VILNIUS UNIVERSITY

Adomas Birštunas

SEQUENT CALCULI WITH AN EFFICIENT LOOP-CHECK FOR BDI LOGICS

Doctoral dissertation
Physical sciences, Informatics (09P)

Vilnius, 2010

This work was performed in 2004-2009 at Vilnius University, Lithuania.

Research supervisor :

Assoc. Prof. Habil. Dr. Regimantas Pliuškevičius (Institute of Mathematics and Informatics, phisical science, mathematics - 01P)

VILNIAUS UNIVERSITETAS

Adomas Birštunas

SEKVENCINIAI SKAIČIAVIMAI BDI LOGIKOMS SU EFEKTYVIA CIKLŲ PAIEŠKA

Daktaro disertacija
Fiziniai mokslai, Informatika (09P)

Vilnius, 2010

Disertacija rengta 2004-2009 metais Vilniaus universitete.

Mokslinis vadovas :

doc. habil. dr. Regimantas Pliuškevičius (Matematikos ir informatikos institutas, fiziniai mokslai, matematika - 01P)

Abstract

BDI logics are widely used for agent system description and implementation. Agents are autonomous systems, those acts in some environment and aspire to achieve preassigned goals. Decision making mechanism is the main and the most complicated part of agent systems implementation. Different logics are used as a basis for the decision making. One of such a logics is BDI logic, which express agent via its beliefs, desires and intentions. In this thesis, there are researched sequent calculi for BDI logics.

Known sequent calculi for BDI logics, like sequent calculi for other modal logics, use loop-check technique to get decidability. Inefficient loop-check takes a major part of the resources used for the derivation. For some modal logics, there are known loop-check free sequent calculi or calculi with an efficient loop-check.

In this thesis, there is presented loop-check free sequent calculus for $KD45$ logic, which is the main fragment of the BDI logics. Introduced calculus not only eliminates loop-check, but also simplifies sequent derivation. For the branching time logic (another BDI logic fragment) there is presented sequent calculus with an efficient loop-check.

Obtained results are adapted for creation sequent calculi for monoagent and multiagent BDI logics. Introduced calculi use only restricted loop-check. Moreover, loop-check is totally eliminated for some types of the loops. These results enables to create more efficient agent systems, those are based on the BDI logics.

Acknowledgements

I would like to take the opportunity to thank the people who have supported me. I owe my deepest gratitude to my supervisor Assoc. Prof. Regimantas Pliuškevičius, for his valuable support through the work.

It is an honor for me to thank Assoc. Prof. Stanislovas Leonas Norgėla, who was my supervisor of the previous works. I am indebted to my many of my colleagues from the University to support and encourage me through all my studies. I am grateful to the staff of the Logical section of the Institute of Mathematics and Informatics, for their advise and interesting discussions.

It is a pleasure to thank my parents and my sister, for their love and support. I would like to show my special gratitude to my wife Vilma and my daughter Miglė, for their patience and love.

Contents

List of Figures

Introduction

In this thesis, new loop-check free sequent calculus for $KD45$ logic, sequent calculi with an efficient loop-check for branching time and BDI logics are presented.

Research Area and Problem Relevance

In this thesis, research on the tasks of the artificial intelligence related to agent implementation is presented. Agents are autonomous systems, those acts in some environment and aspire to achieve preassigned goals. Agents main property is an ability to decide that to do autonomous according to the information obtained from the environment. One of the possible solution for autonomous decision making implementation is the modal logic applications.

For different scopes, different modal logics, those uses one or several modalities to express agent, are used. The most popular modal logic used for agents implementation is BDI logic ([52]), which express agent using three modalities: beliefs, desires and intentions; usually, combined together with temporal logic. The popularity of the BDI logic was based on the fact, that it is suitable to define various scopes ([31, 44, 16, 20, 43, 50, 14]) and it is known complete axiomatization for it.

Sequent calculi are recognized as suitable to be the basis for autonomous decision making implementations. There is known sequent calculus for BDI logic ([39]), which is sound and complete. However this calculus uses inefficient (direct) loop-check to get decidability. In recent years, loop-check elimination and efficient loop-check construction are widely researched areas. In this work, analysis of the mentioned sequent calculus for BDI logic is presented and another equivalent calculus, which eliminates some of the loops and uses efficient loop-check technique for the rest of the loops, is introduced.

Research Objectives

Loop-check free sequent calculi and sequent calculi with an efficient loop-check for BDI logics and their fragments.

Aim of the Work

The main aim of the thesis is to construct an efficient sound and complete derivation search systems for BDI logics, based on the sequent calculi, those do not use loop-check or uses only restricted loop-check.

Work Tasks

To reach the aim of this work, these tasks had to be solved:

1. To construct sound and complete loop-check free sequent calculus, or sequent calculus with an efficient loop-check for $KD45$ logic and evaluate its complexity.
2. To construct sound and complete loop-check free sequent calculus, or sequent calculus with an efficient loop-check for branching time logic and evaluate its complexity.
3. To construct sound and complete loop-check free sequent calculus, or sequent calculus with an efficient loop-check for BDI logic and evaluate its complexity.
4. Generalize results to get sound and complete sequent calculus with an efficient loop-check for multiagent BDI logic.

Methods

Invertable, semi-invertable rules, combination of the and-rules together with or-rules and primal sequents were used to get decidability for the new calculi. Analysis of the modal logics, loop-check technique and graph theory were used to proof specific features of the particular fragments of the BDI logics. Special types of sequent histories (marked modal operators, marked sequents and modal operators with special indexes) and proven features were used to construct loop-check free sequent calculi or sequent calculi with an efficient loop-check for the fragments of the BDI logics. N. NIDE and T. Shiro results ([39]) for BDI logics, efficiency results obtained for the fragments of the BDI logics were used to construct sound and complete systems for the BDI logics.

Scientific Novelty and Practical Significance

In this thesis, there are presented new sequent calculi: loop-check free sequent calculus for $KD45$ logic, sequent calculi with an efficient loop-check for branching time and BDI logics.

New approach to the inference tree construction was applied during loop-check free sequent calculus for $KD45$ logic creation. If particular conditions are satisfied, some sequent on the inference tree, irrespective of the fact that sequent is derivable or not by itself, may be treated as non derivable and its derivation are not proceeded (initial sequent derivability will be determined by other branches of the or-rules).

During sequent calculus for branching time logic creation, ground formula of the sequent (it appears in every sequent inside a loop) was introduced and its usage leads to restrictions for the loop-check. Such a ground formula may be also adapted for construction restrictions for the loop-check for other logics.

Multiagent system implementations require efficient decision procedures for BDI logics. One of such an efficient procedure (based on the sequent calculus) is presented in this thesis. Practical application of the sequent calculi for BDI logics is obvious. Sequent calculi for the fragments of the BDI logic are also practically applicable. $KD45$ logic and branching time logic are also used outside the scope of the BDI logics.

Loop-check free sequent calculus for $KD45$ logic, presented in this thesis, is already applied for the prover for the $KD45$ logic. "The Tableau Workbench (TWB)" ([2]) is a framework used for creating provers for different modal logics. Project is lead by well known logician R. Gore (homepage: http://users.rsise.anu.edu.au/ rpg/). Prover for $KD45$ logic is also implemented in this framework. This prover implementation is based on the loop-check free sequent calculus for $KD45$ logic ([1]) presented in this thesis.

Defending Statements

Statements presented for defence:

1. New constructed sequent calculus for $KD45$ logic is a loop-check free calculus which not only eliminates loop-check, but also decrease the complexity of the derivation search itself.

2. New constructed sequent calculus for branching time logic with until operator uses restrictions those decrease complexity of the used loop-check.

3. New constructed sequent calculi for monoagent and multiagent BDI logics use efficient loop-check for detection of the loops of all the types.

Approval of Research Results

Results of the scientific research are publicated in 7 articles. One publication is in the journal included in Scientific Master Journal List (ISI), another publication is included in the international databases under acceptation of the Science Council of Lithuania. Other publications are included in the international refereed journals.

Intermediate results presented in 5 conference and in the seminars organized by the Logical section of the Institute of Mathematics and Informatics.

Introduced sequent calculus for $KD45$ logic was used to create a prover for $KD45$ logic in the "The Tableau Workbench (TWB)" project.

Publications of the Author

The list of author's publications that are related to the thesis is as follows:

- Scientific articles in the periodical journals, those are included in the international databases list accepted by the Science Council of Lithuania:

  1. A. Birštunas, Efficient loop-check for $KD45$ logic, Lithuanian Mathematical Journal, vol 46, No. 1, 2006, pp. 44–53, Springer, New York.

  2. A. Birštunas, PSPACE complexity of modal logic $KD45_n$, Lithuanian Mathematical Journal, vol 48, No. 2, 2008, pp. 174–187, Springer, New York.

- Scientific articles in the international refereed journals:

  3. A. Birštunas, Efficient decision procedure for Belief modality, Lithuanian Mathematical Journal, vol 45, spec. issue, 2005, pp. 321–325.

  4. A. Birštunas, Sequent calculus usage for BDI agent implementation, Lithuanian Mathematical Journal, vol 46, spec. issue, 2006, pp. 232–237.

  5. A. Birštunas, Efficient loop-check for multimodal $KD45_n$ logic, Lithuanian Mathematical Journal, vol 47, spec. issue, 2007, pp. 351–355.

  6. A. Birštunas, Restrictions for loop-check in sequent calculus for temporal logic, Lithuanian Mathematical Journal, LMD works, vol 48-49, 2008, pp. 269–274.

  7. A. Birštunas, Restrictions for loop-check in sequent calculus for temporal logic with until operator, Lithuanian Mathematical Journal, LMD works, vol 50, 2009, pp. 247–252.

Outline of the Thesis

In the introduction, aim of the work, thesis actuality, tasks and main results were presented.

In the first chapter, there are described agents and agent systems. There is presented basic schema of the agents behaviour. Agent system architecture, based on the BDI logics is also described.

The second chapter is used to present main definitions and concepts used through the dissertation. Semantics of the used logic is presented. Loop-check technique, its advantages and disadvantages are introduced.

In the third chapter, there is presented known sequent calculus for $KD45$ logic. Lemmas for loop-check restrictions are proven. There is introduced loop-check free sequent calculus for $KD45$ logic, which uses marked modal operators. Lemmas for calculus complexity are proven.

In the fourth chapter, there is presented research on the branching time logic. Ground sequent formulas are defined and lemmas for loop-check restrictions are proven. There is introduced sequent calculus with an efficient loop-check for branching time logic. Efficiency of the loop-check is obtained by using marked sequents and indexes.

In the fifth chapter, there are presented sequent calculus for monoagent and multiagent BDI logics. Lemmas, showing that restrictions obtained for the separate fragments of the BDI logics may be applied for the BDI logics, are proven. There are presented sequent calculi for monoagent and multiagent BDI logics those use an efficient loop-check.

In the appendix, there is presented and described pseudocode for implementing loop-check free sequent calculus for $KD45$ logic implementation.

Chapter 1

Agents and BDI logic

In this chapter, agents and agents systems are presented. Basic agent behaviour schema and possible approaches for agents implementations discussed. BDI agent model and its implementation issues are described.

1.1 Agents

In this section, agents and their main properties are discussed from the philosophical point. Agents autonomy is the main property, which requires decision making mechanism, the most important and the most complex part of the agent implementation, to be implemented. Several main decision making mechanism implementation techniques are also presented.

Humans always tried to make machines or systems work for their purposes. It is especially applicable in our days when computer systems and machines are used in the most areas of our life. Some of the functions of these systems can be made by themselves, but we still need a human who instructs the system what to do. We can eliminate human work only if systems can rationally decide what to do by themselves. We want to make machines work independently as possible. Such a systems with decision making possibility became rational agents.

The implementation of such an agent based systems are very perspective area of the research works. The hardest and the most important part of the agent implementation is rational decision making.

There are a lot of agents in our life. We know about autopilots in the aircrafts, different electronic equipment in our cars (like ABS - Anti-lock Braking System), about programs those seeks for special information in the internet, about sorting machines, moon vehicles, different robots and a lot of other machines or software programs those are agents. These agents may vary very much. Some of them are very important, some are not. Some are very complex, some are very simple. All of them are agents, because

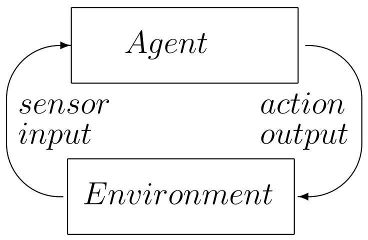


Figure 1.1: Simple agent behaviour schema.
Agent gets information from the environment via input sensors and performs actions those may change environment.

they decide that to do by themselves. Agents performs a lot of work for us every day. A lot of agents help us to do work more easy. Some things are even impossible to do without agents helps (for example, we cannot drive moon vehicle directly from the earth, because signal goes too slow, and moon vehicle should decide by itself how to drive). Therefore, agents are very important in our life and it is the reason why, we can see a great interest on agent research in recent years.

Any agent lives in some environment. Agent should adapt in the changeable environment if it tries to achieve its goals. Agent gets information about current environment state via its sensors (physical sensors or program drivers). Agent may change environment via its actions. According to collected information agent decides what action to perform. It is simple agets behaviour schema (see Figure 1.1). Agent may not know about all existing agents in the environment, but environment may be changed by any agent.

Agent can be defined as entity (machine, software system ...), which can change environment via its actions, can be influenced by the environment, and has the following properties ([52]):

- autonomy,
- proactiveness,
- reactivity, and
- social ability.

The autonomy is agents possibility to operate independently. The proactiveness means that agent acts in a way, which helps him to achieve its goals. The reactivity is agents possibility to change its behaviour if environment changes. The social ability is agents possibility to interact with other self-interested agents to achieve its own goals.

The autonomy property is the most important property of the agent. If we want to construct an agent we have to create some decision making mechanism. It is the core part of the artificial intelligence. This problem may be solved in several different ways.

The simplest way is to use decision trees (or decision tables) ([4]). In such a case, for every possible environment state we define the most suitable action to perform. If agent is very simple, or agent lives in a very restricted environment, then such a solution is suitable, because agent may perform very fast. If environment is complicated decision trees are useless, because sometimes it is even impossible to define all possible environment states. For more complicated environment artificial neural networks (for example [51, 54]) or some mathematical logic may be used.

Artificial neural networks may help agent to decide that to do in very different environments. You can use artificial neural networks for different environments in very similar way. This advantage is one of the main reasons why artificial neural networks are popular. Instead of this, artificial neural network also has disadvantages. Since artificial neural network always does mistakes, agents based on neural networks will perform mistakes too. Mistake probability may be higher or lower. Unfortunately, if environment is complicated, mistake probability is rather high. Instead of this, if mistake probability level is acceptable for constructed agent, then artificial neural networks may be a good solution.

Mathematical logic is treated as rational way of thinking, so, it is not a surprise to use logic for agent implementation. If some mistakes are unallowable (autopilot cannot decide to crash an aircraft even by mistake), then mathematical logic may help. If we implement agent based on the mathematical logic we can assure that some conditions may never be broken. This property of the agent based on the mathematical logic are very important. If we construct an agent based on mathematical logic we do not tell agent what to do, we just tell the rules, those must be followed during agents life. This is why we can assure that special conditions are never broken by an agent.

Agents may be implemented using different kind of the decision making mechanism. Of course, sometimes these mechanisms may be used together to get better results.

There are known a lot of different logics those may be the basis for agent decision making mechanism. These logics differ in application scope, language expressivity, semantics. Logic of knowledge, $KARO$ logic, BDI logic and other are researched and used in different areas.

For example, $KARO$ logic may be described with knowledge, belief, action ($[\alpha]$ 'after performance of α it holds that'), ability and desire modalities ([35, 36]). Logic of knowledge (also known as epistemic logic) uses modal logic $S5$ to represent agents knowledge and logic of belief (also known as doxastic logic) uses modal logic $KD45$ to represent agents beliefs ([26, 53]).

There are lots of reasonable combinations of these modal logics combined with actions, temporal, dynamic logics together with special features (for example [49, 15, 33, 47]). BDI logic itself is some fragment of the $LORA$ logic presented in [52].

BDI logic distinguish from others, because its language is very expressive and it is applicable for very different areas. This is the reason we research agents based on the BDI logics.

Further we will talk only about agents based on the mathematical logic, because this work concentrates on the agents based on the BDI logic.

1.2 BDI Agents

In this section, BDI model for agent implementation is presented. In BDI model, agent is expressed via agents beliefs, desires and intentions. This approach is applicable for different scopes. The main BDI agents implementation schema is discussed. Formal BDI logic usage for BDI agents decision making mechanism is shown.

We concentrate on the agents based on the BDI logic, or simply BDI agents. BDI logics are the most popular logics used for agent implementation ([52]), because of their expressive power and suitability for many applications. BDI logic was introduced by A.S. Rao and M. Georgeff ([44]). There are known a lot of agents implementations based on the BDI logics ([31, 44, 16, 20, 43, 50, 14]). There are even special programming languages created ([13, 46, 17]) for implementing and modeling multi-agent systems, and they are based on formal BDI logic.

In BDI model, every agent can be defined by the set of its actions and plans, and the sets of its beliefs, desires and intentions. Agents belief corresponds the information agent has about the world. Every agent has some vision of the current state of the environment it lives in. There are some facts that are thought to be true by an agent. These facts are agents beliefs. Some agents beliefs may be very stable (for example, moon vehicle beliefs that it cannot go over a rock, and this belief does not change during its life), some beliefs may change frequently (for example moon vehicle may belief that there is no rock in front of him, but this belief may change every time it gets new information from its sensors).

Desire represents state of the world agent wants to be true in an ideal world. Agents desires are its goals. Normally agents desires are set during agents construction, because agent desires describes purpose of the agent. If agents purpose is to clean a room, then it has a desire, that there is no refuse in the room. It is allowed that agent has desires those contradict to each other. For a human it is normal to desire contracting things. For example, human may desire to go to the cinema and to the basketball, but these events may state at the same time. The same logic is also applied for agents.

Finally, intention is such a state of the environment currently agent tries to achieve. Intentions describes that agent plans to do. If agent has an intention, it tries to achieve this intention. Agent does not drop its intention without a good reason. Typically

```
     General control loop for agent implementation
1.  $B$ := $B_0$;    /* $B_0$ are initial beliefs */
2.  $I$ := $I_0$;    /* $I_0$ are initial intentions */
3.  while true do
4.       get next percept $p$;
5.       $B$ := brf($B$, $p$);
6.       $D$ := options($B$, $I$);
7.       $I$ := filter($B$, $D$, $I$);
8.       $\pi$ := plan($B$, $I$);
9.       execute($\pi$);
10.  end while
```

Figure 1.2: General BDI agent implementation scheme.
Here, p denotes all information obtained from the environment, B, D, I - sets of all agents beliefs, desires, intentions respectively, and π denotes some plan chosen by an agent.

agent tries to achieve its intention till it beliefs that intention is already achieved, or it beliefs, that intention cannot be achieved in current situation. Of course, current agents intentions cannot contradict to each over, otherwise, agent will act irrationally.

BDI agent, like all agents, gets information from the environment, chooses action and performs it. Action selection process is described using belief, desire and intention concepts. BDI model was introduced by A.S. Rao and M. Georgeff ([44]). In BDI model, agents beliefs, desires and intentions are defined as BDI logic formulas. Most researches agree that it is reasonable to treat operator belief as modality of modal logic $KD45$ and agents desire and intention to treat as modalities of modal logic KD ([52]). So, we have BDI logic used to describe and implement rational agents. Agent must take into account its beliefs, desires and intentions during action selection. Action selection is the most important part of the agent implementation, since it ensures the main property - autonomy. In [52], Wooldridge suggest to separate action selection into 3 functions (options, filter, plan) as it is shown in Figure 1.2.

According to general scheme (see Figure 1.2), agent repeats defined operations step by step. Agent updates its beliefs about the world and current state according to the information obtained from its sensors. After, agent decides what it wants to achieve (chooses desires). Further, agent has to select an intention used to achieve chosen goals (desires). When intentions are chosen, agent has to decide how such an intention can be achieved (chooses a plan). So, function brf is used to renew current agents beliefs according to information obtained from the environment via agents sensors. In function options, agent chooses current desires according to new beliefs and last intentions. In function filter, agent selects the best intention according to new beliefs, chosen desires and last intentions. Finally, agent must select the most suitable (in current situation) plan to execute (function plan). Every plan consists of the ordered list of actions - $\pi = \alpha_1, \alpha_2, \ldots, \alpha_k$ (α_i is an action). In other words, plan is some simple instructions for an agent. In function execute, agent just performs all actions described in the selected plan π.

In [52], there are presented some more complex schemes for agent implementation. These agent implementation schemas take into account some commitment strategies, those helps agent to drop a plan during execution, if plan became irrational. However, all these schemas bases on functions brf, options, filter, plan.

There are works concentrating on formal description of agent (not only BDI agents) for some scopes using formal logic ([20, 14]). In [44, 46] there was mentioned existence of the gap between BDI agent implementations and the formal logics. Solution for this problem may be found in [53, 44, 32, 6], where main ideas of the formal logic usage for agents action selection are proposed. In [16], there is given an example of agent implementation which uses formal logic for action selection. There are even organized contests on agent implementations (see the first contest on Multi-Agent systems organized during CLIMA-VI[1] in [18]).

If we deal with BDI agents, sets B, D, I are sets of the BDI logic formulas. However, there is no evident places in the general BDI agent implementation scheme, where formal logic method is applied. In general scheme only function's implementations are not defined. Wooldridge left function's implementation issues opened. One can implement them in the way he wants. Since functions deal with sets of BDI logic formulas, it is wise to use some formal logic methods to implement these functions. If we implement all functions without any logic method usage at all, we will construct an agent which is not an agent really based on the BDI logic. In such a case, logic is used only as notation and nothing more (and BDI logic formulas usage has no advantages).

Formal logic methods may be applied in functions options, filter, plan implementation. There are different approaches how formal logic methods can be applied and in which functions implementation formal logic methods are suitable. In works [53, 44], there are used invocation conditions for actions. If invocation condition is satisfied then action must be chosen for execution. Similar approach is used in [32] where 'if-then' statements are used. In [16], there is shown how formal logic can be applied for checking such an invocation conditions. Action choosing is performed in function plan. Result of this function is a plan (ordered list of actions) for execution. Main idea of this function implementation is that plan must be executed only if some invocation condition is satisfied. In such an implementation we have finite list of possible plans and every plan is associated with some condition formula. All these pairs (formula and plan) are ordered in some priority queue. Similar approach is used in [6], where suggested to use formal logic methods for implementing functions filter and plan.

[1]CLIMA-VI - VI International Workshop on Computational Logic in Multi-agent Systems.

Chapter 2

Sequent Calculus and Loop-check

In this chapter, we introduce main definitions used in other chapters. Used language semantics and main issues related to the sequent calculus are presented. Therefore, in this thesis, efficient loop-check for sequent calculus is researched. In the last section, loop-check technique and its appliance are discussed.

2.1 Language

In this section, we define formulas, used modal operators and their variations. We define subformula, extended formula, formula length and other used terms and notations.

Definition 2.1.1 Suppose that P is a set of propositional symbols.

- if $p \in P$, then p is formula.
- if ϕ, ψ are formulas, then
  $\neg\phi$, $\phi \vee \psi$, $\phi \& \psi$,
  $\Box\phi$, $\mathbf{B}\phi$, $\mathbf{D}\phi$, $\mathbf{I}\phi$,
  $\circ\phi$, $A(\phi \cup \psi)$, $E(\phi \cup \psi)$,
  $\Box^*\phi$, $\mathbf{B}^*\phi$
  are also formulas.
- if ϕ, ψ are formulas and i is agent's index, then
  $\mathbf{B}_i\phi$, $\mathbf{D}_i\phi$, $\mathbf{I}_i\phi$, $\mathbf{B}_i^*\phi$
  are also formulas.
- if ϕ, ψ are formulas and α, β are special indexes, then
  $A_\beta^\alpha(\phi \cup \psi)$, $E_\beta^\alpha(\phi \cup \psi)$
  are also formulas.

Formulas $\phi \supset \psi$, $\phi \Leftrightarrow \psi$, are abbreviations of
$\neg\phi \vee \psi$, $(\phi \supset \psi) \wedge (\psi \supset \phi)$, respectively.

Definition 2.1.2 If F is formula of the shape
$\Box\phi$, $\mathbf{B}\phi$, $\mathbf{D}\phi$, $\mathbf{I}\phi$,
$\circ\phi$, $A(\phi \cup \psi)$, $E(\phi \cup \psi)$,
$\Box^*\phi$, $\mathbf{B}^*\phi$,
$\mathbf{B}_i\phi$, $\mathbf{D}_i\phi$, $\mathbf{I}_i\phi$, $\mathbf{B}_i^*\phi$,
$A_\beta^\alpha(\phi \cup \psi)$, $E_\beta^\alpha(\phi \cup \psi)$
then F is modalized formula.

Definition 2.1.3 If F is formula of the shape
$\Box^*\phi$, $\mathbf{B}^*\phi$, $\mathbf{B}_i^*\phi$
then F is marked modalized formula.

For the transparency, we use formulas notation as follows (through all the dissertation):

- with small Greek letter ($\phi, \psi, \gamma, \kappa, \phi_1, \phi_2, \ldots, \psi_1, \psi_2, \ldots$) we denote general formula;
- with Θ we denote one (possibly empty) formula;
- with Σ, Π we denote finite (may be empty) sets of propositional variables (usually, $\Sigma \cap \Pi = \emptyset$).
- with capital Greek letter ($\Gamma, \Gamma_1, \Gamma_2, \ldots, \Delta, \Delta_1, \Delta_2, \ldots$) we denote finite (may be empty) set of the formulas.
- with modal operator followed by capital Greek letter ($\Omega\Gamma$, where $\Omega \in \{\circ, \Box, \mathbf{B}, \mathbf{B}^*, \mathbf{D}, \mathbf{I}, \mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_1^*, \mathbf{B}_2^*, \ldots, \mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{I}_1, \mathbf{I}_2, \ldots\}$) we denote finite (may be empty) set of the formulas of the shape $\Omega\phi$.

