite and successful.
For Bill, this is a reasonable policy. It satisfies the goal formula he knows to be true, with the lever being at the far right position. But for Anne, it is not a reasonable policy because she can not see that the execution of this policy will satisfy the goal formula.

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. [10]). 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 execution 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 distinguish inbetween. 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.

The policy from above, with Bill pulling the lever to the right twice is Bill-strong but not Anne-strong.

Sometimes the agents cannot coordinate their plans but rather have to come up with plans individually. These plans can differ a lot, the agents often have different knowledge about the states, the actions and therefore the 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. $s_{m+1} \in \mathit{Globals}(s_m \otimes a_{m+1})$

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

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 though each individual policy is strong.

The cost of a a policy can be defined as its worst-case execution length, the number of actions in its longest possible execution. An optimal policy is one with minimal costs. Due to partial observability and different knowledge, the agents might assign different costs to the same policy and therefore the costs are subjective.
Let π be a strong policy for a planning task Π. The perspective-sensitive cost (or simply cost) of π from a state $s \in \mathit{Dom}(\pi)$, denoted $\kappa_\pi(s)$ is defined as:

$$\kappa_\pi(s) = \begin{cases} 0 & \text{if there exists no } a \in \pi(s) \\ 1 + \max_{a \in \pi(s), s' \in \mathit{Globals}(s^{\omega(a)} \otimes a)} \kappa_\pi(s') & \text{else.} \end{cases}$$

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.

There are different ways to model the token. We are going to start with adding an action that lets one agent hand the token to the next agent. Modeling the token in the beginning of the planning task is a little bit different, since there are three ways the token can be introduced to the agents.

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 planning task it is also specified which agent will have the token in the beginning.
   The problem with this introduction is that in every new planning task, this has to be written in the definition of the task. In order for the planning task to be efficient, it should be given to an agent 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 planning task from ever reaching a goal state.

There are also two different types of tokens. The first type of token can be easily passed from one agent to the next without any restriction. This token makes sure that no unwanted agent performs an action that could lead to a worse plan than before. the second type of token forces an agent to perform an action before handing the token to the next agent. This type of token makes sure that the token is not just passed between the agents, it makes sure the agents contribute to reaching a goal state.

4.1 Infinite executions to finite executions

We are now going to describe a function that takes a 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 planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, the function *tokenize* will transform the planning task into a tokenized planning task so that for all $a \in A$: if $a = \langle pre, \mathit{eff} \rangle$, then $tokenize(a) = \langle pre \wedge hasToken_{\omega(a)}, \mathit{eff} \rangle$.
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 $i, j \in \mathcal{A}$.
With s_0 depending on the way the token should be introduced to the token:

1. Table Token: The first action in this planning task is for one agent to take the token off the table. Here we need a new action that, if no player has the token,

one player can take the token.

$takeToken_j = \langle \bigwedge_{i \in \mathcal{A}} \neg hasToken_i, hasToken_j \rangle \forall j \in \mathcal{A}$ with $\omega(takeToken_j) = j$.

Then the planning task starts with no agent having the token **(TODO:** $s_0 \cup \neg hasToken_i$ **macht keinen sinn. Was macht der Mengenvereinigung-soperator auf Zuständen und Propositionen? * Auch in den anderen Fällen muss die Konstruktion vom neuen** s_0 **noch formal verbessert werden.)** $s_0^{tableToken} = s_0 \cup \neg hasToken_i$. $V_{tableToken}(hasToken_i) = \emptyset \qquad \forall i \in \mathcal{A}$

$V_{tableToken}(p) = V(p) \qquad \forall p \in P : p \neq hasToken$

2. give Token: a specified agent to be determined by the specifications of the planning tasks. We can model this by having $s_0^{giveToken}(j)$ have the variable j, the agent that is supposed to have the token in the beginning. $s_0^{giveToken}(j) = s_0 \cup \{\neg hasToken_i | i \in \mathcal{A} \backslash j\} \cup hasToken_j$

3. random Token: a random agent to model this we need to define some other things first. $W_{randomToken} = \{w_i | w \in W, i \in \mathcal{A}\}$

   $w^i \sim_{randomToken} v^i$ iff $w \sim v$ and $i = j$

   $V_{randomToken}(p) = \{w_i | w \in V(p), p \neq hasToken, i \in \mathcal{A}\}$

   $V(randomToken_i) = \{w_i | w \in W\}$

   $W_d^{randomToken} = \{w_i | w \in W_d, i \in \mathcal{A}\}$

   $s_0^{randomToken} = \langle W_{randomToken}, \sim_{randomToken}, V_{randomToken}, W_{randomToken} \rangle$

   The intuition behind this is duplicating the model for each agent. Because then there are multiple designated worlds, which leads to nondeterminism between the worlds.

