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 07$^{\text{th}}$, 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.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Algorithms

1 Introduction

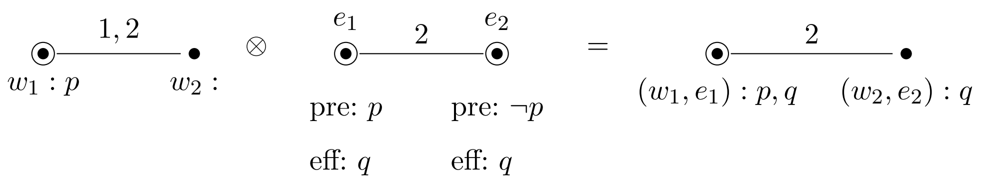
How would it be if we had a perfect execution order?

This thesis is trying to

2 Related Work

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

2.1 An Example



3 Background

Imagine there is a lever between two players, (Anne and Bill) it is upright in the middle position and has 5 positions in total. Player one thinks the lever should be pulled two positions to her side and player two thinks the lever should be pulled two positions to his side. What do the players do? **(TODO: Bild malen mit Hebel)**

3.1 DEL

DEL, or Dynamic Episthemic Logic is a specific mathmatical language used as the framework. Let $\mathcal{A}$ be a finite set of Agents. Let $\mathcal{P}$ be a finite set of atomic propositions. The epistemic language $\mathcal{L}_{\mathrm{KC}}$ 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 $i \in \mathcal{A}$. $K_{i\varphi}$ reads as "Agent i knows φ". C_{φ} reads as "it is common knowledge that φ ". **(DRAFT: Let's look back at the example.)** 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: $K_a p \vee \neg K_a \neg q$. Bill knows that one goal position is the right, but does not know the other position: $\neg K_b \neg p \vee K_b q$.

Formulars are evaluated in episthemic Models

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

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

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 $Globals(s) = \{(\mathcal{M}, w)|w \in W_d\}$ The state $(\mathcal{M}, W_d)$ is called a local state for agend i if W_d is closed under $\sim_i$ (that is , if $w \in W_d$ and $w \sim_i v$ implies $v \in W_d$). Given a state $s = (\mathcal{M}, W_d)$ the associated local state of agent i, denoted s^i, is $(\mathcal{M}\{v|v \sim_i w \text{ and } w \in W_d\})$.

Let $(\mathcal{M}, W_d)$ be a state on $P, \mathcal{A}$ with $\mathcal{M} = (W, (\sim_i)_{i\in\mathcal{A}}, L)$. For $i \in \mathcal{A}$, $p \in P$ and $\varphi, \psi \in \mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$, we define truth as follows:

$$\begin{array}{lll}
(\mathcal{M}, W_d) \models \varphi & \text{iff} & (\mathcal{M}, w) \models \varphi \text{ for all } w \in W_d \\
(\mathcal{M}, w) \models p & \text{iff} & p \in L(w) \\
(\mathcal{M}, w) \models \neg\varphi & \text{iff} & (\mathcal{M}, w) \not\models \varphi \\
(\mathcal{M}, w) \models \varphi \wedge \psi & \text{iff} & (\mathcal{M}, w) \models \varphi \text{ and } (\mathcal{M}, w) \models \psi \\
(\mathcal{M}, w) \models K_i\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim_i w \\
(\mathcal{M}, w) \models C\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim^* w
\end{array}$$

where $\sim^*$ is the transitive closure of $\bigcup_{i\in\mathcal{A}} \sim_i$.
A state $(\mathcal{M}, W_d)$ is called a *local state* for agent i if W_d is closed under $\sim_i$. 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.

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 te domain E is a non-empty finite set of events; $\sim_i \subseteq E^2$ is an equivalence relation called the indistinguishability 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 postcondition, or effect to each event. For all $e \in E$, $\mathit{eff}(e)$ is a conjunction of literals, that means anatomic 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 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 $|E| = 1$, we will write $\mathcal{E} = (pre(e), \mathit{eff}(e))$.

The product update is used to specify the next state resulting from preforming 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 :

- $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$;
- $\sim'_i = \{((w, e), (w', e')) \in (W')^2 \mid w \sim_i w' \text{ and } e \sim_i e'\}$;
- $V'(p) = \{(w, e) \in W' \mid \mathit{eff}(e) \models p \text{ or } (\mathcal{M}, w \models p \text{ and } \mathit{eff}(e) \not\models \neg p)\}$;
- $W'_d = \{(w, e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$.