Definition 2.1.4 If F is formula of the shape
$A(\phi \cup \psi)$, $E(\phi \cup \psi)$, $A_\beta^\alpha(\phi \cup \psi)$, $E_\beta^\alpha(\phi \cup \psi)$
then F is *Æ* formula.

Definition 2.1.5 If F is formula of the shape
$A(\phi \cup \psi)$, $\circ A(\phi \cup \psi)$, $E(\phi \cup \psi)$, $\neg E(\phi \cup \psi)$, $\circ\neg E(\phi \cup \psi)$, $\neg \circ \neg E(\phi \cup \psi)$,
$A_\beta^\alpha(\phi \cup \psi)$, $\circ A_\beta^\alpha(\phi \cup \psi)$, $E_\beta^\alpha(\phi \cup \psi)$, $\neg E_\beta^\alpha(\phi \cup \psi)$, $\circ\neg E_\beta^\alpha(\phi \cup \psi)$, $\neg \circ \neg E_\beta^\alpha(\phi \cup \psi)$
then F is extended *Æ* formula.

Definition 2.1.6 We define formulas F extended set $Ext(F)$ as follows:

- $Ext(A(\phi \cup \psi)) = Ext(\circ A(\phi \cup \psi)) = \{A(\phi \cup \psi), \circ A(\phi \cup \psi)\}$,
- $Ext(E(\phi \cup \psi)) = Ext(\neg E(\phi \cup \psi)) = Ext(\circ\neg E(\phi \cup \psi)) =$
  $Ext(\neg \circ \neg E(\phi \cup \psi)) = \{E(\phi \cup \psi), \neg E(\phi \cup \psi), \circ\neg E(\phi \cup \psi), \neg \circ \neg E(\phi \cup \psi)\}$,
- $Ext(A^\alpha_\beta(\phi \cup \psi)) = Ext(\circ A^\alpha_\beta(\phi \cup \psi)) = \big\{A^\alpha_\beta(\phi \cup \psi), \circ A^\alpha_\beta(\phi \cup \psi)\big\}$,
- $Ext(E^\alpha_\beta(\phi \cup \psi)) = Ext(\neg E^\alpha_\beta(\phi \cup \psi)) =$
  $Ext(\circ\neg E^\alpha_\beta(\phi \cup \psi)) = Ext(\neg \circ \neg E^\alpha_\beta(\phi \cup \psi)) =$
  $\big\{E^\alpha_\beta(\phi \cup \psi), \neg E^\alpha_\beta(\phi \cup \psi), \circ\neg E^\alpha_\beta(\phi \cup \psi), \neg \circ \neg E^\alpha_\beta(\phi \cup \psi)\big\}$,
- otherwise $Ext(F) = \emptyset$.

Definition 2.1.7 If F is formula of the shape
ϕ, $\neg\phi$, $\phi \vee \psi$, $\phi\&\psi$,
$\Box\phi$, $\mathbf{B}\phi$, $\mathbf{D}\phi$, $\mathbf{I}\phi$,
$\circ\phi$, $A(\phi \cup \psi)$, $E(\phi \cup \psi)$,
$\Box^*\phi$, $\mathbf{B}^*\phi$,
$\mathbf{B}_i\phi$, $\mathbf{D}_i\phi$, $\mathbf{I}_i\phi$, $\mathbf{B}^*_i\phi$,
$A^\alpha_\beta(\phi \cup \psi)$, $E^\alpha_\beta(\phi \cup \psi)$
then ϕ and ψ are subformulas of F.

If $\phi \in Ext(F)$, then ϕ is also subformula of F.

If ϕ is subformula of ψ, and ψ is subformula of F, then ϕ is subformula of F.

If ϕ is subformula of F we write $\phi \subseteq_{sf} F$.

Definition 2.1.8 We define length function len for formulas as follows:

- $len(\phi) = 1$, if ϕ is propositional variable,
- $len(\phi \vee \psi) = len(\phi\&\psi) = len(\phi) + len(\psi) + 1$,
- $len(\neg\phi) = len(\phi) + 1$,
  $len(\Box\phi) = len(\mathbf{B}\phi) = len(\mathbf{D}\phi) = len(\mathbf{I}\phi) = len(\phi) + 1$,
  $len(\circ\phi) = len(\phi) + 1$,
  $len(\Box^*\phi) = len(\mathbf{B}^*\phi) = len(\phi) + 1$,
  $len(\mathbf{B}_i\phi) = len(\mathbf{D}_i\phi) = len(\mathbf{I}_i\phi) = len(\mathbf{B}^*_i\phi) = len(\phi) + 1$,
- $len(A(\phi \cup \psi)) = len(E(\phi \cup \psi)) = len(A^\alpha_\beta(\phi \cup \psi)) = len(E^\alpha_\beta(\phi \cup \psi)) =$
  $len(\phi) + len(\psi) + 1$.

Definition 2.1.9 Formula ϕ is proper subformula of ψ,
if $\phi \subseteq_{sf} \psi$ and $len(\phi) < len(\psi)$.

If ϕ is proper subformula of ψ we write $\phi \subset_{sf} \psi$.

Definition 2.1.10 We define formulas modality depth function as follows:

- $depth(\phi) = 0$, if ϕ is propositional variable,
- $depth(\phi \vee \psi) = depth(\phi \& \psi) = max(depth(\phi), depth(\psi))$,
- $depth(\neg\phi) = depth(\phi)$,
- $depth(\Box\phi) = depth(\mathbf{B}\phi) = depth(\mathbf{D}\phi) = depth(\mathbf{I}\phi) = depth(\phi) + 1$,
  $depth(\Box^*\phi) = depth(\mathbf{B}^*\phi) = depth(\phi) + 1$,
  $depth(\mathbf{B}_i\phi) = depth(\mathbf{D}_i\phi) = depth(\mathbf{I}_i\phi) = depth(\mathbf{B}_i^*\phi) = depth(\phi) + 1$,
- $depth(\circ\phi) = depth(\phi) + 1$, if ϕ is not an extended *Æ* formula,
  $depth(\circ\phi) = depth(\phi)$, if ϕ is an extended *Æ* formula,
- $depth(A(\phi \cup \psi)) = depth(E(\phi \cup \psi)) = depth(A^\alpha_\beta(\phi \cup \psi)) =$
  $depth(E^\alpha_\beta(\phi \cup \psi)) = max(depth(\phi), depth(\psi)) + 1$.

We have to mention, that formula $A(\phi \cup \psi)$ and $\circ A(\phi \cup \psi)$ has the same modality depth. It is done so, since every extended *Æ* formula $F' \in Ext(F)$ must have the same modality depth. In other words, until operator combined with $\circ$ (if together compose extended *Æ* formula) is treated as one modality.

2.2 Sequent Calculus

In this section, sequent and main sequent calculus rules are defined. Sequent calculus and-rules and or-rules are described. Inference tree, derivation tree and other definitions related to the sequent calculus are presented.

Definition 2.2.1 If $F_1, F_2, \ldots, F_n, G_1, G_2, \ldots G_m$ are formulas and $n + m > 0$, then $F_1, F_2, \ldots, F_n \rightarrow G_1, G_2, \ldots G_m$ is a sequent.

Definition 2.2.2 We define length function len for the sequents as follows:
If $S = F_1, F_2, \ldots, F_n \rightarrow G_1, G_2, \ldots G_m$ is a sequent then
$len(S) = len(F_1) + len(F_2) + \ldots + len(F_n) + len(G_1) + len(G_2) + \ldots + len(G_m)$.

So, length of the sequent is the sum of sequent formulas lengths.

Definition 2.2.3 We define modality depth function $depth$ for the sequents as follows:
If $S = F_1, F_2, \ldots, F_n \rightarrow G_1, G_2, \ldots G_m$ is a sequent then
$depth(S) = max(depth(F_1), \ldots, depth(F_n), depth(G_1), \ldots, depth(G_m))$.

So, depth of the sequent is the maximum of sequent formulas depths.

Definition 2.2.4 Sequent calculus logical rules are:

$$\frac{\phi, \Gamma \to \Delta \quad \psi, \Gamma \to \Delta}{\phi \vee \psi, \Gamma \to \Delta}\ (\vee L) \qquad \frac{\phi, \psi, \Gamma \to \Delta}{\phi \& \psi, \Gamma \to \Delta}\ (\& L) \qquad \frac{\Gamma \to \phi, \Delta}{\neg\phi, \Gamma \to \Delta}\ (\neg L)$$

$$\frac{\Gamma \to \phi, \Delta \quad \Gamma \to \psi, \Delta}{\Gamma \to \phi \& \psi, \Delta}\ (\& R) \qquad \frac{\Gamma \to \phi, \psi, \Delta}{\Gamma \to \phi \vee \psi, \Delta}\ (\vee R) \qquad \frac{\phi, \Gamma \to \Delta}{\Gamma \to \neg\phi, \Delta}\ (\neg R)$$

Definition 2.2.5 Sequent calculus rule $(Weak)$ is:

$$\frac{\Gamma \to \Delta}{\Gamma, \Gamma' \to \Delta, \Delta'}\ (Weak)$$

Definition 2.2.6 Tree is a sequent S inference tree in the sequent calculus C (or just sequent S tree in the calculus C) if the following conditions are satisfied:

- every tree node contain some sequent,
- in the root node there is a sequent S,
- if node N contains sequent S and it has children $N_1, N_2, \ldots, N_k$ those contain sequents $S_1, S_2, \ldots, S_k$ respectively, then sequent S is a conclusion and sequents $S_1, S_2, \ldots, S_k$ are all premises of some sequent calculus C rule application.

Definition 2.2.7 Sequent S inference tree in the sequent calculus C is sequent S derivation tree in the sequent calculus C if every tree leaf contains some axiom of the sequent calculus C.

Definition 2.2.8 We define sequent S inference tree T height function $height(T)$ to be the function which calculates the maximum count of the sequents placed on one tree branch.

Example 2.2.1 Suppose we have an initial sequent
$S = \phi_1 \vee (\phi_2 \& \neg\phi_3) \to (\neg\phi_2 \& \neg\phi_3) \vee \phi_1$,
then sequent S inference tree may be the following:

$$\dfrac{\dfrac{\overset{\oplus}{\phi_1 \to \neg\phi_2 \& \neg\phi_3, \phi_1}}{\phi_1 \to (\neg\phi_2 \& \neg\phi_3) \vee \phi_1}\ (\vee R) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\phi_2 \to \phi_3, \neg\phi_2 \& \neg\phi_3, \phi_1}{\phi_2, \neg\phi_3 \to \neg\phi_2 \& \neg\phi_3, \phi_1}\ (\neg L)}{\phi_2 \& \neg\phi_3 \to \neg\phi_2 \& \neg\phi_3, \phi_1}\ (\& L)}{\phi_2 \& \neg\phi_3 \to (\neg\phi_2 \& \neg\phi_3) \vee \phi_1}\ (\vee R)}{\phi_1 \vee (\phi_2 \& \neg\phi_3) \to (\neg\phi_2 \& \neg\phi_3) \vee \phi_1}\ (\vee L)$$

This inference tree has height 5. Inference tree is an intermediate tree obtained during sequent derivation. This tree may not end with axioms. Therefore, if we have inference tree, in general, we cannot say whether an initial sequent S is derivable or not. Only if every sequent calculus rule is invertable and some inference tree leaf contains final non axiom sequent, we know that an initial sequent S is non derivable.

Example 2.2.2 Suppose we have an initial sequent

$S = \phi_1 \vee (\phi_2 \& \neg\phi_3) \to (\phi_2 \& \neg\phi_3) \vee \phi_1$,

then sequent S derivation tree may be the following:

$$\dfrac{\dfrac{\overset{\oplus}{\phi_1 \to \neg\phi_2 \& \neg\phi_3, \phi_1}}{\phi_1 \to (\neg\phi_2 \& \neg\phi_3) \vee \phi_1}\,(\vee R) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi_2 \to \phi_3, \phi_2, \phi_1} \quad \dfrac{\overset{\oplus}{\phi_2, \phi_3 \to \phi_3, \phi_1}}{\phi_2 \to \phi_3, \neg\phi_3, \phi_1}\,(\neg R)}{\phi_2 \to \phi_3, \phi_2 \& \neg\phi_3, \phi_1}\,(\& R)}{\phi_2, \neg\phi_3 \to \phi_2 \& \neg\phi_3, \phi_1}\,(\neg L)}{\phi_2, \neg\phi_3 \to (\phi_2 \& \neg\phi_3) \vee \phi_1}\,(\vee R)}{\phi_2 \& \neg\phi_3 \to (\phi_2 \& \neg\phi_3) \vee \phi_1}\,(\& L)}{\phi_1 \vee (\phi_2 \& \neg\phi_3) \to (\phi_2 \& \neg\phi_3) \vee \phi_1}\,(\vee L)$$

This derivations tree has height 7. It is a derivation tree, because it is inference tree and every leaf contains an axiom. If we found a derivation, we know that an initial sequent S is derivable in the given sequent calculus.

Every sequent calculus rule (and-rule) satisfies: if all premises of the rule are derivable, then conclusion of the rule is derivable.

Definition 2.2.9 Sequent calculus rule is invertable if the following condition is satisfied: if conclusion of the rule is derivable, then all premises of the rule are derivable.

Definition 2.2.10 Sequent calculus rule is semi-invertable if the following condition is satisfied: if conclusion of the rule is derivable, then at least one of the premises of the rule is derivable.

It is obviously, that every invertable rule is also semi-invertable rule. We can see that all logical rules are invertable. Rule $(Weak)$ is neither invertable nor semi-invertable.

Definition 2.2.11 If sequents $S_1, S_2, \ldots$ are premises of some calculus rule R application and S is a conclusion, then we say that sequents $S_1, S_2, \ldots$ are children of the sequent S (S will be the father of $S_1, S_2, \ldots$) in the tree.

Definition 2.2.12 We define sequent S as a leaf sequent in the inference tree (or simply a leaf) if sequent S do not have children.

Definition 2.2.13 We say that sequent S is an ancestor of the sequent S' in the inference tree, if there exists sequence of the sequents $S_1 = S, S_2, S_3, S_4 \ldots, S_{n-1}, S_n = S'$ that for every $i = 1, 2, \ldots (n-1)$, sequent S_i is a conclusion and sequent S_{i+1} is a premise of some rule application.

Definition 2.2.14 We say, that sequent S is a final in the sequent calculus C if it is not an axiom of the sequent calculus C and there is no sequent calculus C rule, which can be applied for the sequent S.

Since any rule cannot be applied for some final sequent, final sequent is always placed in the leaf of the inference tree. Since it is not an axiom, final sequent is non derivable.

Definition 2.2.15 We say, that we have a weak loop $S \rightsquigarrow S'$ in the sequent tree if the following conditions are satisfied:

- sequent S is an ancestor of the sequent S',
- S' may be obtained from S by the rule $(Weak)$ application: $\frac{S}{S'} \quad (Weak)$.

If $S \rightsquigarrow S'$ is a weak loop in the sequent tree, when S' is called weak loop-ending sequent, and S is called weak loop-starting sequent.

Definition 2.2.16 We say, that we have a loop $S \rightsquigarrow S'$ in the sequent tree if the following conditions are satisfied:

- sequent S is an ancestor of the sequent S',
- sequents S and S' contains the same formulas on the right sides of the $\rightarrow$, and on the left sides of the $\rightarrow$.

If $S \rightsquigarrow S'$ is a loop in the sequent tree, when S' is called loop-ending sequent, and S is called loop-starting sequent.

It is obviously, that every loop $S \rightsquigarrow S'$ is also a weak loop $S \rightsquigarrow S'$.

Definition 2.2.17 Sequent calculus rule is called and-rule if rule conclusion is derivable if all rule premises are derivable.

Definition 2.2.18 Sequent calculus rule is called or-rule if rule conclusion is derivable if at least one of the rule premises is derivable.

Logical rules, rule $Weak$ and all common sequent calculus rules are and-rules. In this thesis, some calculi uses and-rules together with or-rules. As an example rule $(\Box^W)$ (see Definition 3.2.5.) is or-rule.

In the notation, to separate and-rule premises we use empty space:

$$\frac{S_1 \quad S_2}{S'} \quad (and - rule)$$

In the notation, to separate or-rule premises we use special symbol $||$:

$$\frac{S_1 \quad || \quad S_2}{S'} \quad (or - rule)$$

Definition 2.2.19 In the case when only and-rules are used in the sequent calculus, we say that sequent inference tree is a tree with and-branches.

In the case when or-rules are used in the sequent calculus, we say that sequent inference tree is a tree with or-branches.

When we have or-branches beside and-branches, the sequent can be derivable even if not all tree leaves are derivable (i.e. axioms). This is the reason why, we have to specify derivation tree definition for the sequent calculus having or-rules.

Definition 2.2.20 Sequent S inference tree T in the sequent calculus C (having or-rules) is sequent S derivation tree in the sequent calculus C, if we can transform tree T into the tree T', that the following conditions are satisfied:

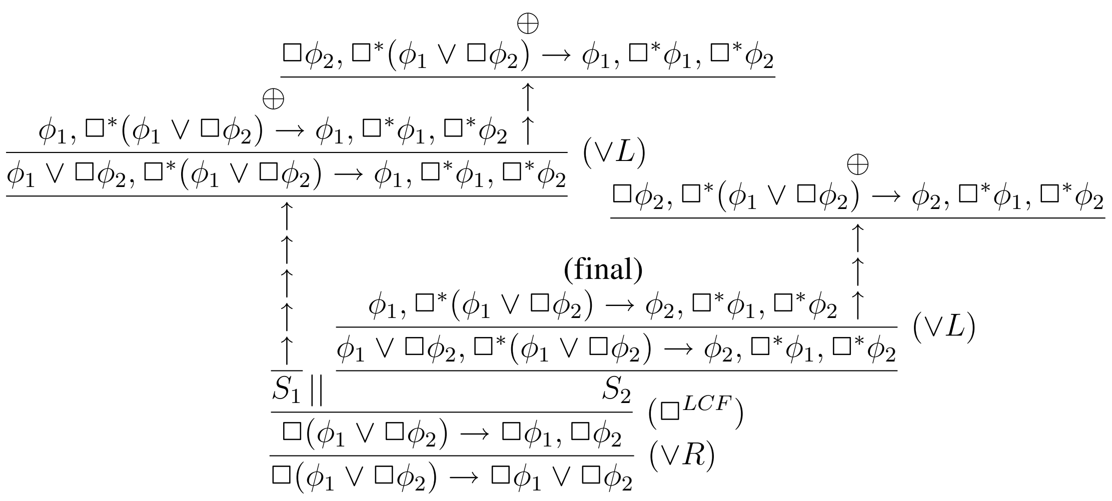
- tree T' have the same sequent S in the root;
- if we have some or-rule R application in the tree T, when only one premise of the rule R application is left in the tree T' (all other premises are deleted with all their subtrees);
- if we have some and-rule R application in the tree T, when it is left in the tree T' unchanged unless whole rule application was deleted according to previous item.
- every tree T' leaf contains some axiom of the sequent calculus C.

For both cases (calculus without or-rules and calculus with or-rules), if we constructed sequent S derivation tree in the sequent calculus C, we show that sequent S is derivable in the sequent calculus C.

Example 2.2.3 Suppose we have an initial sequent S in $KD45$ logic:
$S = \Box(\phi_1 \vee \Box\phi_2) \rightarrow \Box\phi_1 \vee \Box\phi_2$.

In this thesis, we introduce loop-check free sequent calculus $KD45_{lcf}$ (see Definition 3.2.9) which uses or-rule $(\Box^{LCF})$ (see Definition 3.2.8).

Sequent S derivation tree T in the sequent calculus $KD45_{lcf}$ may be the following:



Rules $(\vee R)$, $(\vee L)$ are and-rules, rule $(\Box^{LCF})$ is or-rule. In this case, we have 2 or-branches: S_1 and S_2. S_1 is derivable, because all its leaves ends with axioms. S_2 is treated as non derivable, because one of the leafs contains final sequent, which is not an axiom.

Inference tree T is a derivation tree in the sequent calculus $KD45_{lcf}$, because we can get a subtree T', which satisfies all the conditions described in the Definition 2.2.20. Subtree T' is:

$$\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi_1, \Box^*(\phi_1 \vee \Box\phi_2) \to \phi_1, \Box^*\phi_1, \Box^*\phi_2} \quad \overset{\oplus}{\Box\phi_2, \Box^*(\phi_1 \vee \Box\phi_2) \to \phi_1, \Box^*\phi_1, \Box^*\phi_2}}{\phi_1 \vee \Box\phi_2, \Box^*(\phi_1 \vee \Box\phi_2) \to \phi_1, \Box^*\phi_1, \Box^*\phi_2}\,(\vee L)}{\Box(\phi_1 \vee \Box\phi_2) \to \Box\phi_1, \Box\phi_2}\,(\Box^{LCF})}{\Box(\phi_1 \vee \Box\phi_2) \to \Box\phi_1 \vee \Box\phi_2}\,(\vee R)$$

Subtree T' is obtained from the tree T by removing or-branch S_2. Subtree T' contains only axioms in the leaves. As it is argued in the Section 2.4, or-rules is used instead of the backtracking and obtained tree T' is a derivation tree if we use backtracking instead of the or-rules.

It is worth to mention, that or-branch S_2 derivation is optional. Suppose, that tree T_1 is obtained from the tree T by removing all the sequents above the sequent
$S_2 = \Box(\phi_1 \vee \Box\phi_2) \to \Box\phi_1, \Box\phi_2$ (sequent S_2 derivation not finished).

Tree T_1 is still an inference tree in the sequent calculus $KD45_{lcf}$, because there is no requirements to derive all branches in the Definition 2.2.6. Tree T_1 is also derivation tree in the sequent calculus $KD45_{lcf}$, since it satisfies all the conditions placed in the Definition 2.2.20 (the same subtree T' satisfies definition).

When sequent calculus contains only and-rules, sequent is derivable if all leaves contain some axioms of the sequent calculus. If sequent contains and-rules together with or-rules are used, then every inference tree is some and-or tree. We may treat every leaf containing an axiom as derivable and all other inference tree leaves as non derivable. If all premises of some and-rule application are derivable, then rule conclusion must be treated as derivable; otherwise, rule conclusion must be treated as non derivable. If at least one premise of some or-rule application is derivable, then rule conclusion must be treated as derivable; otherwise, rule conclusion must be treated as non derivable. If we proceed the same arguments for every rule application till we reach the root, we determine whether an initial sequent is derivable or not.

2.3 Semantics

In this section, we present semantics of used modal operators and researched logic. For this purpose, Kripke 'possible worlds' semantics ([24]) is used. In this thesis, several logics are researched, but all of them are some fragments of the multiagent BDI logic. Therefore, semantics only for multiagent BDI logic is presented.

In this thesis, we concentrate on the multiagent BDI logic which is based on 3 modalities: Belief, Desire and Intend, and temporal logic. Belief is modality of the modal logic $KD45$, Desire and Intend are modalities of the modal logic KD ([52]). For Belief modality we use modal operator $\mathbf{B}$ (or $\mathbf{B}_i$ if we have multiple agents), for Desire modality we use modal operator $\mathbf{D}$ (or $\mathbf{D}_i$) and for Intend modality we use modal operator $\mathbf{I}$ (or $\mathbf{I}_i$). In some calculus, for Belief modality we will use marked modal operator $\mathbf{B}^*$ (or $\mathbf{B}^*_i$) besides ordinary $\mathbf{B}$ (or $\mathbf{B}_i$). Modal operator $\mathbf{B}^*$, $\mathbf{B}^*_i$ semantics are the same. We write just modality $\Box$ if its type (Belief, Desire, Intend) is known from the current context.

We research multi-agent BDI logic which has n agents. Agents count n is not known in advance but finite number.

We use branching-time temporal logic with until operator. So, modality $\circ$ is used to represent 'in every next time', modality A is used to represent 'in all futures until', modality E is used to represent 'in some future until'. In some calculus, we will use temporal operators with indexes (A^{α}_{β}, E^{α}_{β}), those semantics do not differ from A, E respectively.

We use Kripke 'possible worlds' semantics to describe multi-agent BDI logic. We choose a set of possible worlds W. For every world $w \in W$ we define a set of time states St_w and binary relation R_w which represents 'next time' relation. For every agent i we define 3 accessability relations $\mathcal{B}_i, \mathcal{D}_i, \mathcal{I}_i$ for Belief, Desire and Intend modalities. Finally, for every world $w \in W$ and time state $s \in St_w$ we define a true-value assignment $L(w,s) \subseteq P$ which describes 'what is valid in a given world state'.

Belief modality is $KD45$ logic modality, Desire and Intend modalities are KD logic modalities, therefore we have the following formal definitions:

Definition 2.3.1 We define multi-agent BDI logic structure as a tuple
$M =< W, \{St_w : w \in W\}, \{R_w : w \in W\}, \mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{I}_1, \ldots, \mathcal{I}_n, L >$,
where: $W \neq \emptyset$ is possible world set,
$St_w \neq \emptyset$ is set of time states in the world $w \in W$,
$R_w \subset St_w \times St_w$ is serial binary relation for time in the world $w \in W$,
L is true-value assignment $L(w,s) \subseteq P$ ($w \subset W$, $s \in St_w$),
$\mathcal{B}_i, \mathcal{D}_i, \mathcal{I}_i \subset W \times \bigcup_{w \in W} St_w \times W$ are accessability ternary relations for Belief, Desire, Intend modalities those satisfies the following conditions:

- (B-D) for each world $w \in W$, there exists $w' \in W$ that $(w, s, w') \in \mathcal{B}_i$;
- (D-D) for each world $w \in W$, there exists $w' \in W$ that $(w, s, w') \in \mathcal{D}_i$;
- (I-D) for each world $w \in W$, there exists $w' \in W$ that $(w, s, w') \in \mathcal{I}_i$;

- (B-4) for each worlds $w, w', w'' \in W$, if $(w, s, w') \in \mathcal{B}_i$ and $(w', s, w'') \in \mathcal{B}_i$, then $(w, s, w'') \in \mathcal{B}_i$;
- (B-5) for each worlds $w, w', w'' \in W$, if $(w', s, w) \in \mathcal{B}_i$ and $(w', s, w'') \in \mathcal{B}_i$, then $(w, s, w'') \in \mathcal{B}_i$.