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

The action *giveToken* can have different costs. One option would be to have the the action cost nothing, then the overall cost of the plan would not change. The problem with this is it does not prevent the agents from just passing the token around without

a goal. The other option is that the action has the cost 1, then the agents will not just pass the token around, they will prefer to act themselves if it is more or equally efficient than to let someone else act before handing the token to the next agent.

In searching for the optimal plan the search tree can have a smaller width, because the amount of agents that could perform the next action is limited. Therefore the search tree will probably have a deeper depth, because the action handing off the token will also have to be modeled. This could be researched in the future.

Given this formalization, we can now show that we can stop infinite executions from happening in planning tasks by transforming the planning task into a tokenized planning task.

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

proof sketch. The execution is finished when an agent that has the token does not want to do anything.
When an agent i that has a plan gets handed the token, that player has three possible actions:

1. to keep the token and execute an action a, which will lower the subjective costs of the remaining policy by at least 1 because the next action is an own action.

2. to give the token to another agent. In order to pass on the token, the agent that has found the plan would have already performed a perspective shift for the next agent in order to check if the next agent could also find that plan and calculated the subjective cost of the remaining plan. The agent that receives the token will only act with respect to a plan that is at least as good as the plan envisioned by agent i.

3. to do nothing. This will stop the execution.

Every agent that will get the token in the plan will either decrease the subjective cost of the policy or stop the execution. Because every agent in the execution decreases the subjective cost, the planning task 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

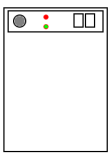


Figure 2: Dishwasher

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.

Let initially, in s_0, the dishwasher be full. Agent 1 has an action a_1 and Agent 2 has an action a_2 that can both be applied in s_0 and have the effect that has the effect that the dishwasher is empty afterwards. Then the policies $\pi_1 = \{s_0 \overset{a_2}{\longmapsto} emptyDishwasher\}$ and $\pi_2 = \{s_0 \overset{a_1}{\longmapsto} emptyDishwasher\}$ are both i-strong policies.

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 planning task $\Pi = \langle s_0, A, \omega, \gamma \rangle$, the function *tokenize-force* will transform the planning task into a tokenized planning task so that for all $a \in A$:
if $a = \langle pre, eff \rangle$, then $tokenize{-}force(a) = \langle pre \wedge hasToken_{\omega(a)}, eff \wedge doneAction_{\omega(a)} \rangle$
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 s_0 depending on the way the token should be introduced to the token:

1. Table Token: The first action in this planning task is for one agent to take the token off the table. Here we need a new action that, if no player has the token, one player can take the token.
   $takeToken_j = \langle \bigwedge_{i \in \mathcal{A}} \neg hasToken_i, hasToken_j \rangle \forall j \in \mathcal{A}$ with $\omega(takeToken_j) = j$.
   Then the planning task starts with no agent having the token or having done an action
   $s_0^{tableToken} = s_0 \cup \neg hasToken_i \cup \neg doneAction_i$
   $V_{tableToken}(hasToken_i) = \emptyset$ and $V_{tableToken}(doneAction_i) = \emptyset \qquad \forall i \in \mathcal{A}$
   $V_{tableToken}(p) = V(p) \qquad \forall p \in P : p \neq hasToken, doneAction$

2. give Token: a specified agent to be determined by the specifications of the planning tasks. We can model this by having $s_0^{giveToken}(j)$ have the variable j,

the agent that is supposed to have the token in the beginning.
$s_0^{giveToken}(j) = s_0 \cup \{\neg hasToken_i | i \in \mathcal{A} \backslash j\} \cup hasToken_j \cup \{\neg doneAction_i | i \in \mathcal{A}\}$