$a = (\mathcal{E}, E_d)$ 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 $(\mathcal{M}, w) \models pre(e)$.

3.1.2 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 $\omega(a)$. **(EXTEND: Definition)**

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.

A policy π for $\Pi = \langle s_0, A, \omega, \gamma \rangle$ is a partial mapping $\pi : S^{gl} \hookrightarrow \mathcal{P}(A)$ so 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 presscribes 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 describe 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 preform 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 ating 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: Do i really need this?) **(EXTEND: übergang)** guaranteed to archieve 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) **(TODO: Quelle?)**.

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 extention 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 iniially 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 agent cannot coordinate their plans but rather have to come up with plans indivudually. 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 b instantaneous and executed asynchronnously. 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 possiblity 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.

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 Approach

Lets go back to 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.

4.0.1 Tokens

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 $tokenize(a) = \langle pre \wedge has(\omega(a), token, eff)$
Former we define an Action $giveToken^{ij} = \langle has(i, token), \neg has(i, token) \wedge has(j, token) \rangle$ 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: $\omega^{Token}(tokenize(a)) = \omega(a)$ for all $a \in A$, and $\omega^{Token}(giveToken^{ij}) = i$ for all $i, j \in \mathcal{A}$.
Then $\Pi^{\text{Token}} = \langle s_0, A^{\text{Token}}, \omega^{\text{Token}}, \gamma \rangle$ **(TODO: Beschreibung fehlt)**

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

proof sketch. The game is finished when the Agend that has the token reaches a goal state. When an agend that has a plan with the subjective cost of $n + m, m \leq 1$, that agent will start executing that plan and if needed, hand over the token to the next player. The agend will always decrease the subjective cost to n because handing over the token will also decrease the subjective cost. Because every agent decreases the subjective cost, the game is executable in infinite executions. □

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

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

3. $\omega(a) \neq i$

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

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 following problem: 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. **(EXTEND: hier vielleicht formal noch mal darstellen?)** In this example you can see that each agent expects the other agent to act, therefore no agent will act.

Here the introduction of tokens forces a player to make a move because the player that has the token has to perform an action. In this example, every action reaches a goal state, even if it is not in the own players policy.

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.

The problem with the tokens is the allocation of the token in the beginning. Consider the following example as the main example in this chapter.

4.1 Lever Problem

The lever has a position betreen -2 and 2.

There are two players that each want to move a lever in a direction. Player 1s' goal is to move the lever to the left, to the position -2 and player 2s' goal is to move the lever to the right, to the positon 2. They do not know the other players' goal and they do not know if there is another goal than their own goal.

4.2 Asynchron execution order

The first possibility is that the players move asynchron, seemingly random. They each have and action to move the lever to the right and to the left. In the beginning the lever is in the middle.

$A = \{\text{Move\_R}(agt, obj, pos) \mid agt \in \mathcal{A} \ \& \ obj \in Obj \ \& \ pos \in \{-2; 1\}\} \cup$
$\{\text{Move\_L}(agt, obj, pos) \mid agt \in \mathcal{A} \ \& \ obj \in Obj \ \& \ pos \in \{-1; 2\}\}$
where $\forall \ agt \in \mathcal{A}, pos \in \{-2; 2\}, obj \in Obj,$

- $\text{Move\_R}(obj, pos) = \langle \text{At}(obj, pos), \text{At}(obj, pos + 1) \wedge \neg\text{At}(obj, pos)\rangle$

- $\text{Move\_L}(obj, pos) = \langle \text{At}(obj, pos), \text{At}(obj, pos - 1) \wedge \neg\text{At}(obj, pos)\rangle$

$\mathcal{A} = \{Player1, Player2\}$
$\varphi_g^{Player1} = \text{At}(\text{lever}, -2)$ **(TODO: das g hier muss anders)**
$\varphi_g^{Player2} = \text{At}(\text{lever}, 2)$
$\varphi_g = \text{At(lever,L}_2) \vee \text{At(lever,R}_2)$?
$s_0 = \{\text{At}(\text{lever}, 0)\}$

(TODO: Die Spielabfolge und das Problem ausführlich formulieren)

The agents here only have an incentive to move the lever in the direction towards their own goal. One possible execution would be that the agents each move the lever one step towards their goal and the other agent would move the lever back to the starting position. This is a infinite sequence which is not successful, no player would reach any goal.