Definition 2.3.2 Infinite sequence of time states $s_0, s_1, s_2, \ldots$ is called (state) path on world $w \in W$ if $s_i \in St_w$ and $(s_i, s_{i+1}) \in R_w$ for every $i = 0, 1, 2, \ldots$.

Definition 2.3.3 Formula ϕ is true in a BDI-structure M, world $w \in W$ and state $s \in St_w$ ($(M, w, s) \models \phi$) if the following conditions are satisfied:

- if ϕ is primitive preposition ($\phi \in P$), then $(M, w, s) \models \phi$ iff $\phi \in L(w, s)$.
- $(M, w, s) \models \neg\phi$ iff $(M, w, s) \not\models \phi$.
- $(M, w, s) \models \phi \vee \psi$ iff $(M, w, s) \models \phi$ or $(M, w, s) \models \psi$.
- $(M, w, s) \models \phi \& \psi$ iff $(M, w, s) \models \phi$ and $(M, w, s) \models \psi$.
- $(M, w, s) \models \mathbf{B}_i(\phi)$ iff for all w', satisfying $(w, s, w') \in \mathcal{B}_i$, $(M, w', s) \models \phi$.
- $(M, w, s) \models \mathbf{D}_i(\phi)$ iff for all w', satisfying $(w, s, w') \in \mathcal{D}_i$, $(M, w', s) \models \phi$.
- $(M, w, s) \models \mathbf{I}_i(\phi)$ iff for all w', satisfying $(w, s, w') \in \mathcal{I}_i$, $(M, w', s) \models \phi$.
- $(M, w, s) \models \circ(\phi)$ iff for all state s', satisfying $(s, s') \in R_w$, $(M, w, s') \models \phi$.
- $(M, w, s_0) \models A(\phi \cup \psi)$ iff every path $s_0, s_1, \ldots, s_k, \ldots$ on world w satisfies: there exists k, that $(M, w, s_i) \models \phi$ for every $0 \leq i < k$ and $(M, w, s_k) \models \psi$.
- $(M, w, s_0) \models E(\phi \cup \psi)$ iff at least one path $s_0, s_1, \ldots, s_k, \ldots$ on world w satisfies: there exists k, that $(M, w, s_i) \models \phi$ for every $0 \leq i < k$ and $(M, w, s_k) \models \psi$.

Definition 2.3.4 Formula ϕ is valid in BDI logic if $(M, w, s) \models \phi$ for every BDI-structure M, every world $w \in W$ and every state $s \in St_w$.

All other logics, presented in the thesis, are some fragments of the multi-agent BDI logic. So, we do not define separate semantics for them.

Example 2.3.1 We present some simple examples how some agents imagination about the environment may be expressed using BDI logic.

Suppose that ϕ - is true if the lamp is lighting,
ψ_1 - is true if the switch 1 is on,
ψ_2 - is true if the switch 2 is on.

Then formula: $\mathbf{B}_1(\phi\&\psi_1)\&\mathbf{D}_1(\neg\phi)$ means, that "agent 1 beliefs that the lamp is lighting and the switch 1 is on, and agent 1 desires that the lamp do not light".

Formula: $\mathbf{B}_1(\neg\psi_1 \to \neg\phi)$ means, that "agent 1 beliefs, that if the switch 1 is off, then the lamp do not light".

Formula $\mathbf{B}_2(\neg\psi_1\&\neg\psi_2)\&\mathbf{D}_2(\psi_1 \vee \psi_2)$ means, that "agent 2 beliefs, that both switches are off, and agent 2 desires, that the switch 1 is on or the switch 2 is on".

Formula $\mathbf{B}_1(\mathbf{B}_2\psi_2 \to \neg\mathbf{I}_2\psi_1)$ means, that "agent 1 beliefs, that if agent 2 beliefs, that the switch 2 is on, then agent 2 do not intend to turn the switch 1 on".

2.4 Loop-check and Backtracking

In this section, loop-check technique and its necessity for decision procedure are discussed. There are some main properties of the sequent calculus, those enables us to construct efficient decision procedure based on the given sequent calculus. These properties are also explained in this section. Sequents with histories and history variations are presented. Backtracking and its elimination is also described.

The objective of this thesis is an efficient sequent calculus for multiagent BDI logic. Since we want to have applicable sequent calculus, we need to use at least sound, complete and decidable calculus. There must exit decision procedure, which always terminates and returns result.

Usual sequent calculi with cut rule are practically unusable in automated environment. Since simple cut rule enables us to choose any formula in the cut rule premise, there may exist infinite count of the inference trees for the particular sequent, and, therefore, decision procedure do not exist.

We say, that sequent calculus uses analytic cut, if there are only finite possible cut rule applications for the particular sequent. So, we need to construct sequent calculus, which is cut free, or only analytic cut is used. Examples of the cut elimination may be found in [47, 23].

Cut free sequent calculus (or sequent calculus with an analytic cut) are not enough. Every sequent calculus rule must be invertable or at least semi-invertable. If some rules are neither invertable nor semi-invertable, then only special derivation tactics may guarantee decidability. Therefore, we try to use only invertable or, if impossible, semi-invertable rules.

Even if every sequent calculus rule is invertable or semi-invertable, we still may fail with decision procedure construction. The following property of the sequent calculus must also hold: every inference tree branch derivation must terminate. For some logics (especially for modal logics), there are used sequent calculus rules those premises contains more (or greater) formulas then conclusion. So, for such a logic, potentially derivation may be infinite. Usually, sequent calculus rules satisfies superformula condition - every formula in any sequent in the inference tree is subformula of the some formula placed in the initial sequent. It means, that, in any inference tree branch, we get one of the following cases:

- branch ends with some axiom of the used calculus, it means, that current branch is derivable,
- branch ends with some final sequent (non axiom sequent for which none of the sequent calculus rule may be applied), it means, that branch is non derivable,
- branch is infinite, and some sequents occur repeatedly on the branch.

For the last case, we may use loop-check technique to get finite decision procedure. If some sequent occurs on the same branch for several times we say that we have a loop. According to logic semantics, every loop may indicate that sequent is non derivable (commonly case), derivable, or unknown.

Sequent calculus may use weak loops or simple loops (see Definition 2.2.15 and Definition 2.2.16). In this thesis, all introduced sequent calculi uses simple loops, because they are based on the known sequent calculii, those use simple loops. Despite of this, all introduced lemmas, those restrict loop-check, are also valid for weak loops.

Loop-check is a technique (introduced by M. Fitting in [27]), which allows terminate derivation process if only some loop is detected. Loop-check may be used only if any loop indicates derivable or non derivable sequent.

Loop-check enables us to get finite decision procedure, and, therefore, it is very important and valuable. The main disadvantage of the loop-check is its inefficiency. Sequent derivation using direct loop-check may be described with the following steps:

1. Find sequent calculus rule, which may be applied for the current sequent.
2. Apply found rule and add new sequent (or several sequents, if rule contains several premises) into the inference tree.
3. If it is an axiom, or final sequent - terminate.
4. Otherwise perform loop-check:
   test if the sequent placed 1 level bellow is the same as current;
   if not, test if the sequent placed 2 level bellow is the same as current;
   if not, test if the sequent placed 3 level bellow is the same as current;
   and so on till we found a loop, or reach the initial sequent (inference tree root).
   If we found a loop - terminate. If not, proceed.
5. Take the first leaf of the inference tree and proceed the same.

When loop-check is unused, one step in the inference tree construction is just one rule application. When loop-check is used, one step in the inference tree construction is not only one rule application, but also testing all ancestor sequents. So, it is obviously, that most of the time is used for the loop-check, but not for the rule applications.

This is the reason we need loop-check free sequent calculus, or calculus with some efficient loop-check. One of the possibilities to construct sequent calculi with an efficient loop-check is sequents with histories usage (as it is done for some modal logics in [29]).

History is such an additional formulas stored in the sequent, those contain information about the rules applied between the root sequent and the current sequent, or other information about ancestor sequents. Using histories loop existence may be checked locally without testing every ancestor sequent.

It depends on the researched logic, but sometimes histories enables us to construct even loop-check free sequent calculi. If used, histories must use as less space as it is possible and loop-check must be performed using admissible time.

For every sequent calculus, which uses loop-check to get decision procedure, we may create a loop-check free sequent calculus, which uses histories. We only have to store every ancestor sequent in the history, and we need to look for a loop in history, instead of testing ancestor sequents. In such a way, formally, we always get a loop-check free sequent calculus. Unfortunately, such an approach is bad, because we still have to test the same count of the sequents and we need to store enormous amount of redundant information.

Usually, then histories are used, not all, but only several formulas from the ancestor sequents are added to the history. Histories store only formulas, those have real impact to loop existence. Therefore, histories may vary for different logics. J.M. Howe (in [30]) compares 2 types of the used histories, those may be used for loop-check restriction.

As a special case of histories used, indexes for formulas, or only indexes for operators may be used. Such a indexes are also histories, because store information about already applied rules. Usage of such an indexes may be found in [40, 41], where loop-check free sequent calculi for the fragments of some logics are presented.

As a very special type of histories used, marked operators may be applied. Marked operator is such an operator, which has the same semantics, but uses special notation to separate it from non marked operator. Marked sequents may also be used as special type of histories.

From the space complexity perspective, indexes are better then normal histories, and marked operators are even better then indexes, because they use less space.

In this thesis, we also use histories to get efficient loop-check: we use marked modal operators to get loop-check free sequent calculus for $KD45$ logic, we use marked sequents and special indexes to get an efficient loop-check for branching time logic.

Another related aspect of the inference tree construction is backtracking. If during sequent derivation we constructed final inference tree, which is not a derivation tree, we need to backtrack (return to some rule application and choose another possible premise for that rule, or even another rule). We need backtracking to ensure that derivation tree do not exist (to prove that sequent is non derivable).

If every calculus rule is invertable, then backtracking is not needed at all. Unfortunately, modal logics contains not only invertable rules, but also semi-invertable rules. In the case then semi-invertable rules are used, backtracking is needed to get decidability.

Backtracking may be eliminated if sequent calculus restricts only with invertable rules. Another way to eliminate backtracking is or-rules used instead of the traditional and-rules. Or-rules are semi-invertable rules, because, according definition, rule conclusion is derivable if at least one of the or-rule premises is derivable.

Backtracking and or-rules are very similar, but for some cases, one method may have advantages. In this thesis, all new efficient sequent calculus uses or-rules and eliminates backtracking. For the modal logic $KD45$, loop-check was eliminated because of the used or-rules.

Chapter 3

Loop-check Free Sequent Calculus for $KD45$ Logic

In this chapter, modal logic $KD45$ is discussed. Modal logic $KD45$ is used to express agents beliefs and is one of the main parts of the BDI logics. Known sequent calculus for modal logic $KD45$ is presented. New weak free sequent calculus and new loop-check free sequent calculus for $KD45$ logic are introduced. Proven complexity results for loop-check free sequent calculus for $KD45$ logic are also presented.

In this chapter, $KD45$ logic is researched. Therefore, only logical operators ($\neg$, $\vee$, $\&$) and modal operators $\Box$, $\Box^*$ are used. Formulas, those contain other modal operators are not well-formed formulas for $KD45$ logic.

3.1 Calculi for $KD45$ Logic

In this section, known calculi for $KD45$ logic are presented. Hilbert style axiomatization and known sound and complete sequent calculus are given.

Modal logic $KD45$ is also known as $weak - S5$ logic, doxastic logic, or logic of the individual belief. This logic uses one modality belief to express agents beliefs about its world. Halpern and Moses (in [28]) have shown that modal logic $KD45$ is the best to express agents beliefs. Different modal logics for agent systems uses $KD45$ logic to express agents beliefs (for example BDI, $KARO$ logics, logic of knowledge and belief and others). In this thesis, $KD45$ is important as the part of the researched BDI logics.

Definition 3.1.1 Hilbert type calculus for $KD45$ logic is calculus with classical non modal axioms, modal axioms as follows:

- K. $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$,
- D. $\Box\phi \rightarrow \neg\Box\neg\phi$,

- 4. $\Box\phi \to \Box\Box\phi$,
- 5. $\neg\Box\phi \to \Box\neg\Box\phi$,

and rules:

$$\frac{\phi, \quad \phi \to \psi}{\psi}, \qquad \frac{\phi}{\Box\,\phi}.$$

Tableau method for $KD45$ logic may be found in [25], where tableaux methods for various normal logics are presented. Single step tableaux method (combination of the Gentzen-type and prefixed tableaux methods) for $KD45$ logic may be found in [37]. Resolution based system (as a part of the system for logic containing CTL and $KD45$ logics) may be found in [21].

Sequent calculus for $KD45$ was introduced in [48]. As the basis for our research we use sequent calculus for $KD45$ logic with a slight modification and it may be found in [39]. It uses modal rule $(\Box)$ besides logical rules (see Definition 2.2.4.):

Definition 3.1.2 Sequent calculus rule $(\Box)$ is:

$$\frac{\Gamma, \Box\Gamma \to \Theta, \Box\Theta, \Box\Delta}{\Box\Gamma \to \Box\Theta, \Box\Delta} \quad (\Box)$$

- Θ is empty or only one formula.
- Γ - set of the formulas obtained from $\Box\Gamma$ by removing the most outer $\Box$ occurence.

Definition 3.1.3 The sequent calculus with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak)$ and modal rule $(\Box)$ we define sequent calculus $KD45_{init}$.

We say, that $KD45_{init}$ is an initial sequent calculus for $KD45$ logic. This sequent calculus contains non invertable (also non semi-invertable) rule $(Weak)$. Therefore, at least some derivation tactics is necessary to get decidability.

Theorem 3.1.1 Sequent calculus $KD45_{init}$ is sound and complete calculus for $KD45$ logic.

Proof.

The proof is presented by N. NIDE and T. Shiro in [39] as the fragment of the BDI logic.

According to [39], decision procedure, based on the sequent calculus $KD45_{init}$, exists (some special tactics is used). This decision procedure uses loop-check technique to get decidability. Loop-check is used to prove that sequent is non derivable. Described loop-check is inefficient, because it requires after every rule application to check all the ancestor sequents of the current sequent. Such a direct loop-check takes most of the time used for derivation. This is the main disadvantage of the sequent calculus $KD45_{init}$. In the following sections we present new sequent calculus, which is loop-check free, and, therefore, it obviate the main disadvantage of the sequent calculus $KD45_{init}$.

3.2 Loop-check Free Sequent Calculus for $KD45$ Logic

In this section, intermediate sequent calculus for $KD45$ logic, which eliminates non invertable rule $(Weak)$ is given. Some features of the created sequent calculus are proven. These features was used to introduce new loop-check free sequent calculus for $KD45$ logic. Equivalence of all presented calculi (initial, weak free and loop-check free sequent calculi) for $KD45$ logic are also proven.

First, we introduce an intermediate sequent calculus for $KD45$ logic which uses new modal rule $(\Box^W)$ instead of the rules $(\Box)$ and $(Weak)$. The new $(\Box^W)$ rule is or-rule. So, derivation tree constructed according new sequent calculus may contain or=branches besides and-branches.

Sequent calculus $KD45_{wf}$ (and $KD45_{init}$) do not use marked modal operator $\Box^*$, but such a marked operator is used in the sequent calculus $KD45_{lcf}$ (introduced later). Therefore, we also use marked modal operator $\Box^*$ in the following definitions, since they are applicable for the both calculi.

Definition 3.2.1 We say that sequent $\Sigma, \Box\Gamma, \Box^*\Gamma' \rightarrow \Pi, \Box\Delta, \Box^*\Delta'$ is a primal sequent for $KD45$ logic if:

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Definition 3.2.2 We say that sequent $\Box\Gamma, \Box^*\Gamma' \rightarrow \Box\Delta, \Box^*\Delta'$ is a strict-primal sequent for $KD45$ logic.

Definition 3.2.3 Sequent calculus rule $(Weak^*_{KD45})$ is:

$$\frac{\Box\Gamma \rightarrow \Box\Delta}{\Sigma, \Box\Gamma \rightarrow \Pi, \Box\Delta} \quad (Weak^*_{KD45})$$

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Simple speaking, rule $(Weak^*_{KD45})$ is just a rule $(Weak)$, which is restricted to be used only for primal sequents. Rule $(Weak^*_{KD45})$ do not delete any modalized formula. Therefore, rule $(Weak^*_{KD45})$ is invertable rule in $KD45$ logic, since Σ and Π contains only propositional variables.

Definition 3.2.4 Sequent calculus rule $(\Box^{or})$ is:

$$\frac{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \dots, \Box\phi_n \quad || \quad \dots \quad || \quad \Gamma, \Box\Gamma \to \phi_n, \Box\phi_1, \dots, \Box\phi_n}{\Box\Gamma \to \Box\phi_1, \Box\phi_2, \dots, \Box\phi_n} \quad (\Box^{or})$$

- The case $n = 0$ is allowed, and then rule transforms into: $\dfrac{\Gamma, \Box\Gamma \to}{\Box\Gamma \to} \quad (\Box^{or})$.
- Γ - set of the formulas obtained from $\Box\Gamma$ by removing the most outer $\Box$ occurence.

It is obviously, that rule $(\Box^{or})$ is semi-invertable rule.

Rule $(Weak^*_{KD45})$ may be applied only for primal sequent and its premise is always some strict-primal sequent. Rule $(\Box^{or})$ may be applied only for strict-primal sequent. We define new rule $(\Box^W)$ which is just a concatenation of the rules $(Weak^*_{KD45})$ and $(\Box^{or})$:

Definition 3.2.5 Sequent calculus rule $(\Box^W)$ is a concatenation of the rules $(Weak^*_{KD45})$, $(\Box^{or})$ and may be defined as follows:

$$\frac{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \dots, \Box\phi_n \quad || \quad \dots \quad || \quad \Gamma, \Box\Gamma \to \phi_n, \Box\phi_1, \dots, \Box\phi_n}{\Sigma, \Box\Gamma \to \Pi, \Box\phi_1, \dots, \Box\phi_n} \quad (\Box^{W})$$

- The case $n = 0$ is allowed, and then rule transforms into: $\dfrac{\Gamma, \Box\Gamma \to}{\Sigma, \Box\Gamma \to \Pi} \quad (\Box^W)$.
- Σ, Π - finite (may be empty) sets of the propositional variables only, $\Sigma \cap \Pi = \emptyset$.
- Γ - set of the formulas obtained from $\Box\Gamma$ by removing the most outer $\Box$ occurence.

If a sequent has $n > 0$ modalized formulas on the right side of $\to$, then rule $(\Box^W)$ contains n premises, those in consequence generates n or-branches. We have or-branches, since a sequent is derivable if at least one of these branches is derivable. It is obviously, that rule $(\Box^W)$ is semi-invertable rule. Actually, rule $(\Box)$ has or-branches, since during rule $(\Box)$ application one formula from the right side must be chosen.

Definition 3.2.6 The sequent calculus with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules and modal rule $(\Box^W)$ we define sequent calculus $KD45_{wf}$.

We say, that $KD45_{wf}$ is a weak free sequent calculus for $KD45$ logic. In this sequent calculus, rule $(\Box^W)$ is semi-invertable rule and all other rules are invertable.

Theorem 3.2.1 A sequent is derivable in the sequent calculus $KD45_{wf}$ if and only if it is derivable in the sequent calculus $KD45_{init}$.

Proof.

If a sequent S is derivable in the $KD45_{init}$, then we have a derivation tree, which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, $(\Box)$, and $(Weak)$. First, we have to eliminate from this tree every rule $(Weak)$ application. There are the following cases of the rule $(Weak)$ application:

1. The premise sequent of the rule $(Weak)$ application is an axiom. Then we can delete such a rule $(Weak)$ application at all, since the conclusion sequent of the rule $(Weak)$ application is also an axiom.

2. Rule $(Weak)$ is consecutively applied two times. Then we can transform $(Trans_1)$ such a part of the tree to use only one application of the rule $(Weak)$:

$$\frac{\dfrac{\dfrac{\cdots}{\Gamma \to \Delta}}{\Gamma, \Gamma' \to \Delta, \Delta'}\,(Weak)}{\Gamma, \Gamma', \Gamma'' \to \Delta, \Delta', \Delta''}\,(Weak)$$

$$\Downarrow \qquad\qquad (Trans_2)$$

$$\frac{\dfrac{\cdots}{\Gamma \to \Delta}}{\Gamma, \Gamma', \Gamma'' \to \Delta, \Delta', \Delta''}\,(Weak)$$

3. Rule $(Weak)$ and logical rules R is consecutively applied (in the bottom-up direction) (rule R is one of the $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$). We can change the order of these rules applications (rule $(Weak)$ application appears above the logical rule R application). We perform transformation $Trans_2$:

$$\frac{\dfrac{\dfrac{\cdots}{\phi, \Gamma \to \Delta} \quad \dfrac{\cdots}{\psi, \Gamma \to \Delta}}{\phi \vee \psi, \Gamma \to \Delta}\,(\vee L)}{\phi \vee \psi, \Gamma, \Gamma' \to \Delta, \Delta'}\,(Weak)$$

$$\Downarrow \qquad\qquad (Trans_2)$$

$$\frac{\dfrac{\dfrac{\cdots}{\phi, \Gamma \to \Delta}}{\phi, \Gamma, \Gamma' \to \Delta, \Delta'}\,(Weak) \quad \dfrac{\dfrac{\cdots}{\psi, \Gamma \to \Delta}}{\psi, \Gamma, \Gamma' \to \Delta, \Delta'}\,(Weak)}{\phi \vee \psi, \Gamma, \Gamma' \to \Delta, \Delta'}\,(\vee L)$$

Transformation $Trans_2$ is performed for the rule $(\vee L)$. The analogous transformation can be also applied for rules $(\neg L)$, $(\neg R)$, $(\vee R)$, $(\& L)$, $(\& R)$. In such a case, we can raise rule $(Weak)$ application by one step in an inference tree.

4. Rules $(Weak)$ and $(\Box)$ are consecutively applied (in the bottom-up direction). In this case, we can do everything rather the same as in the Case 3. The difference is that during rule $(Weak)$ raising we change rule $(\Box)$ application by the new rules $(Weak^*_{KD45})$ and $(\Box^{or})$ applications (i.e. by the one rule $(\Box^W)$ application).

We denote the main formula of the rule $(\Box)$ application by $\Box\phi_1$ (may be empty). We denote all other not deleted modalized formulas by $\Box\Delta = \Box\phi_2, \ldots, \Box\phi_k$ and all deleted modalized formulas by $\Box\Delta' = \Box\phi_{k+1}, \ldots, \Box\phi_n$. We perform transformation $Trans_3$:

$$\dfrac{\dfrac{\dfrac{\ldots}{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \Box\Delta}}{\Box\Gamma \to \Box\phi_1, \Box\Delta}\,(\Box)}{\Lambda, \Box\Gamma, \Box\Gamma' \to \Pi, \Box\phi_1, \Box\Delta, \Box\Delta'}\,(Weak)$$

$$\Downarrow \qquad (Trans_3)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\ldots}{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \Box\Delta}}{\Gamma, \Box\Gamma, \Gamma', \Box\Gamma' \to \phi_1, \Box\phi_1, \Box\Delta, \Box\Delta'}\,(Weak) \quad || \quad \dfrac{\ldots}{S_i}}{\Box\Gamma, \Box\Gamma' \to \Box\phi_1, \Box\Delta, \Box\Delta'}\,(\Box^{or})}{\Lambda, \Box\Gamma, \Box\Gamma' \to \Pi, \Box\phi_1, \Box\Delta, \Box\Delta'}\,(Weak^*_{KD45})$$

Here $S_i = \Gamma, \Box\Gamma, \Gamma', \Box\Gamma' \to \phi_i, \Box\phi_1, \Box\Delta, \Box\Delta'$ - for all $i = 2, 3, \ldots, n$.

Sequents S_i are some unused or-branches of the derivation tree, because we get a derivation tree (in the calculus $KD45_{init}$) by choosing formula $\Box\phi_1$ as the main for the rule $(\Box)$ application. In other words, we do not proceed sequents S_i in the sequent calculus $KD45_{wf}$, because these branches do not impact derivability of the initial sequent and are redundant.

By applying such a transformations in the bottom-up direction we eliminate all applications of the rule $(Weak)$ from the sequent S derivation tree. If there is left some rule $(\Box)$ application in the tree, apply transformation like in the Case 4 to get the rule $(\Box^W)$ application instead of the rule $(\Box)$ application. After these transformations we get a derivation tree for the sequent calculus $KD45_{wf}$, therefore, sequent S is also derivable in the $KD45_{wf}$.

If a sequent S is derivable in the $KD45_{wf}$, then we have a derivation tree, which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\&L)$, $(\&R)$, $(\Box^W)$. We can apply another transformation to change rule $(\Box^W)$ application into $(Weak)$ and $(\Box)$ rules applications. After rule $(\Box^W)$ application, we can get n sequents. At least one of them is derivable if the father sequent is derivable. If an initial sequent S is derivable, then we can leave only one branch from or-branches which is derivable in the $KD45_{wf}$. Suppose that we have chosen such a derivable branches (one for every rule $(\Box^W)$ application) and marked them as derivable. All other or-branches become redundant and, therefore, we mark them as redundant. We use consequently applied rules $(Weak^*_{KD45})$ and $(\Box^{or})$ instead of the rule $(\Box^W)$. Suppose that derivable branch is obtained by taking $i = 1$.

