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

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

# 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?

# 2 Related Work

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

# 3 Background

Explain the math and notation.

**(EXTEND: This needs the definition)**

## 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:

$$\varphi :== \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_{i\varphi} \mid C_{\varphi}$$

**(TODO: Was bedeutet das Top und bottom und warum stehen die dort?)**

with $p \in \mathcal{P}$ and $i \in \mathcal{A}$. $K_{i\varphi}$ reads as "Agent $i$ knows $\varphi$". $C_{\varphi}$ reads as "it is common knowledge that $\varphi$ ".

Formulars are evaluated in episthemic Models

$$\mathcal{M} = (W, (\ _i)_{i\in\mathcal{A}}, L)$$

with the domail $W$ being a nonempty finite set of worlds, $( \ _i)_{i \in \mathcal{A}}$ being an equivalnce relation calle dthe indistinguishability relation for agend $i \in \mathcal{A}$ and

# 4 Approach

## 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)**

$\varphi_g^{Player2} = \text{At}(\text{lever}, 2)$

$\varphi_g = \text{At}(\text{lever,L}_2) \vee \text{At}(\text{lever,R}_2)$ ?

$s_0 = \{\text{At}(\text{lever}, 0)\}$

**(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.

The solution to this problem could be the introduction of a token, which gives one agent the ability to preform multiple actions and prevents the other agents from preforming any action.

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.

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

## 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 $\forall\ agt \in \mathcal{A}, pos \in \{-2; 2\}, obj \in Obj, |Token| = 1$

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

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

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

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

## 4.4 TODO

**Proposition 1.** *Under the condition of optimal plans one can prevent infinite executions with tokens.*

*Proof Sketch.* The token prevents any other player from making a move that is not a part of an optimal plan of the first player to have the token and an optimal plan. □

**Proposition 2.** *If an Agent gets the token that has found a Plan, then the Game is executable in infinite executions.*

*Proof Sketch.* In every version of the game if one agent has found a Plan, even if the others have not, and that agent starts executing the plan, whenever that player gives the token away the recieving player knows that the agent with the plan wants the agent to act. then the plan can be executed.

1. Tabel token - The agent with a plan will take the token and set the plan in Motion.

2. give token - In the game initialization the token just has to be passed to someone who has some kind of plan. Even if that player just passes on the plan.

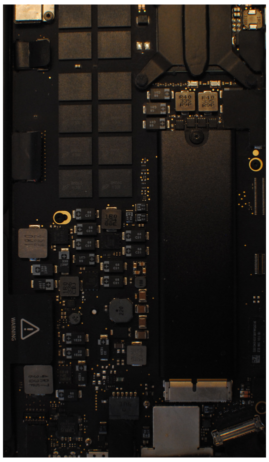
3. random token -

□

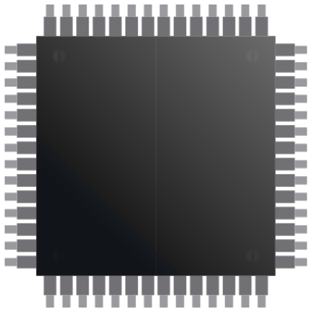
# 5 Experiments

| Type | Accuracy |
|---|---|
| A | $82.47 \pm 3.21$ |
| B | $78.47 \pm 2.43$ |
| C | $84.30 \pm 2.35$ |
| D | $86.81 \pm 3.01$ |

**Tabelle 1: Table caption.** foo bar...



**(a)** Some cool graphic



**(b)** Some cool related graphic

**Abbildung 1: Caption that appears under the fig** Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# 6 Conclusion

# 7 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: 3;   1, 2, 3

Parts to extend: 1;   1

Draft parts: 0;

# Literaturverzeichnis