There are three ways to introduce a token to this game.

1. Tabel token - the token is lying on a table and one of the agents can take the token in the beginning.
   The problem here is in the example before with the dishwasher. In this example, no player 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. It has to be decided in the beginning.

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. An agent who knows nothing and has no action will prevent the game from ever reaching a goal state.

4.3 Token versions

4.3.1 table token

$A = \{\text{Move\_R}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-2; 1\}\} \cup$
$\{\text{Move\_L}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-1; 2\}\}$

$\{\text{Take\_Token}(agt, token) \mid agt \in \mathcal{A} \;\&\; token \in Token\}$
$\{\text{Give\_Token}(agt, token, otheragt) \mid agt \in \mathcal{A} \;\&\; token \in Token \;\&\; otheragt \in \mathcal{A} \backslash agt\}$
where $\forall\ agt \in \mathcal{A}, pos \in \{-2; 2\}, obj \in Obj, |Token| = 1$

- $\text{Move\_R}(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos + 1) \wedge \neg\text{At}(obj, pos)\rangle$

- $\text{Move\_L}(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos - 1) \wedge \neg\text{At}(obj, pos)\rangle$

- $\text{Take\_Token}(agt, token) = \langle \neg\text{Has}(agt, token) \wedge \text{At}(token, table), \text{Has}(agt, token) \wedge \neg\text{At}(token, table)\rangle$

- $\text{Give\_Token}(agt, token, otheragt) = \langle \text{Has}(agt, token), \neg\text{Has}(agt, token) \wedge \text{Has}(otheragt, token)\rangle$

$s_0 = \{\text{At(Hebel, N)}, \text{At(Token, Table)}\}$

4.3.2 give Token

$A = \{\text{Move\_R}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-2; 1\}\} \cup$
$\{\text{Move\_L}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-1; 2\}\}$
$\{\text{Give\_Token}(agt, token, otheragt) \mid agt \in \mathcal{A} \;\&\; token \in Token \;\&\; otheragt \in \mathcal{A} \backslash agt\}$
where $\forall\ agt \in \mathcal{A}, pos \in \{-2; 2\}, obj \in Obj, |Token| = 1$

- $\text{Move\_R}(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos + 1) \wedge \neg\text{At}(obj, pos)\rangle$

- $\text{Move\_L}(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos - 1) \wedge \neg\text{At}(obj, pos)\rangle$

- Give_Token$(agt, token, otheragt) = \langle \text{Has}(agt, token), \neg\text{Has}(agt, token) \wedge \text{Has}(otheragt, token)\rangle$

$s_0 = \{\text{At(Hebel, N), Has}(\text{agent} \in \mathcal{A}, \text{Token})\}$

4.3.3 random token

$A = \{\text{Move\_R}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-2; 1\}\} \cup$
$\{\text{Move\_L}(agt, obj, pos, token) \mid agt \in \mathcal{A} \;\&\; obj \in Obj \;\&\; pos \in \{-1; 2\}\}$
$\{\text{Give\_Token}(agt, token, otheragt) \mid agt \in \mathcal{A} \;\&\; token \in Token \;\&\; otheragt \in$
$\mathcal{A} \backslash agt\}$
where $\forall\ agt \in \mathcal{A}, pos \in \{-2; 2\}, obj \in Obj, |Token| = 1$

- Move_R$(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos + 1) \wedge \neg\text{At}(obj, pos)\rangle$

- Move_L$(agt, obj, pos, token) = \langle \text{At}(obj, pos) \wedge \text{Has}(agt, token), \text{At}(obj, pos - 1) \wedge \neg\text{At}(obj, pos)\rangle$

- Give_Token$(agt, token, otheragt) = \langle \text{Has}(agt, token), \neg\text{Has}(agt, token) \wedge \text{Has}(otheragt, token)\rangle$

$s_0 = \{\text{At(Hebel, N), Has}(\text{randomagent} \in \mathcal{A}, \text{Token})\}$

5 Conclusion

6 Acknowledgments

First and foremost, I would like to thank...

- advisers
- examiner
- person1 for the dataset
- person2 for the great suggestion
- proofreaders

ToDo Counters

To Dos: 6;   1, 2, 3, 4, 5, 6

Parts to extend: 3;   1, 2, 3

Draft parts: 1;   1

Literaturverzeichnis