We apply transformation $Trans_4$:

$$\dfrac{\dfrac{\dfrac{\overset{(derivable)}{\cdots}}{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \ldots, \Box\phi_n} \quad || \ldots || \quad \dfrac{\overset{(redundant)}{\cdots}}{\Gamma, \Box\Gamma \to \phi_n, \Box\phi_1, \ldots, \Box\phi_n}}{\Box\Gamma \to \Box\phi_1, \ldots, \Box\phi_n}\,(\Box^{or})}{\Sigma, \Box\Gamma \to \Pi, \Box\phi_1, \ldots, \Box\phi_n}\,(Weak^*_{KD45})$$

$$(i = 2, 3, \ldots, n)$$

$$\Downarrow \qquad\qquad (Trans_4)$$

$$\dfrac{\dfrac{\dfrac{\overset{(derivable)}{\cdots}}{\Gamma, \Box\Gamma \to \phi_1, \Box\phi_1, \Box\phi_2, \ldots, \Box\phi_n}}{\Box\Gamma \to \Box\phi_1, \Box\phi_2, \ldots, \Box\phi_n}\,(\Box)}{\Sigma, \Box\Gamma \to \Pi, \Box\phi_1, \ldots, \Box\phi_n}\,(Weak)$$

By applying such a transformations we eliminate all applications of the rule $(\Box^W)$ from the derivation tree. After these transformations we get a derivation tree for the sequent calculus $KD45_{init}$, therefore, sequent S is also derivable in the $KD45_{init}$.

The advantage of the sequent calculus $KD45_{wf}$ is that we can determine whether sequent is derivable or not. For such a decision procedure loop-check is needed, but no special tactics is needed anymore.

Now we will show that any formula in the sequent S inference tree is some subformula of the initial sequent S.

Lemma 3.2.1 If F is some formula in any sequent S' in the sequent S inference tree constructed according to the sequent calculus $KD45_{wf}$, then there exists formula G in the initial sequent S, that $F \subseteq_{sf} G$.

Proof.

Premises of any sequent calculus $KD45_{wf}$ rule ($(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, $(\Box^W)$) contains only subformulas of the rule conclusion. According to the Definition 2.1.7., formula F is subformula of some formula G in the initial sequent S.

Sequent calculus $KD45_{wf}$ uses and-rules ($(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$) and one or-rule ($(\Box^W)$). If we have a finite constructed tree in the sequent calculus $KD45_{wf}$, all leaves are known to be derivable or not, and we can determine whether an initial sequent is derivable or not (see Section 2.2.). A leaf sequent is derivable if it is an axiom. A leaf sequent is non derivable if it is a final sequent (it is not an axiom and no rule can be applied for it). The next Lemma says that a leaf sequent is non derivable if it is a loop-ending sequent. Since in any sequent inference tree, we can only have subformulas of formulas from the initial sequent (Lemma 3.2.1.), the derivation procedure always terminates.

Lemma 3.2.2 Sequent S is non derivable in the sequent calculus $KD45_{wf}$ if we can construct inference tree which satisfies the following conditions:

- every tree leaf contains:
  i) some axiom,
  ii) some final sequent, or
  iii) some loop-ending sequent,
- sequent inference tree is not a derivation tree in the sequent calculus $KD45_{wf}$.

Proof.

Proof goes straightforward from the proof placed in [39].

This Lemma says, that if we found some loop-ending sequent in a derivation tree, we have to stop to proceed this branch and we have treat it as non derivable.

Since in any sequent S inference tree we can only have subformulas of the formulas from the initial sequent S (Lemma 3.2.1.), we get an inference tree, that any tree leaf contains an axiom, final sequent or loop-ending sequent. If such a constructed tree is sequent S derivation tree in the sequent calculus $KD45_{wf}$, then sequent S is derivable. If such a constructed tree is not a derivation tree in the sequent calculus $KD45_{wf}$, then sequent S is non derivable. So, for the sequent calculus $KD45_{wf}$, we have finite procedure to determine whether sequent is derivable or not.

We have a sequent calculus which uses loop-check technique to determine the sequent derivability. The main disadvantage of this calculus is an inefficient loop-check. To determine whether sequent S is the loop-ending sequent or not, we have to scan all ancestors of the sequent S. This procedure takes the most of the time during derivation.

According to the Lemma 3.2.1, for every sequent, we can construct a finite tree (with derivable or non derivable sequent in the root).

Lemma 3.2.3 If we have a loop $S \rightsquigarrow S'$ in the sequent calculus $KD45_{wf}$, when at least one rule $(\Box^W)$ application exists between S and S' in a inference tree.

Proof.

Suppose that only rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$ are applied between S and S'. In such a case, we cannot get a loop $S \rightsquigarrow S'$, since all used rules decrease the number of operations used in the sequent and, therefore, $len(S) > len(S')$ and $S \rightsquigarrow S'$ is not a loop. We get a contradiction, and, therefore, at least one rule $(\Box^W)$ application exists between S and S'.

Lemma 3.2.4 If we have a loop $S \rightsquigarrow S'$, then all the sequents between S and S' in an inference tree contain the same modalized formulas.

Proof.

All rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$ do not change any modalized formula (rule premises and conclusion contains the same modalized formulas). Rule $(\Box^W)$

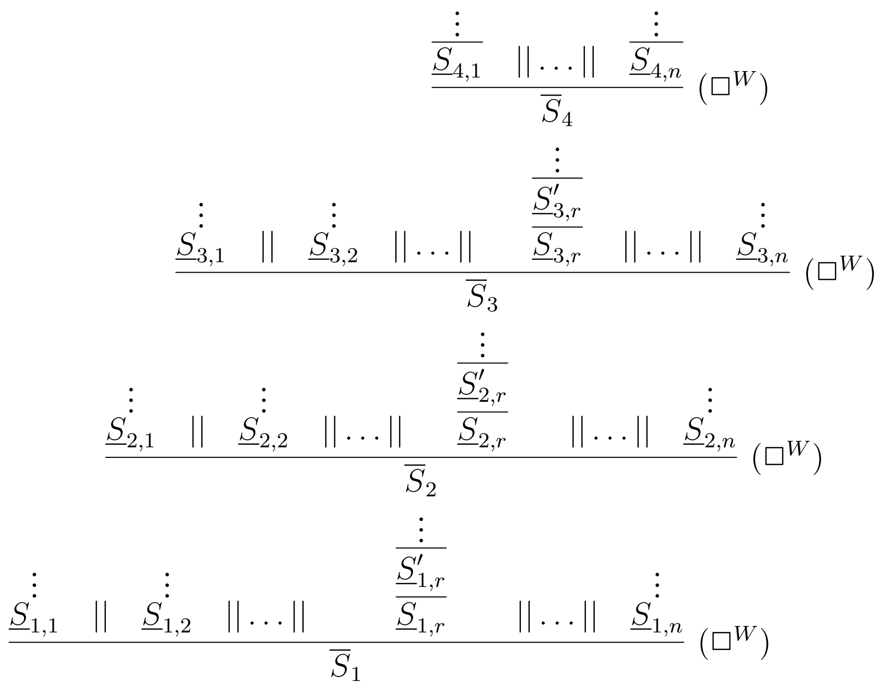


Figure 3.1: Illustration for the proof of the Lemma 3.2.5.
Fragment of the sequent $\overline{S}_1$ inference tree.

can only increase the number of modalized formulas (premises contains more modalized formulas then conclusion). Taking into account that sequents S and S' contain the same formulas, we get that all the sequents between S and S' contain the same modalized formulas.

Now we introduce some features of the sequent calculus $KD45_{wf}$. These features are very useful while optimizing loop-check. After, we introduce a new sequent calculus, which is based on the sequent calculus $KD45_{wf}$. We will use the above mentioned features to get loop-check free sequent calculus for $KD45$ logic.

To separate sequents used in the rule $(\Box^W)$ application in a derivation tree, we mark them with a line. Suppose, that sequent S' is a premise and S'' is a conclusion of the rule $(\Box^W)$ application. Then we mark the sequent S' by a line above ($\overline{S'}$) and the sequent S'' by a line below ($\underline{S''}$).

Now we show that only sequents for which rule $(\Box^W)$ is applied (those are denoted by a line above - $\overline{S}$) may be used in a loop-check. Moreover, we show that such a loop can be checked locally. We say that loop-check acts locally if only current or maybe some one particular ancestor sequent is used to determine the loop existence.

Lemma 3.2.5 Suppose that we have sequents $\overline{S}_1, \overline{S}_2$ and that the following conditions are true:

(a) $\overline{S}_1$ is an ancestor of $\overline{S}_2$ in the inference tree,

(b) $\overline{S}_1$ and $\overline{S}_2$ contain the same modalized formulas,

(c) Exactly only one time rule $(\Box^W)$ was applied between $\overline{S}_1$ and $\overline{S}_2$ (rule $(\Box^W)$ was applied for $\overline{S}_1$, after only rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$ were applied to get $\overline{S}_2$),

(d) The sequents $\underline{S}_{1,0}, \dots \underline{S}_{1,n}$ are children of $\overline{S}_1$, and $\underline{S}_{1,r}$ is such of them, that it is an ancestor of the $\overline{S}_2$ (the path from $\overline{S}_1$ to $\overline{S}_2$ goes through $\underline{S}_{1,r}$: $\overline{S}_1 \to \underline{S}_{1,r} \to \dots \to \overline{S}_2$).

Then $\overline{S}_1$ is derivable if and only if there exists a derivable sequent $\underline{S}_{1,i} \neq \underline{S}_{1,r}$.

Proof. In general, a sequent $\overline{S}_1$ can be derivable if and only if at least one of the sequents $\underline{S}_{1,0}, \dots \underline{S}_{1,n}$ is derivable. So, if the sequent $\overline{S}_1$ is non derivable, then all sequents $\underline{S}_{1,0}, \dots \underline{S}_{1,n}$ are non derivable and, therefore, there do not exist such a derivable sequent $\underline{S}_{1,i} \neq \underline{S}_{1,r}$.

Suppose that the sequent $\overline{S}_1$ is derivable and all conditions of the Lemma are satisfied (see Figure 3.1). Set $\overline{S}_j = \overline{S}_1$. Sequent $\overline{S}_{j+1}$ and $\overline{S}_j$ contain the same modalized formulas and rule $(\Box^W)$ application generates the same premises for both sequents. Then we have 2 possible cases:

1. There exists a sequent $\underline{S}_{j,i} \neq \underline{S}_{j,r}$ ($\underline{S}_{1,i} = \underline{S}_{j,i} \neq \underline{S}_{j,r} = \underline{S}_{1,r}$), which is derivable, and the Lemma is true.

2. $\underline{S}_{j,i}$ is non derivable for every $i \in 1, \dots, n, i \neq r$, the sequent $\underline{S}_{j,r}$ is derivable. If so, we have two subcases:

   2.1. there is no rule $(\Box^W)$ application above the sequent $\underline{S}_{j,r}$, i.e., only rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$ are applied. In such a case, $\underline{S}_{j,r}$ is derivable only if all its children (and its children's children) are derivable, because all these rules creates only and-branches of the tree. Moreover, all of them are axioms and have the same modalized formulas.

   Since, rule $(\Box^W)$ is applied for the sequents $\overline{S}_1, \overline{S}_2$, we have that $j \geq 2$. Sequents $\overline{S}_{j-1}$ and $\overline{S}_j$ contains the same modalized formulas and rule $(\Box^W)$ is applied for them. Suppose that $\underline{S}_{j-1,0}, \dots \underline{S}_{j-1,n}$ are obtained from $\overline{S}_{j-1}$ by the rule $(\Box^W)$ application. Therefore, $\underline{S}_{j,r} = \underline{S}_{j-1,r}$. We get that no rule $(\Box^W)$ is applied for $\underline{S}_{j-1,r}$. We get a contradiction for the sequent $\underline{S}_{j,r}$ existence (it is obtained by the rule $(\Box^W)$ application, and $\underline{S}_{j-1,r}$ is its ancestor).

   2.2. Rule $(\Box^W)$ is applied above the sequent $\underline{S}_{j,r}$ and such a sequent $\overline{S}_{j+1}$ exists. Since $\underline{S}_{j,r}$ is derivable, then $\overline{S}_{j+1}$ must also be derivable. Let

$\underline{S}_{j+1,0}, \ldots \underline{S}_{j+1,n}$ be the sequents obtained from $\overline{S}_{j+1}$ (rule $(\Box^W)$). According to condition (b), $\underline{S}_{j,i} = \underline{S}_{j+1,i}$ for every $i \in 1, \ldots, n$. If the sequent $\overline{S}_{j+1}$ is derivable, then one of the sequents $\underline{S}_{j+1,0}, \ldots \underline{S}_{j+1,n}$ is also derivable:

2.2.1. If there exists derivable $\underline{S}_{j+1,i} \neq \underline{S}_{j,r}$, then $\underline{S}_{1,i} = \underline{S}_{j,i} \neq \underline{S}_{j,r} = \underline{S}_{1,r}$ is also derivable and we get a contradiction to the Case 2 (only $\underline{S}_{j,r}$ is derivable).

2.2.2. If only $\underline{S}_{j+1,r} = \underline{S}_{j,r}$ is derivable, then $\underline{S}_{j+1,i}$ is non derivable for every $i \in 1, \ldots, n, i \neq r$, and the sequent $\underline{S}_{j+1,r}$ is derivable. In such a case, we can apply inductively the same arguments for $\underline{S}_{j+1,r}$ as for $\underline{S}_{j,r}$ in the Case 2 till we reach leaves of the tree. Then we reach leaves, we get the same contradiction as in the Case 2.1.

Lemma 3.2.5 says that if we find two sequents $\overline{S}_1$ and $\overline{S}_2$ satisfying the conditions of the Lemma, then the branch above the sequent $\overline{S}_2$ does not impact derivation at all, and we can terminate proceeding such a branch without any loss. This is the reason why Lemma 3.2.5 is the main for constructing the loop-check free sequent calculus for $KD45$ logic.

We have to mention the fact, that the existence of such a sequents $\overline{S}_1, \overline{S}_2$ (satisfying the conditions of the Lemma 3.2.5) do not guarantee that the sequent $\overline{S}_2$ is non derivable by itself. We only know that nothing depends on the branches above $\overline{S}_2$. Even if the initial sequent $\overline{S}_2$ is derivable by itself, we can treat $\overline{S}_2$ as non derivable branch, since some ancestor of $\overline{S}_2$ will be derivable as well because of the other or-branches placed below $\overline{S}_2$.

Now we can terminate the proceed some sequent S during the derivation tree construction if one of the following 4 cases occurs:

1. S is an axiom (such a branch is derivable and, S is derivable by itself).

2. S has only atomic formulas and none rule can be applied (such a branch is non derivable, and S is non derivable by itself).

3. S is some loop-ending sequent (such a branch is non derivable, and S is non derivable by itself).

4. S is a sequent for which rule $(\Box^W)$ must be applied and there exists such an ancestor sequent $\overline{S}'$ with the same modalized formulas (such a branch is treated as non derivable but S can be derivable or non derivable by itself, as it was described above).

To determine the Case 3 or the Case 4 we have to use loop-check technique. The main difference in loop-check used is that, in the Case 4, we can use loop-check locally, while, in the Case 3, we cannot. In the Case 3, we have to scan all the sequents below S even till the root of the tree, while, in the Case 4, we have to test only one particular sequent below S (the sequent for which the last rule $(\Box^W)$ was applied). Sequent calculus $KD45_{wf}$ uses only the Cases 1, 2 and 3. Now we introduce the new sequent calculus, which uses the Cases 1, 2 and 4 for derivation termination. First we show that the Case 3 is redundant.

Lemma 3.2.6 If some branch in the derivation tree ends with a loop $S \rightsquigarrow S'$, then on that branch, one can find sequents $\overline{S}_1, \overline{S}_2$ satisfying the conditions of the Lemma 3.2.5.

Proof. Proof is straightforward from the Lemma 3.2.4 and the Lemma 3.2.5.

According to the Lemma 3.2.6, the Case 3 for termination of the derivation is redundant, since if some branch derivation was terminated because of the Case 3, its derivation can be also terminated because of the Case 4.

The new calculus uses rule $(\Box^{LCF})$ (rule $\Box$ for loop-check free calculus) instead of the rule $(\Box^W)$ and uses marked modal operator $\Box^*$ to determine whether any new modalized formula appears after the last rule $(\Box^{LCF})$ application:

Definition 3.2.7 Sequent calculus rule $(\Box^*)$ is:

$$\frac{\Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \to \phi_1, \Box^*\phi_1, \ldots, \Box^*\phi_n || \ldots || \Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \to \phi_n, \Box^*\phi_1, \ldots, \Box^*\phi_n}{\Box\Gamma, \Box^*\Gamma_1 \to \Box\phi_1, \ldots, \Box\phi_k, \Box^*\phi_{k+1}, \ldots, \Box^*\phi_n}$$

$(\Box^*)$ can be applied only if $\Box\Gamma \cup \Box\phi_1 \cup \ldots \cup \Box\phi_k \neq \emptyset$.

- The case $n = 0$ is allowed, and then rule transforms into: $\frac{\Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \to}{\Box\Gamma, \Box^*\Gamma_1 \to} \ (\Box^*)$.
- $\Box^*\Gamma$ - set of the formulas obtained from $\Box\Gamma$ by replacing every most outer $\Box$ occurence with $\Box^*$.

It is obviously, that rule $(\Box^*)$ is semi-invertable rule.

Rule $(Weak^*_{KD45})$ may be applied only for primal sequent and its premise is always some strict-primal sequent. Rule $(\Box^*)$ may be applied only for strict-primal sequent. We define new rule $(\Box^{LCF})$, which is just a concatenation of the rules $(Weak^*_{KD45})$ and $(\Box^*)$:

Definition 3.2.8 Sequent calculus rule $(\Box^{LCF})$ is:

$$\frac{\Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \to \phi_1, \Box^*\phi_1, \ldots, \Box^*\phi_n || \ldots || \Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \to \phi_n, \Box^*\phi_1, \ldots, \Box^*\phi_n}{\Sigma, \Box\Gamma, \Box^*\Gamma_1 \to \Pi, \Box\phi_1, \ldots, \Box\phi_k, \Box^*\phi_{k+1}, \ldots, \Box^*\phi_n}$$

$(\Box^{LCF})$ can be applied only if $\Box\Gamma \cup \Box\phi_1 \cup \ldots \cup \Box\phi_k \neq \emptyset$.

- The case $n = 0$ is allowed, and then rule transforms into:
  $\dfrac{\Gamma, \Gamma_1, \Box^*\Gamma, \Box^*\Gamma_1 \rightarrow}{\Sigma, \Box\Gamma, \Box^*\Gamma_1 \rightarrow \Pi} \quad (\Box^{LCF}).$
- Σ, Π - finite (may be empty) sets of propositional variables only, $\Sigma \cap \Pi = \emptyset$.
- $\Box^*\Gamma$ - set of the formulas obtained from $\Box\Gamma$ by replacing every most outer $\Box$ occurence with $\Box^*$.

We have to mention, that if, after some rule application, we have formulas $\Box\phi$, $\Box^*\phi$ on the same side of $\rightarrow$, then we leave only one formula $\Box^*\phi$. Rule $(\Box^{LCF})$ marks all top-level $\Box$ operators with star symbol $(*)$ to indicate that such a formula was used in the last rule $(\Box^{LCF})$ application. Rule $(\Box^{LCF})$ can be applied only for the sequent, which has at least one modalized formula with non marked $\Box$ operator. Since after rule $(\Box^{LCF})$ application, all old top-level $\Box$ operators become marked, rule $(\Box^{LCF})$ can be applied for the second time in such a branch only if some new modalized formula with non marked $\Box$ operator appears. Such a restriction dramatically decreases the number of choices.

Definition 3.2.9 The sequent calculus with an axiom $\phi, \Gamma \rightarrow \phi, \Delta$, logical rules and modal rule $(\Box^{LCF})$ we define sequent calculus $KD45_{lcf}$.

We say, that $KD45_{lcf}$ is a loop-check free sequent calculus for $KD45$ logic. Sequent calculus $KD45_{lcf}$ rule $(\Box^{LCF})$ is semi-invertable rule and all the others are invertable.

Theorem 3.2.2 A sequent is derivable in the sequent calculus $KD45_{lcf}$ if and only if it is derivable in the sequent calculus $KD45_{wf}$.

Proof.

There is only one major difference between calculi $KD45_{wf}$ and $KD45_{lcf}$. Rule $(\Box^{LCF})$ is the same as in $(\Box^{W})$ but cannot be applied if $\Box\Gamma \cup \Box\phi_1 \cup \ldots \cup \Box\phi_k = \emptyset$. If some modalized formula has a marked $\Box$ operator $(\Box^*)$, then such a formula was used in the last rule $(\Box^{LCF})$ application. Suppose that, for a sequent $\overline{S}_2$, rule $(\Box^{LCF})$ cannot be applied, since $\Box\Gamma \cup \Box\phi_1 \cup \ldots \cup \Box\phi_k = \emptyset$, i.e., all modalized formulas have marked $\Box$ operators. So, all sequent $\overline{S}_2$ modalized formulas were used in the last rule $(\Box^{LCF})$ application. Suppose that the last rule $(\Box^{LCF})$ was applied for the sequent $\overline{S}_1$. Then the sequents $\overline{S}_1, \overline{S}_2$ have the same modalized formulas and satisfy all conditions placed in the Lemma 3.2.5. Therefore, the sequent $\overline{S}_2$ derivation can be stopped according the Case 3 without any loss.

If some sequent S is non derivable in the sequent calculus $KD45_{wf}$, then S is non derivable in the sequent calculus $KD45_{lcf}$ (Lemma 3.2.6). If a sequent is derivable in the sequent calculus $KD45_{wf}$, then it remains derivable in the sequent calculus $KD45_{lcf}$ (Lemma 3.2.5).

We got that sequent calculus $KD45_{lcf}$ is equivalent to the sequent calculus $KD45_{wf}$. Sequent calculus $KD45_{wf}$ is equivalent to the sequent calculus $KD45_{init}$. Therefore, sequent calculus $KD45_{lcf}$ is equivalent to the sequent calculus $KD45_{init}$.

Theorem 3.2.3 Sequent calculus $KD45_{lcf}$ is sound and complete calculus for $KD45$ logic.

Proof.

The proof goes straightforward from the Theorems 3.1.1, 3.2.1, 3.2.2.

Inference tree construction in the sequent calculus $KD45_{lcf}$ always terminates, because after every rule $(\Box^{LCF})$ application we get one more formula modalized with marked operator $\Box^*$. Marked operator $\Box^*$ cannot become unmarked. Therefore, finally we get only marked modalized formulas in the sequent, but, for such a sequent, rule $(\Box^{LCF})$ cannot be applied. It is the reason why, we can manage without loop-check.

Intermediate results related to this section are published in [5, 7]. The same approach was used to get a loop-check free sequent calculus for multimodal $KD45_n$ logic ([8]).

3.3 Complexity Results for Sequent Calculus $KD45_{lcf}$

In this section, complexity results for the new loop-check free sequent calculus are presented. Proven complexity results shows, that new sequent calculus $KD45_{lcf}$ is essentially better then known sequent calculus $KD45_{init}$. In this section, it is shown that trees constructed according to the calculus $KD45_{lcf}$ are smaller than constructed according to the calculus $KD45_{wf}$ and, moreover, that loop-check is totally eliminated. In addition, it includes a full-scale example of non derivable sequent derivation in the sequent calculus $KD45_{lcf}$.

The efficiency of the used loop-check can be estimated via time used to check the loop existence. Suppose that we have a branch with height n and we want to check the loop existence for some sequent placed in a tree leaf. For such a check in the $KD45_{wf}$, we have to review all the sequents placed bellow even till the root. If a loop exists, then we have to review only the sequents placed between loop-starting and loop-ending sequents, but if there is no loop, then we have to review all n sequents places below the current sequent. Such a review must be done for every sequent for which any rule can be applied and which is not an axiom. For the comparison in the sequent calculus $KD45_{lcf}$, we need to check only one special condition for the rule $(\Box^{LCF})$ application. Moreover, we need to do such a check only for the sequents for which rule $(\Box^{LCF})$ must be applied.

To avoid a blind review of all the sequents bellow the current sequent some histories can be used. Such an approach is used in works [29] and [38]. Even such an approach cannot be applied directly for the sequent calculus $KD45_{wf}$. Newly introduced sequent calculus $KD45_{lcf}$ does not explicitly use histories but the main idea of histories is used. We use a marked $\Box$ operator ($\Box^*$) to indicate the formula usage in the last rule ($\Box^{LCF}$) application. Usage of a marked modal operator $\Box^*$ allowed us to construct loop-check free sequent calculus, since all loop-check is included inside the rule ($\Box^{LCF}$) application restrictions.

Sequent calculus $KD45_{lcf}$ has one more advantage against the $KD45_{wf}$. Both sequent calculi construct finite trees (one uses loop-check and the other does not) but the tree constructed according to the $KD45_{lcf}$ is some subtree (with some generalization) of the tree constructed according to the $KD45_{wf}$ (this follows from Lemma 3.2.6). Of course, there are such cases when both trees have the same size, but for the most sequents containing modalized formulas, such a tree will be smaller. Since rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$ applications reduce the number of operators, the constructed tree size mostly depends on the modal rule application number. The next Lemma determines the maximal number of the rule ($\Box^{LCF}$) applications in one branch and, therefore, determines bounds for height of the whole tree.

Lemma 3.3.1 If sequent S contains k logical operators and m different occurrences of the modal operator $\Box$, then for any inference tree T in the $KD45_{lcf}$ $height(T) \leq m \cdot k$.

Proof.

Any sequent in a inference tree contains only subformulas of the initial sequent S. Every rule ($\Box^{LCF}$) changes at least one subformula of the shape $\Box\phi$ into $\Box^*\phi$. There is no rule which changes $\Box^*\phi$ into $\Box\phi$. So, at most m times rule ($\Box^{LCF}$) can be applied in any inference tree branch. Between two rules ($\Box^{LCF}$) applications at most k logical rules can be applied. Therefore, any branch in an inference tree do not exceed height $m \cdot k$. So, $height(T) \leq m \cdot k$.

We have to mention, that, for the sequent calculus $KD45_{wf}$, such a lemma is not correct. Lemma 3.3.1 determines the upper bound of the time complexity. Algorithm time complexity directly depends on the constructed inference tree size.

Lemma 3.3.2 If sequent S contains k logical operators and m occurrences of the modal operator $\Box$, then decision algorithm for the sequent calculus $KD45_{lcf}$ time complexity is $\leq 2 \cdot m^{m \cdot k}$.

Proof.

Since during the decision algorithm, every node is performed ≤ 2 times (see Appendix A for algorithm pseudocode). For any inference tree T, $height(T) \leq m \cdot k$, and every node has $\leq m$ branches (every rule contains $\leq m$ premises). Therefore, decision algorithm for the sequent calculus $KD45_{lcf}$ time complexity is $\leq 2 \cdot m^{m \cdot k}$.