3. random Token: a random agent to model this we need to define some other things first. $W_{randomToken} = \{w_i | w \in W, i \in \mathcal{A}\}$
$w^i \sim_{randomToken} v^i$ iff $w \sim v$ and $i = j$
$V_{randomToken}(p) = \{w_i | w \in V(p), p \neq hasToken, i \in \mathcal{A}\}$
$V(randomToken_i) = \{w_i | w \in W\}$
$W_d^{randomToken} = \{w_i | w \in W_d, i \in \mathcal{A}\}$
$s_0^{randomToken} = \langle W_{randomToken}, \sim_{randomToken}, V_{randomToken}, W_d^{randomToken} \rangle$
(TODO: Wie modelliere ich hier doneAction rein?) The intuition behind this is duplicating the model for each agent. Because then there are multiple designated worlds, which leads to nondeterminism between the worlds.

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

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.

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. Also, the deadlock in this case can only appear in the initial state, in every other state there is always exactly one acting agent. **(TODO: Ist das so verständlich?)**

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

proof sketch. The execution is finished when an agent that has the token does not want to do anything.
When an agent i 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. to keep the token and execute another action, which will lower the subjective costs of the remaining policy profile or

2. to give the token to another agent. In order to pass on the token, the agent that has found the plan would have already performed a perspective shift for the next agent in order to check if the next agent could also find that plan and calculated the subjective cost of the remaining plan. The agent that receives the token will only act with respect to a plan that is at least as good as the plan envisioned by agent i.

Every agent that will get the token in the plan will either decrease the subjective cost of the policy or stop the execution. Because every agent in the execution decreases the subjective cost, the planning task is executable in infinite executions. □

Proposition 3. *Under the condition of optimal plans one can prevent the appearance of deadlocks in planning tasks with asynchronous execution order with the introduction of a force token based execution order with the give token and random 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. □

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 every agents policy there is an exact sequence of acting agents. Because only one agent can act, the state that some agents are waiting on each other will never appear. The one acting agent will then specify the next acting agent, therefore there always will be something done. □

With these results, we do not need to enforce the agents to be eager to avoid deadlock if we use the tokens with costs. We also do not need optimally eager agents to prevent infinite executions if we use tokens. The only problem with tokens is the introduction of the token, especially with the table token. But as long as an agent takes the token off the table, there will be no more deadlocks.

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 table 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 $\longrightarrow$ Proposition 1 doesn't solve deadlocks	solves infinite executions $\longrightarrow$ Proposition 1 solves deadlocks $\longrightarrow$ Proposition 4	solves infinite executions $\longrightarrow$ Proposition 1 solves deadlocks $\longrightarrow$ Proposition 4
force action token	solves infinite executions $\longrightarrow$ Proposition 2 doesn't solve deadlocks	solves infinite executions $\longrightarrow$ Proposition 2 solves deadlocks $\longrightarrow$ Proposition 3	solves infinite executions $\longrightarrow$ Proposition 2 solves deadlocks $\longrightarrow$ Proposition 3

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.

For future work, the way the token changes the search of a plan should be researched. With Tokens, the search tree of a plan might have a smaller width because the agents can be prevented from acting. The tree will also have a deeper depth because the agents each have a new action of handing the token to the next player.
We also did not look at how a tokenized planning task changes the existence of i-strong policies. Could some planning problems become unsolvable through the tokenize function?

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 amount of time. For this we would need to extend the formalism since we cannot model time or a waiting agent explicitly.

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] T. Bolander and M. B. Andersen, "Epistemic planning for single-and multi-agent systems," *Journal of Applied Non-Classical Logics*, vol. 21, no. 1, pp. 9–34, 2011.

[4] B. Nebel, T. Bolander, T. Engesser, and R. Mattmüller, "Implicitly coordinated multi-agent path finding under destination uncertainty: success guarantees and computational complexity," *Journal of Artificial Intelligence Research*, vol. 64, pp. 497–527, 2019.

[5] L. K. Loucks and A. A. Shaheen, "System and method for multi-level token management for distributed file systems," May 27 1997. US Patent 5,634,122.

[6] 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.

[7] 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.

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

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

[10] 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.