Corollary 3.3.1 Sequent calculus $KD45_{lcf}$ time complexity is EXPTIME.

Of course, we get an exponential time complexity, but it is much better result then we can get for the sequent calculus $KD45_{wf}$ (or $KD45_{init}$), which uses direct loop-check and creates bigger (or equal) inference trees.

Now we propose some lemmas to show space complexity of the sequent calculus $KD45_{lcf}$.

Lemma 3.3.3 Decision algorithm for the sequent calculus $KD45_{lcf}$ space complexity is at most $O(l^3)$ (here $l = len(S)$).

Proof.

Suppose that sequent S has length $l = len(S)$. Every sequent in an inference tree contains only subformulas of S (including $\Box^*\phi$ for subformula $\Box\phi$). Every subformula has length $\leq l$ and there are $< 2 \cdot l$ different subformulas of S (because every formula F has $\leq l$ different non marked modalized subformulas, and every formula $\Box\phi$ has one additional subformula $\Box^*\phi$). We can give an index for any subformula of sequent S. We need $< 2 \cdot l \cdot l = 2 \cdot l^2$ space to store table of subformulas and their indexes.

Every sequent in the inference tree can be defined by two $2 \cdot l$ length arrays of subformulas indexes (one array for the left side, and one for the right side of the sequent). According to the Lemma 3.3.1, height of any branch $< m \cdot k < l^2$. Therefore, we need $< 2 \cdot l^2 + 2 \cdot 2 \cdot l \cdot l^2$ space if we use deep first search algorithm and stack (see Annex A for pseudocode). So, sequent calculus $KD45_{lcf}$ space complexity requires $O(l^3)$ space.

Corollary 3.3.2 Sequent calculus $KD45_{lcf}$ space complexity is PSPACE.

Example 3.3.1 Suppose we have a sequent $S = \Box(\phi \vee \psi) \rightarrow \Box\psi, \Box\Box\phi$. Sequent S has length $= 9$. Table of subformulas for the sequent S (used in the proof of the Lemma 3.3.3) may be the following:

Index of subformula	Subformula	Index of subformula	Subformula
1	$\Box(\phi \vee \psi)$	7	$\Box^*\psi$
2	$\Box^*(\phi \vee \psi)$	8	$\Box\Box\phi$
3	$\phi \vee \psi$	9	$\Box^*\Box\phi$
4	ϕ	10	$\Box\phi$
5	ψ	11	$\Box^*\phi$
6	$\Box\psi$		

Sequent $S' = \phi \vee \psi, \Box^*(\phi \vee \psi) \rightarrow \psi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ may be stored as 2 arrays of indexes: $[3, 2] \rightarrow [5, 11, 7, 9]$.

Example 3.3.2 We will show that the sequent $S_1^0 = \Box(\phi \vee \psi) \to \Box\psi, \Box\Box\phi$ is non derivable in the calculus $KD45_{lcf}$. If no rule can be applied for the sequent S, we mark it by $\times$, if the sequent S is an axiom we mark it by $\oplus$.

$$\dfrac{\dfrac{\overset{\oplus}{S_1^2}\quad\overset{\times}{S_2^2}}{S_1^1}(\vee L)\quad||\quad\dfrac{\dfrac{\dfrac{\overset{\oplus}{S_1^4}\quad\overset{\times}{S_2^4}}{S_1^3}(\vee L)\;||\;\dfrac{\overset{\oplus}{S_3^4}\quad\overset{\times}{S_4^4}}{S_2^3}(\vee L)\;||\;\dfrac{\overset{\times}{S_5^4}\quad\overset{\times}{S_6^4}}{S_3^3}(\vee L)}{S_3^2}(\Box^{LCF})\quad\dfrac{\overset{\vdots}{S_4^3}\;||\;\overset{\vdots}{S_5^3}\;||\;\overset{\vdots}{S_6^3}}{S_4^2}(\Box^{LCF})}{S_2^1}(\vee L)}{S_1^0}(\Box^{LCF})$$

$S_1^0 = \Box(\phi \vee \psi) \to \Box\psi, \Box\Box\phi : (\Box^{LCF}) :$

- $S_1^1 = \phi \vee \psi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\psi, \Box^*\Box\phi : (\vee L) :$
  - $S_1^2 = \phi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\psi, \Box^*\Box\phi$ (terminate)
  - $S_2^2 = \psi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\psi, \Box^*\Box\phi$ (axiom)
- $S_2^1 = \phi \vee \psi, \Box^*(\phi \vee \psi) \to \Box\phi, \Box^*\psi, \Box^*\Box\phi : (\vee L) :$
  - $S_3^2 = \phi, \Box^*(\phi \vee \psi) \to \Box\phi, \Box^*\psi, \Box^*\Box\phi : (\Box^{LCF}) :$
    - $S_1^3 = \phi \vee \psi, \Box^*(\phi \vee \psi) \to \phi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi : (\vee L) :$
      - $S_1^4 = \phi, \Box^*(\phi \vee \psi) \to \phi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (axiom)
      - $S_2^4 = \psi, \Box^*(\phi \vee \psi) \to \phi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (terminate)
    - $S_2^3 = \phi \vee \psi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi : (\vee L) :$
      - $S_3^4 = \phi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (terminate)
      - $S_4^4 = \psi, \Box^*(\phi \vee \psi) \to \psi, \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (axiom)
    - $S_3^3 = \phi \vee \psi, \Box^*(\phi \vee \psi) \to \Box^*\phi, \Box^*\psi, \Box^*\Box\phi : (\vee L) :$
      - $S_5^4 = \phi, \Box^*(\phi \vee \psi) \to \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (terminate)
      - $S_6^4 = \psi, \Box^*(\phi \vee \psi) \to \Box^*\phi, \Box^*\psi, \Box^*\Box\phi$ (terminate)
  - $S_4^2 = \psi, \Box^*(\phi \vee \psi) \to \Box\phi, \Box^*\psi, \Box^*\Box\phi : (\Box^{LCF}) :$
    - S_4^3 same as S_1^3
    - S_5^3 same as S_2^3
    - S_6^3 same as S_3^3

Such a tree has the maximal height 5 and consists of 22 sequents. Rule $(\Box^{LCF})$ was applied at most 2 times in any branch. According to the Lemma 3.3.1, rule $(\Box^{LCF})$

can be applied at most 4 times in any branch. In our case, we get only 2 modal rule applications, since we have marked three modal operators at the same time.

For the comparison, the sequent $S_1^0 = \Box(\phi \vee \psi) \to \Box\psi, \Box\Box\phi$ tree obtained by the sequent calculus $KD45_{wf}$ has maximal height 10 and consists even of 127 sequents. There is a branch, where rule $(\Box^W)$ was applied 5 times. Moreover, we have to mention that, during tree construction in the sequent calculus $KD45_{wf}$, we have to test the sequents placed below in a tree to detect loop existence, while in the sequent calculus $KD45_{lcf}$, such a check is not needed at all.

Lemma 3.3.4 There exists sequent with length l, for which decision algorithm for the sequent calculus $KD45_{lcf}$ requires $O(l^3)$ space.

Proof.

To prove this lemma, we construct the sequent (set of the sequents), which has an inference tree T with at least one branch having $height(T) = O(l^2)$. We define formulas $F_1, F_2, \dots, F_k$ as follows:
$F_1 = \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1$,
$F_2 = \neg\Box(\neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1$,
$F_3 = \neg\Box(\neg(\neg\Box(\neg(\neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3) \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1$,
$\dots$
$F_k = \neg\Box(\neg(\neg\Box(\dots\neg(\neg\Box(\psi_k \vee \phi_k) \vee \Box\phi_k)\dots \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1$.

Formula F_{j+1} is obtained from formula F_j by replacing subformula ψ_j with a new subformula $\neg(\neg\Box(\psi_{j+1} \vee \phi_{j+1}) \vee \Box\phi_{j+1})$.

In the sequent $\to F_k$ inference tree T, we have one branch, which is obtained by applying the following rules in the defined order (in the bottom-up direction):
$(\vee R), (\neg R), (\Box^{LCF})$,
$(\vee L), (\neg L), (\vee R), (\neg R), (\Box^{LCF})$,
$(\vee L), (\neg L), (\vee R), (\neg R), (\vee L), (\neg L), (\vee R), (\neg R), (\Box^{LCF})$,
$\dots$
$(\vee L), (\neg L), (\vee R), (\neg R), \dots, (\vee L), (\neg L), (\vee R), (\neg R), (\Box^{LCF})$.

So, in this branch, there are $3 + (1 \cdot 4 + 1) + (2 \cdot 4 + 1) + \dots + ((k-1) \cdot 4 + 1) =$ $= 3 + 4 \cdot \frac{k \cdot (k-1)}{2} + 1 \cdot (k-1) = 2 \cdot k^2 - k + 2$ rules applied. And, therefore, sequent $\to F_k$ inference tree T $height(T) \geq 2 \cdot k^2 - k + 2$.

Sequent $\to F_k$ has length $l = 8 \cdot k$, because $len(\to F_k) = len(\to F_{k-1}) - 1 + 9 =$ $= len(\to F_{k-1}) + 8$.

Sequent $\to F_k$ inference tree T height is at least $2 \cdot k^2 - k + 2$. In other words, sequent $\to F_k$ inference tree $height(T) \geq 2 \cdot (\frac{l}{8})^2 - \frac{l}{8} + 2$, and maximum height is $O(l^2)$.

According to the Lemma 3.3.3, we use $O(l)$ space to store one sequent in the stack, and, therefore, we use $O(l^3)$ space for the sequent $\to F_k$ derivation.

Analogous complexity results are obtained for the multimodal $KD45_n$ logic. These results are published in [9].

Example 3.3.3 We show inference trees for the sequents $\to F_1, \to F_2, \to F_3$ (taken from the Lemma 3.3.4 proof).

Sequent $\to F_1 = \to \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1$ inference tree is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\ominus}{\Box^*(\psi_1 \vee \phi_1), \psi_1 \to \Box^*\phi_1, \phi_1} \qquad \overset{\oplus}{\phi_1, \ldots \to \phi_1, \ldots}}{\Box^*(\psi_1 \vee \phi_1), \psi_1 \vee \phi_1 \to \Box^*\phi_1, \phi_1}\,(\vee L)}{\Box(\psi_1 \vee \phi_1) \to \Box\phi_1}\,(\Box^{LCF})}{\to \neg\Box(\psi_1 \vee \phi_1), \Box\phi_1}\,(\neg R)}{\to \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1}\,(\vee R)}{\to F_1}$$

We denote $\psi_1 = \neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2)$. Then formula
$F_2 = \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1 = \neg\Box(\neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1$
and its inference tree is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_1 \vee \phi_1, \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2}}{\Box^*(\psi_1 \vee \phi_1), \Box(\psi_2 \vee \phi_2) \to \Box^*\phi_1, \phi_1, \Box\phi_2}\,(\Box^{LCF})}{\Box^*(\psi_1 \vee \phi_1) \to \Box^*\phi_1, \phi_1, \neg\Box(\psi_2 \vee \phi_2), \Box\phi_2}\,(\neg R)}{\Box^*(\psi_1 \vee \phi_1) \to \Box^*\phi_1, \phi_1, \neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2}\,(\vee R)}{\Box^*(\psi_1 \vee \phi_1), \neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) \to \Box^*\phi_1, \phi_1}\,(\neg L)}{\Box^*(\psi_1 \vee \phi_1), \psi_1 \to \Box^*\phi_1, \phi_1} \qquad \overset{\oplus}{\phi_1 \to \phi_1}}{\Box^*(\psi_1 \vee \phi_1), \psi_1 \vee \phi_1 \to \Box^*\phi_1, \phi_1}\,(\vee L)}{\Box(\psi_1 \vee \phi_1) \to \Box\phi_1}\,(\Box^{LCF})}{\to \neg\Box(\psi_1 \vee \phi_1), \Box\phi_1}\,(\neg R)}{\to \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1}\,(\vee R)}{\to F_2}$$

We denote $\psi_2 = \neg(\neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3)$, and
$\psi_1 = \neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) = \neg(\neg\Box(\neg(\neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3) \vee \phi_2) \vee \Box\phi_2)$.
Then formula
$F_3 = \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1 = \neg\Box(\neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1 =$
$\neg\Box(\neg(\neg\Box(\neg(\neg\Box(\neg(\neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3) \vee \phi_2) \vee \Box\phi_2) \vee \phi_2) \vee \Box\phi_2) \vee \phi_1) \vee \Box\phi_1$
and its inference tree is:

$$\begin{array}{cl}
\vdots & \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \Box^*(\psi_3 \vee \phi_3), \ldots \to \Box^*\phi_1, \Box^*\phi_2, \Box^*\phi_3, \ldots & \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \Box(\psi_3 \vee \phi_3) \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2, \Box\phi_3 & (\Box^{LCF}) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2) \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2, \neg\Box(\psi_3 \vee \phi_3), \Box\phi_3 & (\neg R) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2) \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2, \neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3 & (\vee R) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \neg(\neg\Box(\psi_3 \vee \phi_3) \vee \Box\phi_3) \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & (\neg L) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & \\ \hline
\uparrow & \\
\uparrow \qquad \oplus & \\
\uparrow \qquad \phi_2 \to \phi_2 & \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & (\vee L) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2, \neg\Box(\psi_2 \vee \phi_2) & (\neg R) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2, \neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2 & (\vee R) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2), \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & (\neg L) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_1, \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & \\ \hline
\uparrow & \\
\uparrow \quad \oplus & \\
\uparrow \ \phi_1 \to \phi_1 & \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box^*(\psi_2 \vee \phi_2), \psi_1 \vee \phi_1, \psi_2 \vee \phi_2 \to \Box^*\phi_1, \Box^*\phi_2, \phi_1, \phi_2 & (\vee L) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \Box(\psi_2 \vee \phi_2) \to \Box^*\phi_1, \phi_1, \Box\phi_2 & (\Box^{LCF}) \\ \hline
\Box^*(\psi_1 \vee \phi_1) \to \Box^*\phi_1, \phi_1, \neg\Box(\psi_2 \vee \phi_2), \Box\phi_2 & (\neg R) \\ \hline
\Box^*(\psi_1 \vee \phi_1) \to \Box^*\phi_1, \phi_1, \neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2 & (\vee R) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \neg(\neg\Box(\psi_2 \vee \phi_2) \vee \Box\phi_2) \to \Box^*\phi_1, \phi_1 & (\neg L) \\ \hline
\Box^*(\psi_1 \vee \phi_1), \psi_1 \to \Box^*\phi_1, \phi_1 \qquad \overset{\oplus}{\phi_1 \to \phi_1} & \\ \hline
\Box^*(\psi_1 \vee \phi_1), \psi_1 \vee \phi_1 \to \Box^*\phi_1, \phi_1 & (\vee L) \\ \hline
\Box(\psi_1 \vee \phi_1) \to \Box\phi_1 & (\Box^{LCF}) \\ \hline
\to \neg\Box(\psi_1 \vee \phi_1), \Box\phi_1 & (\neg R) \\ \hline
\to \neg\Box(\psi_1 \vee \phi_1) \vee \Box\phi_1 & (\vee R) \\ \hline
\to F_3 &
\end{array}$$

Chapter 4

Sequent Calculus With an Efficient Loop-check for Branching Time Logic

In this chapter, branching time logic is discussed. Temporal logics are used to represent time. Linear and branching time logics are the most popular in agents world. Usually, BDI model uses branching time logic for time representation. In this chapter, new weak free sequent calculus and sequent calculus with an efficient loop-check for branching time logic are introduced. Both new sequent calculi create the same inference trees, and efficiency was gained by restricted loop-check used. In the last section, complexity results for the efficient loop-check are also presented.

In this chapter, branching time logic with until operators is researched. Therefore, only logical operators ($\neg$, $\vee$, $\&$) and modal operators $\circ$, A, E, A^{α}_{β}, E^{α}_{β} are used. Formulas, those contain other modal operators are not well-formed formulas for branching time logic.

4.1 Temporal Logics

In this section, the most popular temporal logics are presented. Linear and branching time logics with always and until operators are discussed. Calculi for known temporal logics are presented. Hilbert style axiomatization, sound and complete sequent calculus for branching time logic with until operators are given.

Temporal logics are used to express time flow in the different agent systems. Normally, time is discrete and defined as infinite sequence of time points $t_1, t_2, t_3, \ldots$. t_1 stands for the current time point, t_2 stands for the next time point, and so on. There are two big groups of the temporal logics: linear time logic and branching time logic. In the Linear Time Logic (LTL), there exists only one possible future (see Figure 4.1). In the branching time logic, there may exist several different futures (see Figure 4.2). Since branching time logic creates trees (of the time points), branching time logic is

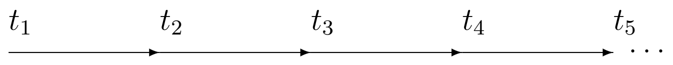


Figure 4.1: Linear time logic.

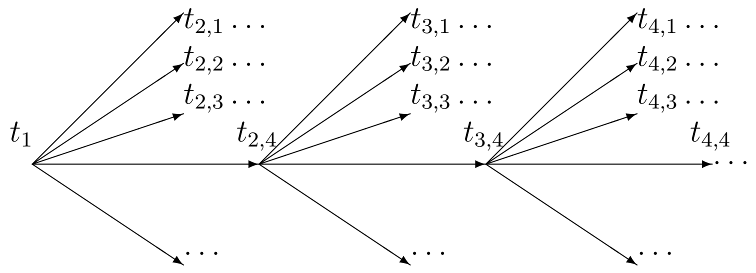


Figure 4.2: Branching time logic.
Here, $t_{2,1}, t_{2,2}, t_{2,3}, \ldots$ stands for the different possible next future points and so on.

named as Computational Tree Logic (CTL).

Temporal logics use modalities to express next time ($\circ$), and modalities to express later future: always and sometimes modalities ($\Box, \Diamond$), or until modalities ($A(\phi \cup \psi), E(\phi \cup \psi)$). Modalities meanings are the following:

- $\circ\phi$ means 'ϕ is true in the next time point'.
- $\Box\phi$ means 'ϕ is true in every future point t'.
- $\Diamond\phi$ means 'there exists future point t, that ϕ is true in the time point t'.
- $A(\phi \cup \psi)$ means 'for every time path $t_1, t_2, \ldots$, there exists time point t, that ϕ is true in every time point t', $t_1 \leq t' < t$ and ψ is true in the time point t'.
- $E(\phi \cup \psi)$ means 'there exists a path $t_1, t_2, \ldots$ and time point t, that ϕ is true in every time point t', $t_1 \leq t' < t$ and ψ is true in the time point t'.

There are other not so popular temporal logics. For example, temporal logics with the past ([34]), temporal logics with intervals ([19]), temporal logic with gaps([3]) and other variations, those are applicable for special scopes.

For the BDI logic usually until modalities are used. There is known axiomatization for CTL (branching time logic) expressed as Hilbert style calculus ([22]). Modal operator AX is the same $\circ$ operator.

Definition 4.1.1 Hilbert type calculus for CTL logic (branching time logic) is calculus with classical non modal axioms, modal axioms as follows:

- $EX\, true\ \&\ AX\, true$,
- $AG(\xi \rightarrow (\neg\psi\ \&\ EX\xi)) \rightarrow (\xi \rightarrow \neg A(\phi \cup \psi)$,
- $AG(\xi \rightarrow (\neg\psi\ \&\ EX\xi)) \rightarrow (\xi \rightarrow \neg AF\psi)$,
- $AG(\xi \rightarrow (\neg\psi\ \&\ (\phi \rightarrow AX\xi))) \rightarrow (\xi \rightarrow \neg E(\phi \cup \psi))$,
- $AG(\xi \rightarrow (\neg\psi\ \&\ AX\xi)) \rightarrow (\xi \rightarrow \neg EF\psi)$,
- $AG(\phi \rightarrow \psi) \rightarrow (EX\phi \rightarrow EX\psi)$,

and rules:

$$\frac{\phi, \quad \phi \rightarrow \psi}{\psi}, \qquad \frac{\phi}{AG\,\phi}.$$

Here

- $EF\phi \equiv E(true \cup \phi)$,
- $AG\phi \equiv \neg EF\neg\phi$,
- $AF\phi \equiv A(true \cup \phi)$,
- $EG\phi \equiv \neg AF\neg\phi$,
- $EX(\phi \vee \psi) \equiv EX\phi \vee EX\psi$,
- $AX\phi \equiv \neg EX\neg\phi$,
- $E(\phi \cup \psi) \equiv \psi \vee (\psi\ \&\ EXE(\phi \cup \psi))$,
- $A(\phi \cup \psi) \equiv \psi \vee (\psi\ \&\ AXA(\phi \cup \psi))$.

Resolution methods for linear time logic and branching time logic may be found in [12]. Tableau method for the fragment of the CTL may by found in [42].

There is known sequent calculus for linear temporal logic with until operator, which is cut free and invariant free sequent calculus ([23]). Another cut free and invariant free sequent calculus for branching time temporal logic with until operator may be find in [39] (as a special fragment of the presented sequent calculus for BDI logic). Both calculi requires loop-check to get decision procedure. In the first paper ([23]), loop leads only to non derivable sequent. In the second paper ([39]), loop leads to non derivable sequent or to special loop-axiom. In this chapter, we define restrictions those are valid for the both loop types.

In [10], there is presented sequent calculus with restricted loop-check for linear temporal logic with modal operators next and always. In this chapter, we concentrate on the branching time temporal logic with modal operators: $\circ$ ('next'), $A(\phi \cup \psi)$ ('in all futures until') and $E(\phi \cup \psi)$ ('in at least one future until').

We have sequent calculus for branching time temporal logic, which uses loop-check ([39]). First of all, we define modal rules used in this sequent calculus.

Definition 4.1.2 Sequent calculus rule $(\circ)$ is:

$$\frac{\Gamma \to \Theta}{\circ\Gamma \to \circ\Theta} \quad (\circ)$$

- Θ - is empty or only one formula.
- Γ - set of the formulas obtained from $\circ\Gamma$ by removing the most outer $\circ$ occurence.

Definition 4.1.3 Sequent calculus rules (AU-L), (AU-R), (EU-L), (EU-R) are:

$$\frac{\psi, \Gamma \to \Delta \quad \phi, \circ A(\phi \cup \psi), \Gamma \to \Delta}{A(\phi \cup \psi), \Gamma \to \Delta} \quad \text{(AU-L)}$$

$$\frac{\Gamma \to \psi, \phi, \Delta \quad \Gamma \to \psi, \circ A(\phi \cup \psi), \Delta}{\Gamma \to A(\phi \cup \psi), \Delta} \quad \text{(AU-R)}$$

$$\frac{\psi, \Gamma \to \Delta \quad \phi, \neg \circ \neg E(\phi \cup \psi), \Gamma \to \Delta}{E(\phi \cup \psi), \Gamma \to \Delta} \quad \text{(EU-L)}$$

$$\frac{\Gamma \to \psi, \phi, \Delta \quad \Gamma \to \psi, \neg \circ \neg E(\phi \cup \psi), \Delta}{\Gamma \to E(\phi \cup \psi), \Delta} \quad \text{(EU-R)}$$

Definition 4.1.4 We say, that sequent S' is a loop-axiom if there exists sequent S satisfying the following conditions:

- $S \rightsquigarrow S'$ is a loop in the inference tree.
- Between sequents S and S', there exists such a rule (AU-L) or (EU-L) application, that its right premise is sequent S' ancestor.

Definition 4.1.5 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak)$ and modal rules $(\circ)$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus PTL_{init} .

We say, that PTL_{init} is an initial sequent calculus for branching time logic. This sequent calculus contains non invertable (also non semi-invertable) rule $(Weak)$ and semi-invertable rule $(\circ)$. Therefore, at least some derivation tactics is necessary to get decidability.

Theorem 4.1.1 Sequent calculus PTL_{init} is sound and complete calculus for branching time logic with until operator.

Proof.

The proof is presented by N. NIDE and T. Shiro in [39] as the fragment of the BDI logic.

4.2 Sequent Calculus With an Efficient Loop-check for Branching Time Logic

In this section, weak free sequent calculus for branching time temporal logic with until operator is presented. Some properties for the weak free sequent calculus are proven. These properties was used to construct sequent calculus with an efficient loop-check. Efficient loop-check technique was obtained by radically restricted the count of the sequents, for those loop-check is performed, and the count of the tested (during loop-check) sequents.

We prove that we can use not all but only several special sequents from the inference tree to use in the loop-check. We use indexes to discover these special sequents in the sequent calculus. These restrictions let us to get an efficient decision procedure based on the introduced sequent calculus.

There is known decision procedure (described in [39]) for the sequent calculus PTL_{init} . Unfortunately, this procedure uses direct loop-check technique to detect non derivable sequent, or to detect loop-axiom.

In other words, sequent calculus PTL_{init} deals with two types of the loops. One is a loop-axiom (see Definition 4.1.4) which may lead an initial sequent to be derivable. For this type of the loop we use term loop-axiom. Other is a simple loop, which is not a loop-axiom, and leads an initial sequent to be non derivable. For this type of the loop we use term 'non derivable' loop. We use term loop to denote both types of the loop. All presented restrictions for the loop-check are applied for the both loop types.

Now we introduce sequent calculus PTL_{wf} which uses only invertable or semi-invertable rules. To get such a sequent calculus, we use primary sequents.

Definition 4.2.1 Sequent S is a primary if S has the shape $\Sigma, \circ\Gamma \rightarrow \Pi, \circ\Delta$ and formula sets Σ, Π contains only propositional variables, $\Sigma \cap \Pi = \emptyset$.

Definition 4.2.2 We say, that sequent $\circ\Gamma \rightarrow \circ\Delta$ is a strict-primal sequent for branching time logic.

Definition 4.2.3 Sequent calculus rule $(Weak^{*}_{PTL})$ is:

$$\frac{\circ\Gamma \rightarrow \circ\Delta}{\Sigma, \circ\Gamma \rightarrow \Pi, \circ\Delta} \quad (Weak^{*}_{PTL})$$

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Simple speaking, rule $(Weak^{*}_{PTL})$ is just a rule $(Weak)$ which is restricted to be used only for primal sequent. Rule $(Weak^{*}_{PTL})$ do not delete any modalized formula. Therefore, rule $(Weak^{*}_{PTL})$ is invertable rule in branching time logic, since Σ and Π contains only propositional variables.

Definition 4.2.4 Sequent calculus rule $(\circ^{or})$ is:

$$\frac{\Gamma \rightarrow \phi_1 \quad || \quad \Gamma \rightarrow \phi_2 \quad || \dots || \quad \Gamma \rightarrow \phi_n}{\circ\Gamma \rightarrow \circ\phi_1, \circ\phi_2, \dots, \circ\phi_n} \quad (\circ^{or})$$

- The case $n = 0$ is allowed and then rule transforms into: $\frac{\Gamma \rightarrow}{\circ\Gamma \rightarrow} \quad (\circ^{or})$
- Γ - set of the formulas obtained from $\circ\Gamma$ by removing the most outer $\circ$ occurence.

It is obviously, that rule $(\circ^{or})$ is semi-invertable rule.

Rule $(Weak^*_{PTL})$ may be applied only for primal sequent and its premise is always some strict-primal sequent. Rule $(\circ^{or})$ may be applied only for strict-primal sequent. We define new rule $(\circ^{W})$ which is just a concatenation of the rules $(Weak^*_{PTL})$ and $(\circ^{or})$:

Definition 4.2.5 Sequent calculus rule $(\circ^{W})$ is:

$$\frac{\Gamma \rightarrow \phi_1 \quad || \quad \Gamma \rightarrow \phi_2 \quad || \dots || \quad \Gamma \rightarrow \phi_n}{\Sigma, \circ\Gamma \rightarrow \Pi, \circ\phi_1, \circ\phi_2, \dots, \circ\phi_n} \quad (\circ^{W})$$

- Σ, Π contains only propositional variables and $\Sigma \cap \Pi = \emptyset$.
- The case $n = 0$ is allowed and then rule transforms into: $\frac{\Gamma \rightarrow}{\Sigma, \circ\Gamma \rightarrow \Pi} \quad (\circ^{W})$
- Γ - set of the formulas obtained from $\circ\Gamma$ by removing the most outer $\circ$ occurence.

Simple speaking, rule $(\circ^{W})$ may be applied only for primary sequents.

Definition 4.2.6 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \rightarrow \phi, \Delta$, logical rules, and modal rules $(\circ^{W})$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus PTL_{wf} .

We say, that PTL_{wf} is a weak free sequent calculus for branching time temporal logic. This sequent calculus contains invertable rules and one semi-invertable rule $(\circ^{W})$.

Theorem 4.2.1 Sequent S is derivable in the sequent calculus PTL_{init} if and only if sequent S is derivable in the sequent calculus PTL_{wf} .

Proof.

If a sequent S is derivable in the PTL_{init} , then we have a derivation tree, which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (AU-L), (AU-R), (EU-L), (EU-R), $(\circ)$, and $(Weak)$. First, we have to eliminate every rule $(Weak)$ application from the inference tree. There are the following cases of the rule $(Weak)$ application:

1. The premise sequent of the rule $(Weak)$ application is an axiom. Then we can delete such a rule $(Weak)$ application at all, since the conclusion sequent of the rule $(Weak)$ application is also an axiom.

2. Rule $(Weak)$ is consecutively applied two times. Then we can transform such a part of the tree to use only one application of the rule $(Weak)$. For this purposes we use the transformation $Trans_1$ as in the proof of the Theorem 3.2.1.

3. Rule $(Weak)$ and rule R are consecutively applied (in the bottom-up direction) (rule R is one of the $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (AU-L), (AU-R), (EU-L), (EU-R)). We can change the order of these rules applications (rule $(Weak)$ will appear above rule R application). For the logical rules we use transformation analogous to the transformation $Trans_2$ as in the proof of the Theorem 3.2.1. For the rule (AU-R), we perform transformation $Trans_5$:

$$\dfrac{\dfrac{\dfrac{\dots}{\Gamma \to \psi, \phi, \Delta} \quad \dfrac{\dots}{\Gamma \to \psi, \circ A(\phi \cup \psi), \Delta}}{\Gamma \to A(\phi \cup \psi), \Delta}\ \text{(AU-R)}}{\Gamma, \Gamma' \to A(\phi \cup \psi), \Delta, \Delta'}\ (Weak)$$

$$\Downarrow \qquad (Trans_5)$$

$$\dfrac{\dfrac{\dfrac{\dots}{\Gamma \to \psi, \phi, \Delta}}{\Gamma, \Gamma' \to \psi, \phi, \Delta, \Delta'}\ (Weak) \quad \dfrac{\dfrac{\dots}{\Gamma \to \psi, \circ A(\phi \cup \psi), \Delta}}{\Gamma, \Gamma' \to \psi, \circ A(\phi \cup \psi), \Delta, \Delta'}\ (Weak)}{\Gamma, \Gamma' \to A(\phi \cup \psi), \Delta, \Delta'}\ \text{(AU-R)}$$

Transformation $Trans_5$ is used for the rule (AU-R). The analogous transformation can be also applied for the rules (AU-L), (EU-L), (EU-R). In such a case, we can raise the rule $(Weak)$ application by one step in a tree.

4. Rules $(Weak)$ and $(\circ)$ are consecutively applied (in the bottom-up direction). In this case, we can do everything rather the same as in the Case 3. The difference is that during the rule $(Weak)$ raising we change rule $(\circ)$ application by the new rules $Weak^*_{PTL}$ and $\circ^{or}$ applications (i.e. by the one rule $(\circ^W)$ application).

   We denote the main formula of the rule $(\circ)$ application as $\circ\phi_1$ (may be empty). Then we perform transformation $Trans_6$:

$$\dfrac{\dfrac{\dfrac{\dots}{\Gamma \to \phi_1}}{\circ\Gamma \to \circ\phi_1}\ (\circ)}{\Lambda, \circ\Gamma, \circ\Gamma' \to \Pi, \circ\phi_1, \circ\phi_2, \circ\phi_3, \dots, \circ\phi_n}\ (Weak) \qquad (Trans_6)$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\Gamma \to \phi_1}}{\Gamma, \Gamma' \to \phi_1}\ (Weak) \quad || \quad \overset{(redundant)}{\dfrac{\dots}{\Gamma, \Gamma' \to \phi_2}} \quad || \dots || \quad \overset{(redundant)}{\dfrac{\dots}{\Gamma, \Gamma' \to \phi_n}}}{\circ\Gamma, \circ\Gamma' \to \circ\phi_1, \circ\phi_2, \circ\phi_3, \dots, \circ\phi_n}\ (\circ^{or})}{\Lambda, \circ\Gamma, \circ\Gamma' \to \Pi, \circ\phi_1, \circ\phi_2, \circ\phi_3, \dots, \circ\phi_n}\ (Weak^*_{PTL})$$

Sequents $\Gamma, \Gamma' \to \phi_2; \Gamma, \Gamma' \to \phi_3; \dots; \Gamma, \Gamma' \to \phi_n$ are some unused or-branches of the derivation tree, because we get derivation tree (in calculus the PTL_{init}) by choosing formula $\circ\phi_1$ as the main for the rule $(\circ)$ application. In other words, we do not proceed sequents $\Gamma, \Gamma' \to \phi_2, \Gamma, \Gamma' \to \phi_3, \dots, \Gamma, \Gamma' \to \phi_n$, because these branches do not effect derivability of the initial sequent and are redundant.

By applying such a transformations in the bottom-up direction we eliminate all applications of the rule $(Weak)$ from the sequent S derivation tree. If there is left rule $(\circ)$ application in the derivation tree, apply transformation like in the Case 4 to get the rule $(\circ^W)$ application instead of the rule $(\circ)$ application. After these transformations, we get a derivation tree for the sequent calculus PTL_{wf} , therefore, sequent S is also derivable in the calculus PTL_{wf} .

If a sequent S is derivable in the PTL_{wf} , then we have derivation tree, which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (AU-L), (AU-R), (EU-L), (EU-R), $(\circ^W)$. We can apply another transformation to change rule $(\circ^W)$ application into the $(Weak)$ and $(\circ)$ rules applications. After rule $(\circ^W)$ application, we can get n sequents. At least one of them is derivable if the father sequent is derivable. If the initial sequent S is derivable, then we can leave only one branch from or-branches which is derivable in the PTL_{wf} . Suppose that we have chosen such a derivable branches (one for every rule $(\circ^W)$ application) and marked them as derivable. All other or-branches become redundant and, therefore, we mark them as redundant. We use consequently applied rules $(Weak^*_{PTL})$ and $(\circ^{or})$ instead of the rule $(\circ^W)$. Suppose that derivable branch is obtained by taking premise with $\circ\phi_1$ (for the case $n = 0$, $\circ\phi_1$ and ϕ_1 denotes an empty sequents) and apply transformation $Trans_7$:

$$
\dfrac{\dfrac{\dfrac{\overset{(derivable)}{\cdots}}{\Gamma \to \phi_1} \quad || \quad \dfrac{\overset{(redundant)}{\cdots}}{\Gamma \to \phi_2} \quad || \dots || \quad \dfrac{\overset{(redundant)}{\cdots}}{\Gamma \to \phi_n}}{\circ\Gamma \to \circ\phi_1, \circ\phi_2, \dots, \circ\phi_n}\,(\circ^{or})}{\Sigma, \circ\Gamma \to \Pi, \circ\phi_1, \circ\phi_2, \dots, \circ\phi_n}\,(Weak^*_{PTL})
$$

$$
\Downarrow \qquad\qquad (Trans_7)
$$

$$
\dfrac{\dfrac{\dfrac{\overset{(derivable)}{\cdots}}{\Gamma \to \phi_1}}{\circ\Gamma \to \circ\phi_1}\,(\circ)}{\Sigma, \circ\Gamma \to \Pi, \circ\phi_1, \circ\phi_2, \dots, \circ\phi_n}\,(Weak)
$$

By applying such a transformations we eliminate all applications of the rule $(\circ^W)$ from the derivation tree. After these transformations we get a derivation tree for the sequent calculus PTL_{init} , therefore, sequent S is also derivable in the sequent calculus PTL_{init} .

Now we introduce sequent calculus PTL_{ol} which uses loop-check only for the sequents, those are some premises of the rule $(\circ^W)$ application. This modification reduces the number of the checked sequents in the inference tree. We define special type of the loop ($\circ$-loop) which is used in the sequent calculus PTL_{ol} .

Definition 4.2.7 Loop $S \rightsquigarrow S'$ is a $\circ$-loop if there exist primary sequents S_1, S_1' in the inference tree that the following conditions are satisfied:

- S_1 is obtained from the sequent S by the rule $(\circ^W)$ applications (i.e., $\dfrac{S}{S_1}(\circ^W)$),
- S_1' is obtained from the sequent S' by the rule $(\circ^W)$ applications (i.e., $\dfrac{S'}{S_1'}(\circ^W)$).

Definition 4.2.8 We say, that sequent S' is a $\circ$-loop-axiom if there exists sequent S satisfying the following conditions:

- $S \rightsquigarrow S'$ is a $\circ$-loop in the derivation tree.
- $S \rightsquigarrow S'$ is a loop-axiom.

Example 4.2.1 Suppose we have a sequent
$S = \phi_2, \circ A(\phi_1 \cup \psi) \to \circ A((\phi_1 \vee \phi_2) \cup \psi)$.
Sequent S derivation tree in the sequent calculus PTL_{wf} is:

$$\dfrac{\dfrac{\overset{\oplus}{\psi \to \phi_1 \vee \phi_2, \psi} \quad \overset{\oplus}{\psi \to \psi, \circ A((\phi_1 \vee \phi_2) \cup \psi)}}{\psi \to A((\phi_1 \vee \phi_2) \cup \psi)}(\text{AU-L}) \quad \begin{matrix} \dfrac{\begin{matrix} \dfrac{\overset{\oplus}{\phi_1, \circ A(\phi_1 \cup \psi) \to \phi_1, \phi_2, \psi}}{\phi_1, \circ A(\phi_1 \cup \psi) \to \phi_1 \vee \phi_2, \psi}(\vee R) \\ \uparrow \\ \uparrow \\ \uparrow \end{matrix} \quad \dfrac{\overset{(\circ\text{-loop})}{S'' = A(\phi_1 \cup \psi) \to A((\phi_1 \vee \phi_2) \cup \psi)}}{\phi_1, \circ A(\phi_1 \cup \psi) \to \psi \circ A((\phi_1 \vee \phi_2) \cup \psi)}(\circ^W)}{\phi_1, \circ A(\phi_1 \cup \psi) \to A((\phi_1 \vee \phi_2) \cup \psi)}(\text{AU-L}) \\ \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \end{matrix}}{\dfrac{S' = A(\phi_1 \cup \psi) \to A((\phi_1 \vee \phi_2) \cup \psi)}{\phi_2, \circ A(\phi_1 \cup \psi) \to \circ A((\phi_1 \vee \phi_2) \cup \psi)}(\circ^W)}(\text{AU-R})$$

Sequents S' and S'' compose a $\circ$-loop $S' \rightsquigarrow S''$, since sequents S' and S'' are premises of the rule $(\circ^W)$ applications. $\circ$-loop $S' \rightsquigarrow S''$ is also a $\circ$-loop-axiom, since there is rule (AU-L) application between sequents S' and S'' (see Definition 4.1.4). Therefore, sequent S is derivable in the sequent calculus PTL_{wf} , since all inference tree leaves contains axioms (usual or loop-axiom).

Definition 4.2.9 The sequent calculus with $\circ$-loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, and modal rules $(\circ^W)$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus PTL_{ol} .

We say, that PTL_{ol} is a sequent calculus for branching time logic with $\circ$-loops. There is the only difference between the sequent caluclus PTL_{wf} and the sequent calculus PTL_{ol} . New sequent calculus PTL_{ol} uses only $\circ$-loops, and sequent calculus PTL_{wf} uses simple loops (those may be $\circ$-loops and may be not).

Now we prove the Lemma, which will be used to show, that restriction to use only $\circ$-loops do not impact any sequent derivability.

Lemma 4.2.1 If we have a loop $S \rightsquigarrow S'$ in the inference tree in the sequent calculus PTL_{wf} and S' is a loop-axiom, when at least one rule $(\circ^W)$ application exists between sequents S and S'.

Proof.

Since S' is a loop-axiom, then between sequents S and S', there exists such a sequents S_1 and S_2 between S and S', that S_1 is a conclusion and S_2 is a right premise of the rule (AU-L) (or (EU-L)) application (see Definition 4.1.4). See Figure 4.3 for the transparency. Therefore, sequent S_1 contains formula $A(\phi \cup \psi)$ (or $E(\phi \cup \psi)$) and sequent S_2 contains formula $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$ respectively).

Suppose that there is no rule $(\circ^W)$ application between S and S'. Suppose that sequent S_1 contains k formulas having the shape $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$, $\circ \neg E(\phi \cup \psi)$). So, sequent S_2 contains $k+1$ formulas having the shape $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$, $\circ \neg E(\phi \cup \psi)$).

Since there is no rule $(\circ^W)$ application between S and S', sequent S' contains $\geq k+1$ formulas having the shape $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$, $\circ \neg E(\phi \cup \psi)$). Since $S \rightsquigarrow S'$ is a loop, sequent S also contains $\geq k+1$ formulas having the shape $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$, $\circ \neg E(\phi \cup \psi)$).

Since there is no rule $(\circ^W)$ application between S and S', sequent S_1 contains $\geq k+1$ formulas having the shape $\circ A(\phi \cup \psi)$ (or $\neg \circ \neg E(\phi \cup \psi)$, $\circ \neg E(\phi \cup \psi)$).

We get a contradiction (because sequent S_1 contains k such a formulas), and, therefore, at least one rule $(\circ^W)$ application exists between S and S'.

Theorem 4.2.2 Sequent S is derivable in the sequent calculus PTL_{wf} if and only if sequent S is derivable in the sequent calculus PTL_{ol} .

Proof.

If sequent S is derivable in the sequent calculus PTL_{ol} , then it is derivable in the sequent calculus PTL_{wf} , because every derivation tree in the sequent calculus PTL_{ol} is also a derivation tree in the sequent calculus PTL_{wf} (every $\circ$-loop is also a loop and every $\circ$-loop-axiom is also a loop-axiom).

If sequent S is derivable in the sequent calculus PTL_{wf} then we have a derivation tree T in the sequent calculus PTL_{wf} . If some tree T leaf contains an axiom $\phi, \Gamma \rightarrow \phi, \Delta$, then it is also an axiom for the sequent calculus PTL_{ol} .

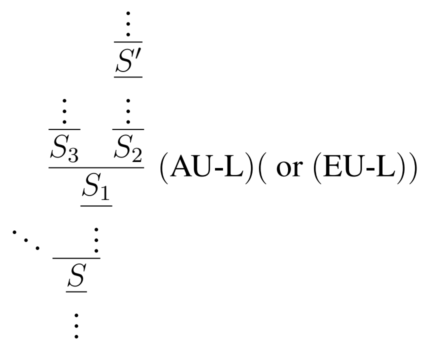


Figure 4.3: Illustration for the proof of the Lemma 4.2.1.
Loop $S \rightsquigarrow S'$, in some inference tree, contains rule (AU-L) (or (EU-L)) application.

If some tree T leaf contains a loop-axiom S_1, which is not $\circ$-loop-axiom. According to the Lemma 4.2.1, there exist sequents S_0', S_0'' between sequents S_0 and S_1 that S_0' is a conclusion and S_0'' is premise of the rule $(\circ^W)$ application.

We can extend tree T into the tree T' by applying the same rules for the sequent S_1 as for sequent S_0 till we reach sequent $S_1'' = S_0''$, or sequent S''' which was a leaf in the derivation tree T.

After modification, we get a tree T' which is still a derivation tree in the sequent calculus PTL_{wf} . Since S_1'', S_0'' contains the same formulas and S_1'', S_0'' are premises of the rule $(\circ^W)$ application, S_1'' is a $\circ$-loop-axiom.

We apply such a modifications for every loop-axiom, which is not $\circ$-loop-axiom in the derivation tree T. Finally, we get a tree T'', those every leaf is an axiom or a $\circ$-loop-axiom. Tree T' is a derivation tree in the sequent calculus PTL_{ol} . Therefore, sequent S is also derivable in the sequent calculus PTL_{ol} .

Now we prove some lemmas to introduce main restrictions to the loop-check used. If formula F is a subformula of G we write $F \subseteq_{sf} G$ (see Definition 2.1.7), if F is proper subformula of G (F has less length then G has) we write $F \subset_{sf} G$ (see Definition 2.1.9). $Ext(F)$ is formula F extended set (see Definition 2.1.6).

Lemma 4.2.2 If $F \subseteq_{sf} G$ and $G \subseteq_{sf} F$, then $Ext(F) = Ext(G)$.

Proof.
The proof goes straightforward from subformula and extended set definitions.

Definition 4.2.10 Formula F is ground in a sequent S if for every formula $G \in S$, formula F is not a proper subformula of G ($F \not\subset_{sf} G$).

It is evident, that, for every formula F in a sequent S, there exists formula G in the sequent S, which is ground and $F \subseteq_{sf} G$, or formula F is ground in the sequent S.

Example 4.2.2 Suppose, that a sequent

$S = A(\phi \cup \circ\neg E(\circ\psi \cup \phi)), \circ\psi \to E(\circ\psi \cup \phi), \circ\phi, \circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$.

Formulas $\circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$ and $\circ\phi$ are ground in the sequent S, because they are not proper subformulas of any other formula in the sequent S.

Formulas $A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$, $E(\circ\psi \cup \phi)$ and $\circ\psi$ are not ground, because they are proper subformulas of $\circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$. It is worth to mention, that

$A(\phi \cup \circ\neg E(\circ\psi \cup \phi)) \subseteq_{sf} \circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$ and

$\circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi)) \subseteq_{sf} A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$, but

$\circ A(\phi \cup \circ\neg E(\circ\psi \cup \phi)) \not\subset_{sf} A(\phi \cup \circ\neg E(\circ\psi \cup \phi))$.

Lemma 4.2.3 If formula F is ground in a sequent S and $F \subseteq_{sf} G$ for some formula $G \in S$, then $F, G \in Ext(F) = Ext(G)$.

Proof.

The proof goes straightforward from the ground formula definition and the Lemma 4.2.2.

Lemma 4.2.4 If $F \subset_{sf} G$, and $G \subseteq_{sf} H$ (or $F \subseteq_{sf} G$, and $G \subset_{sf} H$), then $F \subset_{sf} H$ or formulas $F, G, H \in Ext(F) = Ext(G) = Ext(H)$.

Proof.

It is evident, that $F \subseteq_{sf} H$. Therefore,

- $F \subset_{sf} H$ (satisfies lemma), or
- $H \subseteq_{sf} F$. In this case, we get that $H \subseteq_{sf} F \subset_{sf} G \subseteq_{sf} H$ (or $H \subseteq_{sf} F \subseteq_{sf} G \subset_{sf} H$).

  According to the Lemma 4.2.2, $F, H \in Ext(F) = Ext(H)$, because $H \subseteq_{sf} F$ and $F \subseteq_{sf} H$, and $G, H \in Ext(G) = Ext(H)$, because $H \subseteq_{sf} G$ and $G \subseteq_{sf} H$. And we get the proof.

We have to notice that F has one of the shape: $A(\phi \cup \psi), \circ A(\phi \cup \psi)$, $E(\phi \cup \psi), \neg E(\phi \cup \psi), \circ\neg E(\phi \cup \psi), \neg\circ\neg E(\phi \cup \psi)$ if $Ext(F) \neq \emptyset$.

Lemma 4.2.5 If $S \rightsquigarrow S'$ is a weak $\circ$-loop and T is any sequent inside this weak loop, then any ground formula F in the sequent T is such, that $Ext(F) \neq \emptyset$.

Proof.

Since $S \rightsquigarrow S'$ is a weak $\circ$-loop, there exists such a sequent S'_1, that S'_1 is a conclusion and S' is a premise of the rule $(\circ^W)$ application (see Figure 4.4).

There exists ground formula $G \in S$, that $F \subseteq_{sf} G$, because sequent S is an ancestor of the sequent T. Since $S \rightsquigarrow S'$ is a weak $\circ$-loop, $G \in S'$ and $\circ G \in S'_1$.

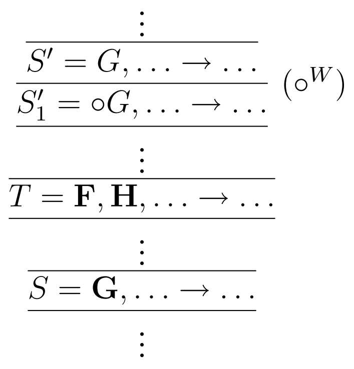

$S' = G, \ldots \to \ldots$
$S'_1 = \circ G, \ldots \to \ldots$
$(\circ^W)$
$T = \mathbf{F}, \mathbf{H}, \ldots \to \ldots$
$S = \mathbf{G}, \ldots \to \ldots$

Figure 4.4: Illustration for the proof of the Lemma 4.2.5.
$\circ$-loop $S \rightsquigarrow S'$ in some inference tree. We use bold characters for ground formulas, those are indeed ground (non bold may be ground or not).

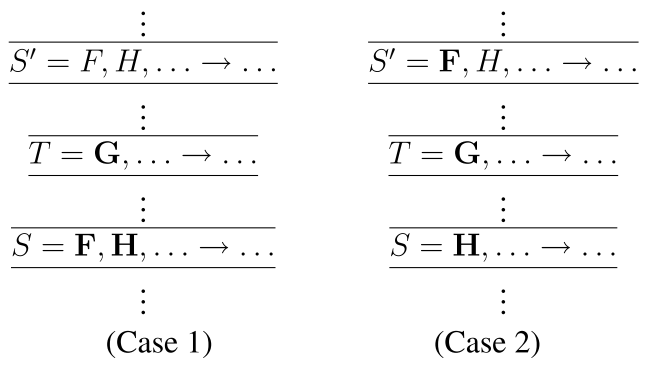

$S' = F, H, \ldots \to \ldots$
$T = \mathbf{G}, \ldots \to \ldots$
$S = \mathbf{F}, \mathbf{H}, \ldots \to \ldots$
(Case 1)
$S' = \mathbf{F}, H, \ldots \to \ldots$
$T = \mathbf{G}, \ldots \to \ldots$
$S = \mathbf{H}, \ldots \to \ldots$
(Case 2)

Figure 4.5: Illustration for the proof of the Lemma 4.2.6.
$\circ$-loop $S \rightsquigarrow S'$ in some inference tree. We use bold characters for ground formulas, those are indeed ground (non bold may be ground or not). Formula F is indeed not ground in the sequent T.

There exists ground formula $H \in T$, that $\circ G \subseteq_{sf} H$, because sequent T is an ancestor of the sequent S'_1 (or S').

We have, that $F \subseteq_{sf} G$, $G \subset_{sf} \circ G$, $\circ G \subseteq_{sf} H$. So, according to the Lemma 4.2.4, $F \subset_{sf} H$ (we get a contradiction, because F is ground formula in T),
or $F, G \in Ext(F) = Ext(G) \neq \emptyset$ (satisfies Lemma).

Lemma 4.2.6 Suppose, that $S \rightsquigarrow S'$ is a weak $\circ$-loop in the derivation tree constructed according to the sequent calculus PTL_{ol} .

If F is ground formula in the sequent S or in the sequent S', then, for every sequent T in a weak $\circ$-loop $S \rightsquigarrow S'$, there exists formula $F' \in Ext(F)$, which is ground in T.

Proof.

Every rule premise contains only subformulas of the rule conclusion.

- Case 1) Ground formula $F \in S$ (see Figure 4.5). Then $F \in S'$, because $S \rightsquigarrow S'$

is a weak $\circ$-loop. We show that in every sequent inside a $\circ$-loop $S \rightsquigarrow S'$ formula $F' \in Ext(F)$ is also ground.

Suppose, that formula F is not ground on some sequent T inside the weak $\circ$-loop $S \rightsquigarrow S'$. So, there exists ground formula $G \in T$, that $F \subset_{sf} G$.

There exists ground formula $H \in S$, that $G \subseteq_{sf} H$, because sequent S is an ancestor of the sequent T. We have that $F \subset_{sf} G$ and $G \subseteq_{sf} H$. According to the Lemma 4.2.4, $F \subset_{sf} H \in S$ (contradiction for F being ground in S), or $G \in Ext(F)$ and G is ground in T (satisfies Lemma). We got that if F is ground in the sequent S, then, for any sequent T in a weak $\circ$-loop $S \rightsquigarrow S'$, there exists formula $F' \in Ext(F)$ which is ground in T.

- Case 2) Ground formula $F \in S'$ (see Figure 4.5). We show that in every sequent inside a weak $\circ$-loop $S \rightsquigarrow S'$ formula F or formula $F' \in Ext(F)$ is also ground.

  Suppose, that formula F is not ground on some sequent T inside the weak $\circ$-loop $S \rightsquigarrow S'$. So, there exists ground formula $G \in T$, that $F \subset_{sf} G$.

  There exists ground formula $H \in S$, that $G \subseteq_{sf} H$, because sequent S is an ancestor of the sequent T. According to the Lemma 4.2.4, $F \subset_{sf} H$ or $G \in Ext(F)$ (satisfies Lemma). In the case $F \subset_{sf} H$, we have that $H \in S'$, because $S \rightsquigarrow S'$ is a weak $\circ$-loop. Formula F is ground in S'. Therefore, $F \not\subset_{sf} H$, and we get a contradiction, because $F \subset_{sf} H$.

  We get that if F is ground in the sequent S', then F or $F' \in Ext(F)$ is ground in every sequent in a weak $\circ$-loop $S \rightsquigarrow S'$.

Corollary 4.2.1 If we have an inference tree satisfying the following conditions:

- sequent T is a conclusion and T' is a premise of some rule R application,
- there exists a ground formula F in the sequent T,
- there is no ground formula $F' \in Ext(F)$ in the sequent T'.

Then sequent T is not inside any weak $\circ$-loop $S \rightsquigarrow S'$.

Proof.
Proof goes straightforward from the Lemma 4.2.5 and the Lemma 4.2.6.

Simple speaking, if some ground formula was deleted during some rule application (in the bottom-up direction), then we do not need to check any sequent below that rule application in order to catch a loop. The main problem is to identify such a situation, because every time we delete some ground formula, at least one new ground formula appears.

Rules (in the calculus PTL_{ol}), those may satisfy above conditions, are (AU-L), (AU-R), (EU-L), (EU-R), then we take left premise, and the rule $(\circ^W)$. So, we can add special indexes for until operators to catch such a situation.

We add different upper-indexes for every different subformula $A(\phi \cup \psi)$, $E(\phi \cup \psi)$ in the initial sequent S. The bottom-index will be a set of indexes.

Definition 4.2.11 If $A(\phi \cup \psi)$ (or $E(\phi \cup \psi)$) is a subformula in the initial sequent S, then $A^i_U(\phi \cup \psi)$ (or $E^i_U(\phi \cup \psi)$) is indexed formula if it satisfies the following conditions:

- $i \in \mathbb{N}$ is a unique number for every different subformula $A(\phi \cup \psi)$ (or $E(\phi \cup \psi)$).
- U is a set of numbers (upper indexes of the other subformulas).
- If there exists subformula F in a sequent S having the shape $A^j_V(\gamma \cup \kappa)$, $E^j_V(\gamma \cup \kappa)$, that $A^i_U(\phi \cup \psi) \subset_{sf} F$, or $E^i_U(\phi \cup \psi) \subset_{sf} F$, then $j \in U$ and $V \subset U$.
- If there is no subformula F in a sequent S having the shape $A^j_V(\gamma \cup \kappa)$, $E^j_V(\gamma \cup \kappa)$, that $A^i_U(\phi \cup \psi) \subset_{sf} F$, or $E^i_U(\phi \cup \psi) \subset_{sf} F$, then $U = \emptyset$.

It means that every ground formula F in any sequent S have until operators with an empty set as its bottom index: $A^i_\emptyset(\phi \cup \psi)$, $\circ A^i_\emptyset(\phi \cup \psi)$, $E^i_\emptyset(\phi \cup \psi)$, $\neg E^i_\emptyset(\phi \cup \psi)$, $\circ\neg E^i_\emptyset(\phi \cup \psi)$, $\neg \circ \neg E^i_\emptyset(\phi \cup \psi)$.

Example 4.2.3 Suppose that we have a sequent S:
$\circ\phi_1 \& \circ A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup (\phi_2 \& \circ \neg A(\phi_3 \cup \phi_1))) \rightarrow A(\phi_2 \cup \circ E(\phi_2 \cup \circ\phi_3))$.

Sequent S with indexed formulas will be:
$\circ\phi_1 \& \circ A^1_\emptyset(\circ\neg E^2_{\{1,4\}}(\phi_2 \cup \circ\phi_3) \cup (\phi_2 \& \circ\neg A^3_{\{1\}}(\phi_3 \cup \phi_1))) \rightarrow A^4_\emptyset(\phi_2 \cup \circ E^2_{\{1,4\}}(\phi_2 \cup \circ\phi_3))$.

There are 2 subformula $E(\phi_2 \cup \circ\phi_3)$ occurrences, so, they have the same upper indexes (2). Every other ÆE subformula has only one occurrence and its upper index is unique.

Formula $E(\phi_2 \cup \circ\phi_3)$ is a proper subformula of $A(\phi_2 \cup \circ E(\phi_2 \cup \circ\phi_3))$ and $\circ\phi_1 \& \circ A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup (\phi_2 \& \circ\neg A(\phi_3 \cup \phi_1)))$. Therefore, its bottom index is $\{1, 3\}$.

Formula $A(\phi_3 \cup \phi_1)$ is a proper subformula of
$\circ\phi_1 \& \circ A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup (\phi_2 \& \circ \neg A(\phi_3 \cup \phi_1)))$, therefore its bottom index is $\{1\}$.

Formulas $\circ\phi_1 \& \circ A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup (\phi_2 \& \circ\neg A(\phi_3 \cup \phi_1)))$ and $A(\phi_2 \cup \circ E(\phi_2 \cup \circ\phi_3))$ are ground formulas in the sequent S. Therefore, they contain $\emptyset$ as their bottom indexes.

Definition 4.2.12 We say that index j is surplus index in the sequent S if:

- there is no subformula in the sequent S with upper-index j,
- there exists such a subformula in the sequent S with bottom-index V, that $j \in V$.

In other words, index j is surplus if it occurs as the bottom-index, but it does not occur as the upper-index in the sequent S.

Definition 4.2.13 If $S = \Gamma \to \Delta$ is a sequent, then we say that $S' = [S]^+ = [\Gamma \to \Delta]^+$ is a index-cleaned sequent if $S' = \Gamma' \xrightarrow{\delta} \Delta'$ and the following conditions are satisfied:

- If there exists such a surplus index j in the sequent S, then S' is a sequent obtained from the sequent S by removing every surplus index. In this case, $\delta = +$.
- If there does not exist any surplus index in the sequent S, then $S' = S$. In this case, δ is empty.

Definition 4.2.14 Sequent calculus rule $(\circ^+)$ is:

$$\frac{[\Gamma \xrightarrow{\circ} \phi_1]^+ \quad || \quad [\Gamma \xrightarrow{\circ} \phi_2]^+ \quad || \dots || \quad [\Gamma \xrightarrow{\circ} \phi_n]^+}{\circ\Gamma \to \circ\phi_1, \dots, \circ\phi_n} \quad (\circ^+)$$

- The case $n = 0$ is allowed and then rule transforms into: $\dfrac{[\Gamma \xrightarrow{\circ}]^+}{\circ\Gamma \to} \quad (\circ^+)$
- Γ - set of the formulas obtained from $\circ\Gamma$ by removing the most outer $\circ$ occurence.

It is obviously, that rule $(\circ^+)$ is semi-invertable rule.

Rule $(Weak^*_{PTL})$ may be applied only for the primal sequent and its premise is always some strict-primal sequent. Rule $(\circ^+)$ may be applied only for strict-primal sequent. We define new rule $(\circ^{RLC})$ which is just a concatenation of the rules $(Weak^*_{PTL})$ and $(\circ^+)$:

Definition 4.2.15 Sequent calculus rule $(\circ^{RLC})$ is:

$$\frac{[\Gamma \xrightarrow{\circ} \phi_1]^+ \quad || \quad [\Gamma \xrightarrow{\circ} \phi_2]^+ \quad || \dots || \quad [\Gamma \xrightarrow{\circ} \phi_n]^+}{\Sigma, \circ\Gamma \to \Pi, \circ\phi_1, \dots, \circ\phi_n} \quad (\circ^{RLC})$$

- Σ, Π contains only propositional variables and $\Sigma \cap \Pi = \emptyset$.
- The case $n = 0$ is allowed and then rule transforms into: $\dfrac{[\Gamma \xrightarrow{\circ}]^+}{\Sigma, \circ\Gamma \to \Pi} \quad (\circ^{RLC})$
- Γ - set of the formulas obtained from $\circ\Gamma$ by removing the most outer $\circ$ occurence.

Definition 4.2.16 Sequent calculus rules $(AU - L^+)$, (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) are:

$$\frac{[\psi, \Gamma \to \Delta]^+ \quad \phi, \circ A_U^j(\phi \cup \psi), \Gamma \to \Delta}{A_U^j(\phi \cup \psi), \Gamma \to \Delta} \quad \text{(AU-L}^+\text{)}$$

$$\frac{[\Gamma \to \psi, \phi, \Delta]^+ \quad \Gamma \to \psi, \circ A_U^j(\phi \cup \psi), \Delta}{\Gamma \to A_U^j(\phi \cup \psi), \Delta} \quad \text{(AU-R}^+\text{)}$$

$$\frac{[\psi, \Gamma \to \Delta]^+ \quad \phi, \neg \circ \neg E_U^j(\phi \cup \psi), \Gamma \to \Delta}{E_U^j(\phi \cup \psi), \Gamma \to \Delta} \quad \text{(EU-L}^+\text{)}$$

$$\frac{[\Gamma \to \psi, \phi, \Delta]^+ \quad \Gamma \to \psi, \neg \circ \neg E_U^j(\phi \cup \psi), \Delta}{\Gamma \to E_U^j(\phi \cup \psi), \Delta} \quad \text{(EU-R}^+\text{)}$$

Simple speaking, if we get a sequent marked by $+$, we know, that some ground formula was just deleted and loop cannot appear here. If we delete some ground formula with some upper-index j, we also delete every index j occurence. Then some ground formula was deleted, we must get some new ground formula. These new ground formulas are formulas containing modalized subformula with an empty set as their bottom index.

Indexes are some kind of the histories, because they store information about applied rules. Histories are one of the possible instruments used to make an efficient loop-check (some calculi with used histories may be found in [29, 38]).

Definition 4.2.17 The sequent calculus with $\circ$-loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, and modal rules $(\circ^{RLC})$, (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) we define sequent calculus PTL_{rlc} .

We say, that PTL_{rlc} is a sequent calculus with an efficient loop-check for branching time logic. This sequent calculus contains invertable rules and one semi-invertable rule $(\circ^{RLC})$.

Theorem 4.2.3 Sequent S is derivable in the sequent calculus PTL_{ol} if and only if sequent S is derivable in the sequent calculus PTL_{rlc} .

Proof.
The proof is straightforward from the Corollary 4.2.1.

Theorem 4.2.4 Sequent calculus PTL_{rlc} is sound and complete sequent calculus for branching time logic with until operator.

Proof.

The proof goes straightforward from the Theorems 4.1.1, 4.2.1, 4.2.2, 4.2.3.

Inference tree construction in the sequent calculus PTL_{rlc} always terminates, because every rule application contains only the finite premises count, and every sequent in the inference tree contains only subformulas of the initial sequent, and derivation is not proceed for every loop-ending sequent.

We obtain an efficient decision procedure for branching time temporal logic if we use sequent calculus PTL_{rlc} with restricted loop-check (for the both loop types). Since sequent calculus PTL_{rlc} uses only $\circ$-loops, we get that:

- loop-check is performed not for every sequent, but only for the sequents, those are marked by $\circ$ (those are premises of some rule $(\circ^{RLC})$ application), because only these sequents may be $\circ$-loop-ending sequents.
- during loop-check, only sequents marked by $\circ$ are tested (those are premises of some rule $(\circ^{RLC})$ application), because only these sequents may be $\circ$-loop-starting sequents.
- during loop-check, only the sequents between the current sequent and the first sequent marked by $+$ (in the top-down direction) are tested, because every sequent marked by $+$ is outside any $\circ$-loop.

Intermediate results related to this chapter are published in [10, 11].

Example 4.2.4 Suppose we have a sequent:

$S = \circ A(\neg A(\phi \cup \neg(\phi \vee \psi)) \cup A(\gamma \cup (\psi \& \kappa))) \to \circ A(\gamma \cup \psi)$.

Sequent S with indexed formulas is:

$S = \circ A^1_{\emptyset}(\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)) \cup A^3_{\{1\}}(\gamma \cup (\psi \& \kappa))) \to \circ A^4_{\emptyset}(\gamma \cup \psi)$.

The following short formula notation is used to denote formulas:

- $G_1 = A^1_{\emptyset}(\neg G_2 \cup G_3) = A^1_{\emptyset}(\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)) \cup A^3_{\{1\}}(\gamma \cup (\psi \& \kappa)))$,
- $G_2 = A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi))$, $G_2' = A^2_{\emptyset}(\phi \cup \neg(\phi \vee \psi))$,
- $G_3 = A^3_{\{1\}}(\gamma \cup (\psi \& \kappa))$, $G_3' = A^3_{\emptyset}(\gamma \cup (\psi \& \kappa))$,
- $G_4 = A^4_{\emptyset}(\gamma \cup \psi)$.

Sequent $S = \circ G_1 \to \circ G_4$ inference tree may be the following:

$$\dfrac{\dfrac{\dfrac{\phi, \circ G_1 \overset{\oplus}{\to} \phi, G_4 \qquad \dfrac{\dfrac{\psi, \circ G_1 \overset{\oplus}{\to} \phi, \gamma, \psi \quad \psi, \circ G_1 \overset{\oplus}{\to} \phi, \psi, \circ G_4}{\uparrow}}{\psi, \circ G_1 \to \phi, A^4_\emptyset(\gamma \cup \psi)}\ (\text{AU-R}^+)}{\dfrac{\phi \vee \psi, \circ G_1 \to \phi, G_4}{\circ G_1 \to \phi, \neg(\phi \vee \psi), G_4}\ (\neg R)}\ (\vee L) \qquad \dfrac{\dfrac{S_2}{\phi \vee \psi, \circ G_1 \to \circ G_2, G_4}}{\circ G_1 \to \neg(\phi \vee \psi), \circ G_2, G_4}\ (\neg R)}{\begin{array}{c}\uparrow\\ \uparrow\end{array}}$$

$$\dfrac{\dfrac{\dfrac{S_1}{G'_3 \xrightarrow{+} G_4}}{A^3_\emptyset(\gamma \cup (\psi \& \kappa)) \xrightarrow{+} G_4} \dfrac{\dfrac{\uparrow}{\circ G_1 \to A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)), G_4}\ (\text{AU-R}^+)}{\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)), \circ G_1 \to G_4}\ (\neg L)}{\dfrac{\dfrac{A(\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)) \cup A^3_{\{1\}}(\gamma \cup (\psi \& \kappa))) \to G_4}{G_1 \xrightarrow{\circ} G_4}}{\circ G_1 \to \circ G_4}\ (\circ^{RLC})}\ (\text{AU-L}^+)$$

Inference tree fragment for the sequent S_1 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\psi, \kappa \overset{\oplus}{\xrightarrow{+}} \gamma, \psi \quad \psi, \kappa \overset{\oplus}{\to} \psi, \circ G_4}{\psi, \kappa \to A^4_\emptyset(\gamma \cup \psi)}\ (\text{AU-R}^+)}{\begin{array}{c}\uparrow\\ \uparrow\end{array}}\ (\& L)}{\psi \& \kappa \to G_4} \qquad \dfrac{\gamma, \circ G'_3 \overset{\oplus}{\xrightarrow{+}} \gamma, \psi \quad \dfrac{\overset{(\circ\text{-loop})\oplus}{G'_3 \xrightarrow{\circ} G_4}}{\gamma, \circ G'_3 \to \psi, \circ G_4}\ (\circ^{RLC})}{\gamma, \circ G'_3 \to A^4_\emptyset(\gamma \cup \psi)}\ (\text{AU-R}^+)}{\dfrac{\dfrac{A^3_\emptyset(\gamma \cup (\psi \& \kappa)) \xrightarrow{+} G_4}{G'_3 \xrightarrow{+} G_4}}{S_1}}\ (\text{AU-L}^+)$$

Inference tree fragment for the sequent S_2 is

$$\dfrac{\dfrac{\dfrac{\dfrac{S_3}{G_1 \xrightarrow{\circ} G_2}}{\phi, \circ G_1 \xrightarrow{+} \gamma, \psi, \circ G_2}\ (\circ^{RLC}) \qquad \dfrac{\dfrac{S_3}{G_1 \xrightarrow{+\circ} G_2} \quad || \quad \overset{(\circ\text{-loop})\oplus}{G_1 \xrightarrow{\circ} G_4}}{\phi, \circ G_1 \to \psi, \circ G_2, \circ G_4}\ (\circ^{RLC})}{\phi, \circ G_1 \to \circ G_2, A^4_\emptyset(\gamma \cup \psi)}\ (\text{AU-R}^+)}{\begin{array}{c}\uparrow\\ \uparrow\\ \uparrow\\ \uparrow\end{array}} \quad \dfrac{\psi, \circ G_1 \overset{\oplus}{\xrightarrow{+}} \gamma, \psi, \circ G_2 \quad \psi, \circ G_1 \overset{\oplus}{\to} \psi, \circ G_2, \circ G_4}{\psi, \circ G_1 \to \circ G_2, A^4_\emptyset(\gamma \cup \psi)}\ (\text{AU-R}^+)$$

$$\dfrac{\dfrac{\cdots}{\phi \vee \psi, \circ G_1 \to \circ G_2, G_4}\ (\vee L)}{S_2}$$

Inference tree fragment for the sequent S_3 is

$$
\begin{prooftree}
\AxiomC{S_8}
\UnaryInfC{$\psi\&\kappa \xrightarrow{+} G_2'$}
\UnaryInfC{$\psi\&\kappa \xrightarrow{+} A^2_\emptyset(\phi \cup \neg(\phi \vee \psi))$}
\AxiomC{S_4}
\UnaryInfC{$\gamma, \circ G_3' \to G_2'$}
\UnaryInfC{$\gamma, \circ G_3' \to A^2_\emptyset(\phi \cup \neg(\phi \vee \psi))$}
\RightLabel{(AU-L$^+$)}
\BinaryInfC{$A^3_\emptyset(\gamma \cup (\psi\&\kappa)) \xrightarrow{+} A^2_\emptyset(\phi \cup \neg(\phi \vee \psi))$}
\UnaryInfC{$\uparrow$}
\end{prooftree}
$$

$$
\begin{prooftree}
\AxiomC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\AxiomC{S_6}
\UnaryInfC{$\circ G_1 \to G_2$}
\RightLabel{$(\neg L)$}
\UnaryInfC{$\neg G_2, \circ G_1 \to G_2$}
\RightLabel{(AU-L$^+$)}
\BinaryInfC{$A^1_\emptyset(\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)) \cup A^3_{\{1\}}(\gamma \cup (\psi\&\kappa))) \xrightarrow{+\circ} A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi))$}
\UnaryInfC{$G_1 \xrightarrow{+\circ} G_2$}
\UnaryInfC{S_3}
\end{prooftree}
$$

Inference tree fragment for the sequent S_4 is

$$
\begin{prooftree}
\AxiomC{$\oplus$}
\noLine
\UnaryInfC{$\phi, \gamma, \circ G_3' \to \phi$}
\AxiomC{S_{10}}
\UnaryInfC{$G_3' \xrightarrow{\circ}$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\psi, \gamma, \circ G_3' \to \phi$}
\RightLabel{$(\vee L)$}
\BinaryInfC{$\phi \vee \psi, \gamma, \circ G_3' \to \phi$}
\RightLabel{$(\neg R)$}
\UnaryInfC{$\gamma, \circ G_3' \xrightarrow{+} \phi, \neg(\phi \vee \psi)$}
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\AxiomC{S_5}
\UnaryInfC{$G_3' \xrightarrow{\circ} G_2'$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\phi, \gamma, \circ G_3' \to \circ G_2'$}
\AxiomC{S_5}
\UnaryInfC{$G_3' \xrightarrow{\circ} G_2'$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\psi, \gamma, \circ G_3' \to \circ G_2'$}
\RightLabel{$(\vee L)$}
\BinaryInfC{$\phi \vee \psi, \gamma, \circ G_3' \to \circ G_2'$}
\RightLabel{$(\neg R)$}
\UnaryInfC{$\gamma, \circ G_3' \to \neg(\phi \vee \psi), \circ G_2'$}
\RightLabel{(AU-R$^+$)}
\BinaryInfC{$\gamma, \circ G_3' \to A^2_\emptyset(\phi \cup \neg(\phi \vee \psi))$}
\UnaryInfC{$\gamma, \circ G_3' \to G_2'$}
\UnaryInfC{S_4}
\end{prooftree}
$$

Inference tree fragment for the sequent S_5 is

$$
\begin{prooftree}
\AxiomC{S_8}
\UnaryInfC{$\psi\&\kappa \xrightarrow{+} G_2'$}
\AxiomC{$\oplus$}
\noLine
\UnaryInfC{$\phi, \gamma, \circ G_3' \to \phi$}
\AxiomC{S_{10}}
\UnaryInfC{$G_3' \xrightarrow{\circ}$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\psi, \gamma, \circ G_3' \to \phi$}
\RightLabel{$(\vee L)$}
\BinaryInfC{$\phi \vee \psi, \gamma, \circ G_3' \to \phi$}
\RightLabel{$(\neg R)$}
\UnaryInfC{$\gamma, \circ G_3' \xrightarrow{+} \phi, \neg(\phi \vee \psi)$}
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\noLine
\UnaryInfC{$\uparrow$}
\AxiomC{($\circ$-loop)$\oplus$}
\noLine
\UnaryInfC{$G_3' \xrightarrow{\circ} G_2'$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\phi, \gamma, \circ G_3' \to \circ G_2'$}
\AxiomC{($\circ$-loop)$\oplus$}
\noLine
\UnaryInfC{$G_3' \xrightarrow{\circ} G_2'$}
\RightLabel{$(\circ^{RLC})$}
\UnaryInfC{$\psi, \gamma, \circ G_3' \to \circ G_2'$}
\RightLabel{$(\vee L)$}
\BinaryInfC{$\phi \vee \psi, \gamma, \circ G_3' \to \circ G_2'$}
\RightLabel{$(\neg R)$}
\UnaryInfC{$\gamma, \circ G_3' \to \neg(\phi \vee \psi), \circ G_2'$}
\RightLabel{(AU-R$^+$)}
\BinaryInfC{$\gamma, \circ G_3' \to A^2_\emptyset(\phi \cup \neg(\phi \vee \psi))$}
\RightLabel{(AU-L$^+$)}
\BinaryInfC{$A^3_\emptyset(\gamma \cup (\psi\&\kappa)) \xrightarrow{\circ} G_2'$}
\UnaryInfC{$G_3' \xrightarrow{\circ} G_2'$}
\UnaryInfC{S_5}
\end{prooftree}
$$

Inference tree fragment for the sequent S_6 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi, \circ G_1 \to \phi} \quad \dfrac{\dfrac{S_7}{G_1 \overset{\circ}{\to}}}{\psi, \circ G_1 \to \phi}\,(\circ^{RLC})}{\phi \vee \psi, \circ G_1 \to \phi}\,(\vee L)}{\circ G_1 \to \phi, \neg(\phi \vee \psi)}\,(\neg R) \quad \dfrac{\dfrac{\dfrac{\overset{(\circ\text{-loop})\oplus}{G_1 \overset{\circ}{\to} G_2}}{\phi, \circ G_1 \to \circ G_2}\,(\circ^{RLC}) \quad \dfrac{\overset{(\circ\text{-loop})\oplus}{G_1 \overset{\circ}{\to} G_2}}{\psi, \circ G_1 \to \circ G_2}\,(\circ^{RLC})}{\phi \vee \psi, \circ G_1 \to \circ G_2}\,(\vee L)}{\circ G_1 \to \neg(\phi \vee \psi), \circ G_2}\,(\neg R)}{\circ G_1 \to A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi))}\,(\text{AU-R}^+)}{\circ G_1 \to G_2}}{S_6}$$

Inference tree fragment for the sequent S_7 is

$$\dfrac{\dfrac{\dfrac{\dfrac{S_{10}}{G_3' \overset{+}{\to}}}{A^3_{\emptyset}(\gamma \cup (\psi \& \kappa)) \overset{+}{\to}} \quad \dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi, \circ G_1 \to \phi} \quad \dfrac{\overset{(\circ\text{-loop})\oplus}{G_1 \overset{\circ}{\to}}}{\psi, \circ G_1 \to \phi}\,(\circ^{RLC})}{\phi \vee \psi, \circ G_1 \to \phi}\,(\vee L)}{\circ G_1 \to \phi, \neg(\phi \vee \psi)}\,(\neg R) \quad \dfrac{\dfrac{\dfrac{\overset{(\circ\text{-loop})\oplus}{G_1 \overset{\circ}{\to} G_2}}{\phi, \circ G_1 \to \circ G_2}\,(\circ^{RLC}) \quad \dfrac{\overset{(\circ\text{-loop})\oplus}{G_1 \overset{\circ}{\to} G_2}}{\psi, \circ G_1 \to \circ G_2}\,(\circ^{RLC})}{\phi \vee \psi, \circ G_1 \to \circ G_2}\,(\vee L)}{\circ G_1 \to \neg(\phi \vee \psi), \circ G_2}\,(\neg R)}{\circ G_1 \to A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi))}\,(\text{AU-R}^+) \\ \hline \uparrow \\ \uparrow \\ \uparrow\end{array}}{\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)), \circ G_1 \to}\,(\neg L)}{A^1_{\emptyset}(\neg A^2_{\{1\}}(\phi \cup \neg(\phi \vee \psi)) \cup A^3_{\{1\}}(\gamma \cup (\psi \& \kappa))) \overset{\circ}{\to}}\,(\text{AU-L}^+)}{\dfrac{G_1 \overset{\circ}{\to}}{S_7}}$$

Inference tree fragment for the sequent S_8 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi, \psi, \kappa \to \phi} \quad \overset{(\text{final})}{\psi, \kappa \to \phi}}{\phi \vee \psi, \psi, \kappa \to \phi}\,(\vee L)}{\psi, \kappa \overset{+}{\to} \phi, \neg(\phi \vee \psi)}\,(\neg R) \quad \dfrac{\dfrac{\dfrac{\dfrac{S_9}{\overset{\circ}{\to} G_2'}}{\phi, \psi, \kappa \to \circ G_2'}\,(\circ^{RLC}) \quad \dfrac{\dfrac{S_9}{\overset{\circ}{\to} G_2'}}{\psi, \kappa \to \circ G_2'}\,(\circ^{RLC})}{\phi \vee \psi, \psi, \kappa \to \circ G_2'}\,(\vee L)}{\psi, \kappa \to \neg(\phi \vee \psi), \circ G_2'}\,(\neg R)}{\psi, \kappa \to A^2_{\emptyset}(\phi \cup \neg(\phi \vee \psi))}\,(\text{AU-R}^+)}{\psi \& \kappa \overset{+}{\to} A^2_{\emptyset}(\phi \cup \neg(\phi \vee \psi))}\,(\& L)}{S_8}$$

Inference tree fragment for the sequent S_9 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\oplus}{\phi \to \phi} \quad \overset{(\text{final})}{\psi \to \phi}}{\phi \vee \psi \to \phi}\,(\vee L)}{\overset{+}{\to} \phi, \neg(\phi \vee \psi)}\,(\neg R) \quad \dfrac{\dfrac{\dfrac{\overset{(\circ\text{-loop})}{\overset{\circ}{\to} G_2'}}{\phi \to \circ G_2'}\,(\circ^{RLC}) \quad \dfrac{\overset{(\circ\text{-loop})}{\overset{\circ}{\to} G_2'}}{\psi \to \circ G_2'}\,(\circ^{RLC})}{\phi \vee \psi \to \circ G_2'}\,(\vee L)}{\to \neg(\phi \vee \psi), \circ G_2'}\,(\neg R)}{\overset{\circ}{\to} A^2_{\emptyset}(\phi \cup \neg(\phi \vee \psi))}\,(\text{AU-R}^+)}{\overset{\circ}{\to} G_2'}}{S_9}$$

Inference tree fragment for the sequent S_{10} is

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{(final)}}{\psi, \kappa \to}\ \ \text{}}{\psi \& \kappa \overset{+}{\to}}\ (\& L) \qquad \dfrac{\dfrac{\dfrac{(\circ\text{-loop})\oplus}{G'_3 \overset{\circ}{\to}}}{A^3_\emptyset(\gamma \cup (\psi \& \kappa)) \overset{\circ}{\to}}}{\gamma, \circ A^3_\emptyset(\gamma \cup (\psi \& \kappa)) \to}\ (\circ^{RLC})}{A^3_\emptyset(\gamma \cup (\psi \& \kappa)) \overset{\circ}{\to}}\ (\text{AU-L}^+)}{\dfrac{G'_3 \overset{\circ}{\to}}{S_{10}}}$$

Since we use short formula notation (using G_1, G_2, G'_2, G_3, G'_3, G_4) to denote formulas, there are used nodes without any rule application. These nodes just rewrites the same sequent using other notation (short notation instead of the full notation, or conversely) and, in fact, they are not the part of the inference tree.

We write $\oplus$ to denote axioms, we write '(final)' to denote final sequents and we write '($\circ$-loop)' to denote loop-ending sequents. There are only 2 non axiom $\circ$-loops: the ones having the loop-ending sequent $\overset{\circ}{\to} G'_2$ placed in the fragment S_9. All other $\circ$-loops are axioms.

We presented full sequent S inference tree. It is obviously, that some tree branches may be left unfinished, since we can show that sequent S is non derivable in the sequent calculus PTL_{rlc} even if some branches are left unfinished.

Presented inference tree is not a derivation tree, and sequent S is non derivable in the sequent calculus PTL_{rlc} . There is only two or-branches: sequents $G_1 \overset{+\circ}{\to} G_2$ and $G_1 \overset{\circ}{\leftarrow} G_4$ placed in the fragment S_2. Sequent $G_1 \overset{\circ}{\leftarrow} G_4$ is treated as derivable, because it is a $\circ$-loop axiom. Sequent $G_1 \overset{+\circ}{\to} G_2$ is treated a non derivable, because it contains final sequents in its inference tree (see fragment S_3) and no or-branches.

Sequent S inference tree contains a lot of the rule $(\circ^{RLC})$ applications and loops. All the count information is placed in the following table:

Sequent	Loop count	Rule $(\circ^{RLC})$ applications	Height	All rule count
S_{10}	1	1	2	3
S_9	2	2	4	7
S_8	4	6	9	22
S_7	4	4	6	13
S_6	6	7	10	21
S_5	7	10	10	34
S_4	15	24	14	79
S_3	35	37	16	125
S_2	51	75	19	255
S_1	1	1	3	5
S	52	77	24	268

Inference tree contains the sequents marked by $\circ$ and the sequents marked by $+$. If sequent S' is marked by $\circ$, then S' is potentially a $\circ$-loop-ending (and a $\circ$-loop-starting) sequent and loop-check must be performed for it. If loop-check is performed for the sequent S', then from zero to several sequents are tested (the sequents those are potentially a $\circ$-loop-starting sequents). Only the ancestor sequents marked by $\circ$ are tested. Moreover, the sequents placed below sequent marked by $+$ are not tested.

Since only sequents marked by $\circ$ are used in the loop-check in the sequent calculus PTL_{rlc} , and only sequents till the first sequent (in the top-down direction) marked by $+$ are tested, we get that loop-check is performed only for 77 sequents and only 58 sequents are tested during loop-check. In this example, only for 2 sequents (marked by $\circ$) during loop-check 2 ancestor sequents are tested. These sequents are sequents $G_1 \overset{\circ}{\to} G_2$ placed in the inference tree fragment S_7 (and 2 tested sequents are $G_1 \overset{\circ}{\to}$ placed in S_7 and $G_1 \overset{+\circ}{\to} G_2$ placed in S_3). For all other sequents (marked by $\circ$) during loop-check only one or zero ancestor sequents are tested. This is the reason, why we get the less count of the tested sequents then there are sequents marked by $\circ$.

For the comparison, loop-check is performed for every sequent from the inference tree in the sequent calculus PTL_{wf} (and we have more then 268 sequents, because there are 268 rules applications). For every sequent from 0 to 24 sequents are tested, because inference tree has maximum height $= 24$. Therefore, restricted loop-check used in the sequent calculus PTL_{rlc} is essentially better then loop-check used in the sequent calculus PTL_{wf} .

4.3 Complexity Results for Sequent Calculus PTL_{rlc}

In this section, complexity results for the loop-check used in the introduced sequent calculus PTL_{rlc} are presented. Applied restrictions radically reduce the time used for the loop-check. Proven upper bound of the tested sequents during loop-check illustrates, that the new sequent calculus with an efficient loop-check for branching time logic improves the decision procedure performance.

We use different modalized formulas counts and formulas modality depth (see Definition 2.1.10) to evaluate sequent calculus PTL_{rlc} complexity.

In this section, if S is an initial sequent, then

- m_a denotes the count of the different *Æ* subformulas (formulas having the shape $A^{\beta}_{\alpha}(\phi \cup \psi)$ or $E^{\beta}_{\alpha}(\phi \cup \psi)$),
- $m_\circ$ denotes the count of the different modalized subformulas having the shape $\circ\phi$, those are not extended *Æ* formulas,
- $m = m_a + m_\circ$,

- k' denotes logical operators count,
- $k = k' + 3 \cdot m_a$ (maximum count of the logical and until rules applied consecutively),
- d denotes initial sequent S modality depth.

The maximum possible count of the logical rules and rules (AU-L^+), (AU-R^+), (EU-L^+), (EU-R^+) applied consecutively in the inference tree is denoted by k. There may be at most m_a rules (AU-L^+), (AU-R^+), (EU-L^+), (EU-R^+) applications. Premise of the rule (EU-L^+) (or (EU-R^+)) application may contain 2 new logical operators $\neg$ (not presented in rule conclusion). Therefore, there may be applied at most $k = k' + m_a + 2 \cdot m_a$ rules (logical and until rules) consecutively.

We have to mention, that m_a counts only *Æ* subformulas (not every extended *Æ* formula) and $m_\circ$ do not include extended *Æ* subformulas. This is illustrated in the Example 4.3.1.

Example 4.3.1 Suppose that we have a sequent S:
$\circ\phi_1 \& \circ A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup \circ\phi_1), \circ E(\phi_2 \cup \circ\phi_3) \to A(\phi_2 \cup \circ E(\phi_2 \cup \circ\phi_3))$.

Sequent S with indexed formulas will be:
$\circ\phi_1 \& \circ A^1_\emptyset(\circ\neg E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \cup \circ\phi_1), \circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \to A^3_\emptyset(\phi_2 \cup \circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$.

There are 3 subformula $E(\phi_2 \cup \circ\phi_3)$ occurrences, so, they have the same upper-index. Subformula $E(\phi_2 \cup \circ\phi_3)$ are proper subformula of $A(\circ\neg E(\phi_2 \cup \circ\phi_3) \cup \circ\phi_1)$ and $A(\phi_2 \cup \circ E(\phi_2 \cup \circ\phi_3))$, and, therefore, its bottom-index is $\{1,3\}$.

Formulas $\circ\phi_1 \& \circ A^1_\emptyset(\circ\neg E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \cup \circ\phi_1)$ and $A^3_\emptyset(\phi_2 \cup \circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$ are ground formulas in the sequent S. Formula $\circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3)$ is not a ground formula, because it is a proper subformula of the formula $A^3_\emptyset(\phi_2 \cup \circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$.

Sequent S contains only $m_a = 3$ different *Æ* subformulas (this count is equal to the maximum upper index). Sequent S contains only $m_\circ = 3$ different modalized subformulas having the shape $\circ\psi$, those are not extended *Æ* formulas: $\circ\phi_1$, $\circ\phi_3$, $\circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3)$.

We have to mention, that formulas
$\circ\neg E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3)$ and $\neg E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \in Ext(E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$,
but $\circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \notin Ext(E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$.

Formulas $\circ\phi_1, \circ\phi_3$ have modality depth $= 1$. Formula $\circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3)$ has modality depth $= 3$. Formula $A^3_\emptyset(\phi_2 \cup \circ E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3))$ has modality depth $= 4$.

Formula $\circ A^1_\emptyset(\circ\neg E^2_{\{1,3\}}(\phi_2 \cup \circ\phi_3) \cup \circ\phi_1)$ has modality depth only $= 3$. $\circ A(\ldots)$ and $\circ\neg E(\ldots)$ are counted as one modality, because they are extended *Æ* formulas (see Definition 2.1.10.).

Therefore, sequent S modality depth is $4 = max(3,3,4)$.

Lemma 4.3.1 Suppose that we have k_i different modalized subformulas (Æ subformulas or subformulas of the shape $\circ\phi$, those are not extended Æ formulas) having modality depth $= i$. If so, then any branch in the sequent S inference tree contains at most $k_1 \cdot k_2 \cdot \ldots \cdot k_d$ rule $(\circ^{RLC})$ applications.

Proof.

If formulas F' and F'' are the same subformula in the sequent S, then any non axiom sequent in the sequent S inference tree contains at most one of the formulas F' and F''.

Suppose that subtree T' is such a sequent S inference tree branch, that it contains the maximum rule $(\circ^{RLC})$ application count.

Suppose, that sequents $S_1, S_2, \ldots, S_n$ and $S'_1, S'_2, \ldots, S'_n$ are such a sequents on the branch T', that the following conditions are satisfied:

- S is an ancestor of S_1 and there is no rule $(\circ^{RLC})$ application between S and S_1,
- sequents $S_1, S_2, S_3, \ldots, S_n$ are non axiom primary sequents,
- Sequents S'_i are such, that S_i is a conclusion and S'_i is a premise of the rule $(\circ^{RLC})$ application (for every $i = 1, 2, \ldots, n$).

We use mathematical induction according modality depth d to prove this fact.

If $d = 1$. Sequent S'_1 do not contain any formula of the shape $\circ\phi$ (there may be left only extended Æ formulas), because $d = 1$.

Suppose that S'_1 contains $k' \leq k_1$ extended Æ formulas, then S'_2 contains $k'' \leq k'$ extended Æ formulas. If $k'' = k'$ then we have a $\circ$-loop $S'_1 \rightsquigarrow S'_2$ (and inference tree construction stops). If $k'' < k'$, then we proceed the same arguments for S'_2 as for S'_1. Since any rule (including $(\circ^{RLC})$) do not increase count of the extended Æ formulas, after k_1 rule $(\circ^{RLC})$ applications we get a sequent without modalized formulas, or we get a $\circ$-loop. Therefore, any branch in the sequent S inference tree contains at most k_1 rule $(\circ^{RLC})$ applications.

Assumption of the mathematical induction: if sequent S modality depth $d' < d$, then sequent S inference tree contains at most $k_1 \cdot k_2 \cdot \ldots \cdot k_{d'}$ rule $(\circ^{RLC})$ applications.

Suppose, that sequent S has modality depth $= d$.

If S' and S'' are sequents on the same inference tree, S' is an ancestor of S'', there is a ground formula G in the sequent S', but there is no ground formula G (or ground $G' \in Ext(G)$) in the sequent S'', then there is no formula G (or $G' \in Ext(G)$) in any sequent S''', which is placed above the sequent S'' (S'' is an ancestor of the sequent S'''). Simply speaking, if we delete ground formula in the sequent S', then we cannot get it above the sequent S', because all above sequents contain only subformulas of the S'.

If formula $\circ\phi$ is not a ground formula in the sequent S_i, then there exists a ground formula G (in the sequent S_i), that $\circ\phi \subset_{sf} G$. Since G also has the shape $\circ\psi$ (sequent S_i contains only propositional variables or only formulas having the shape $\circ\psi$), G modality depth is greater then $\circ\phi$ modality depth (because there exists only one extended Æ subformula $F' \in Ext(F)$ having the shape $\circ\phi$).

If formula F in the sequent S_i has modality depth d, then F is a ground formula in the sequent S_i, because all formulas in the sequent S_i contains only propositional variables (having modality depth $= 0$) or only formulas having the shape $\circ\phi$.

Sequent S contains k_d ground formulas, those has modality depth $= d$ (there may exist other ground formulas, those has modality depth $< d$). Therefore S_1 contains $m \leq k_d$ ground formulas, those have modality depth $= d$. Suppose, that any sequent in $S_1, S_2, S_3, \ldots, S_l$ contains all the ground formulas having modality depth $= d$ (we indicate them with $\circ G_1, \circ G_2, \ldots, \circ G_m$). Since $\circ G_1, \circ G_2, \ldots, \circ G_m$ are the only formulas those have modality depth $= d$, all other modalized formulas in any sequent S_i have modality depth $< d$. Therefore, there exists $l \leq k_1 \cdot k_2 \cdot \ldots \cdot k_{d-1}$ rule $(\circ^{RLC})$ applications between S_1 and S_l.

If we delete one ground formula (having modality depth $= d$) we can have at most l rule $(\circ^{RLC})$ applications till the next ground formula (having modality depth $= d$) deletion.

If some sequent S_p do not contain any formula from $\circ G_1, \circ G_2, \ldots, \circ G_m$, then sequent S'_p modality depth $< d$ and, according assumption, we get that sequent S_p inference tree contains at most $k_1 \cdot k_2 \cdot \ldots \cdot k_{d-1}$ rule $(\circ^{RLC})$ applications.

Therefore, there exist at most $m \cdot l \leq k_d \cdot (k_1 \cdot k_2 \cdot k_3 \cdot \ldots \cdot k_{d-1}) \leq k_1 \cdot k_2 \cdot k_3 \cdot \ldots \cdot k_d$ rule $(\circ^{RLC})$ applications between S_1 and S_n.

According to mathematical induction, any branch in the sequent S inference tree contains at most $k_1 \cdot k_2 \cdot \ldots \cdot k_d$ rule $(\circ^{RLC})$ applications.

Lemma 4.3.2 Suppose that sequent S has modality depth d and contains m different modalized subformulas (Æ subformulas or subformulas of the shape $\circ\phi$, those are not extended Æ formulas), then any branch in the sequent S inference tree has at most $(\frac{m}{d})^d$ rule $(\circ^{RLC})$ applications.

Proof.

Suppose that k_i is the count of the different modalized subformulas (extended Æ subformulas or subformulas of the shape $\circ\phi$) having modality depth $= i$ (for every $i = 1, 2, \ldots, d$). According to the Lemma 4.3.1, any branch in the sequent S inference tree contains at most $k_1 \cdot k_2 \cdot \ldots \cdot k_d$ rule $(\circ^{RLC})$ applications. Since $m = k_1 + k_2 + \ldots + k_d$, any branch in the sequent S inference tree contains

$\leq k_1 \cdot k_2 \cdot \ldots \cdot k_d \leq \underbrace{\frac{m}{d} \cdot \frac{m}{d} \cdot \ldots \cdot \frac{m}{d}}_{d \text{ times}} = (\frac{m}{d})^d$ rule $(\circ^{RLC})$ applications.

These two lemmas (the Lemma 4.3.1 and the Lemma 4.3.2) shows the upper bound for the inference tree height in the sequent calculus PTL_{rlc} . Since sequent calculi PTL_{wf} and PTL_{rlc} constructs the same inference trees, both lemmas are also valid for the sequent calculus PTL_{wf} .

Lemma 4.3.3 Suppose that sequent S has modality depth d, contains m_a different Æ subformulas, $m_\circ$ different modalized subformulas having the shape $\circ\phi$, those are not extended Æ formulas, and k' logical rules. If $k = (k' + 3 \cdot m_a)$ and $m = m_a + m_\circ$, then any branch in the sequent S inference tree has height $\leq k \cdot (\frac{m}{d})^d$.

Proof.

According to the Lemma 4.3.2, any branch in the sequent S inference tree has at most $(\frac{m}{d})^d$ rule $(\circ^{RLC})$ applications. Between two rule $(\circ^{RLC})$ applications, only logical rules and rules (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) are applied. Rules (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) may be applied at most m_a times. If it is the rule (EU-L$^+$) or rule (EU-R$^+$), then premise may contain 2 new logical operations $\neg$. So, between two rule $(\circ^{RLC})$ applications, at most $k' + m_a + 2 \cdot m_a = k' + 3 \cdot m_a = k$ rules may be applied. Therefore, any branch in the sequent S inference tree has height $\leq k \cdot (\frac{m}{d})^d$.

Lemma 4.3.4 Suppose that sequent S has modality depth d and contains m different modalized subformulas (Æ subformulas or subformulas having the shape $\circ\phi$, those are not extended Æ formulas), then any weak $\circ$-loop $S \rightsquigarrow S'$ in the sequent calculus PTL_{rlc} contains at most $(\frac{m}{d})^{d-1}$ rule $(\circ^{RLC})$ applications.

Proof.

According to the Lemma 4.2.6, any weak $\circ$-loop contains the same ground formulas (those are extended Æ formulas). Suppose that k_i is the count of the different modalized subformulas (Æ subformulas or subformulas of the shape $\circ\phi$, those are not extended Æ formulas) having modality depth $= i$ (for every $i = 1, 2, \ldots, d$). Only ground formulas have modality depth $= d$.

Therefore, according to the Lemma 4.3.1 proof, there are at most $k_1 \cdot k_2 \cdot \ldots \cdot k_{d-1}$ rule $(\circ^{RLC})$ applications inside weak $\circ$-loop $S \rightsquigarrow S'$. Since $k_1 + k_2 + \ldots + k_{d-1} < k_1 + k_2 + \ldots + k_d = m$, then any weak $\circ$-loop $S \rightsquigarrow S'$ in the sequent calculus PTL_{rlc} contains at most

$k_1 \cdot k_2 \cdot \ldots \cdot k_{d-1} \leq \underbrace{\frac{m}{d} \cdot \frac{m}{d} \cdot \ldots \cdot \frac{m}{d}}_{d-1 \text{ times}} = (\frac{m}{d})^{d-1}$ rule $(\circ^{RLC})$ applications.

Lemma 4.3.4 says, that, during loop-check, at most $(\frac{m}{d})^{d-1}$ sequents are tested for every sequent inside the sequent S inference tree in the sequent calculus PTL_{rlc} . For the comparison, during loop-check at most $k \cdot (\frac{m}{d})^d$ sequents are tested for every sequent inside the sequent S inference tree in the sequent calculus PTL_{init} . Moreover, in the sequent calculus PTL_{init} , loop-check is performed for every sequent in the sequent S inference tree, and, in the sequent calculus PTL_{rlc} , loop-check is performed only for sequents, those are rule $(\circ^{RLC})$ premises. Therefore, sequent calculus PTL_{rlc} greatly restricts loop-check performed.

To be precise, sequent S inference tree in the sequent calculus PTL_{rlc} contains at most $m^{(\frac{m}{d})^d}$ rule applications. So, loop-check is performed at most $m^{(\frac{m}{d})^d}$ times. According to the Lemma 4.3.4, every loop-check tests at most $(\frac{m}{d})^{d-1}$ sequents. Therefore, during sequent S inference tree construction, in the sequent calculus PTL_{rlc} , there are at most $(\frac{m}{d})^{d-1} \cdot m^{(\frac{m}{d})^d}$ sequents tested for the loop-check.

For the comparison, sequent S inference tree in the calculus PTL_{wf} $(PTL_{init}$) have the same size like sequent S inference tree in the sequent calculus PTL_{rlc} (with some generalization, they create the same inference trees). Every logical rule and rules (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) have at most 2 premises and rule $(\circ^{RLC})$ may have m premises. According to the Lemmas 4.3.2 and 4.3.4, sequent S inference tree contains at most $m^{(\frac{m}{d})^d} \cdot 2^{(k\cdot(\frac{m}{d})^d-(\frac{m}{d})^d)}$ sequents. In the sequent calculus PTL_{wf} $(PTL_{init}$), loop-check is performed for every sequent in the inference tree and every ancestor sequent is tested. Therefore, during sequent S inference tree construction, in the sequent calculus PTL_{wf} $(PTL_{init}$), there are at most $k \cdot (\frac{m}{d})^d \cdot m^{(\frac{m}{d})^d} \cdot 2^{((k-1)\cdot(\frac{m}{d})^d)}$ sequents tested for the loop-check.

Of course there are only the upper bounds for the loop-check, and they are not accessible, but numbers

$(\frac{m}{d})^{d-1} \cdot m^{(\frac{m}{d})^d}$ and $k \cdot (\frac{m}{d})^d \cdot m^{(\frac{m}{d})^d} \cdot 2^{((k-1)\cdot(\frac{m}{d})^d)}$ are more then self-explanatory.

Chapter 5

Sequent Calculi With an Efficient Loop-check for BDI Logics

In this chapter, new sequent calculi for monoagent and multiagent BDI logics are introduced. BDI logic is used to describe agents using their beliefs, desires and intentions. Special modalities are used to represent beliefs, desires, intentions and some modalities to represent time. Therefore, BDI logic is a combination of several modal logics. In this chapter, sequent calculi for the fragments of the BDI logics (introduced in the previous chapters) are combined to get sequent calculi with an efficient loop-check for BDI logics.

In this chapter, monoagent and multiagent BDI logics are researched. For monoagent BDI logic, only logical operators ($\neg$, $\vee$, $\&$) and modal operators $\mathbf{B}$, $\mathbf{D}$, $\mathbf{I}$, $\mathbf{B}^*$, $\circ$, A, E, A^α_β, E^α_β are used. Formulas, those contain other modal operators are not well-formed formulas for monoagent BDI logic. For multiagent BDI logic, only logical operators ($\neg$, $\vee$, $\&$) and modal operators $\mathbf{B}_i$, $\mathbf{D}_i$, $\mathbf{I}_i$, $\mathbf{B}^*_i$, $\circ$, A, E, A^α_β, E^α_β are used. Formulas, those contain other modal operators are not well-formed formulas for multiagent BDI logic.

5.1 Calculi for BDI Logic

In this section, known calculi for monoagent BDI logic are presented. Hilbert style axiomatization, sound and complete sequent calculus for BDI logic are given.

BDI is multimodal logic used to describe agents via their beliefs, desires and intentions. BDI logic is widely described by M. Wooldridge in [52]. Usually, BDI logic is a combination of several modal logics and branching time logic. Therefore, BDI logic deals with several modalities:

- Belief, which is represented by a modal operator $\mathbf{B}$, which is a modality of the modal logic $KD45$. In Chapter 3, we presented a loop-check free sequent calculus for $KD45$ logic.

- Intend, which is represented by a modal operator $\mathbf{I}$, which is a modality of the modal logic KD.
- Desire, which is represented by a modal operator $\mathbf{D}$, which is a modality of the modal logic KD.
- Next and Until modalities are represented by modal operators $\circ$, $A(\phi \cup \psi)$ and $E(\phi \cup \psi)$ those are branching time logic until modalities. In Chapter 4, we presented sequent calculus for branching time logic, which uses an efficient loop-check.

Since, BDI logic uses modalities from $KD45$, KD and branching time logic (CTL), Hilbert style axiomatization is the following:

Definition 5.1.1 Hilbert type calculus for BDI logic is calculus with classical non modal axioms and modal axioms as follows:

- (B-K) $\mathbf{B}(\phi \to \psi) \to (\mathbf{B}\phi \to \mathbf{B}\psi)$,
- (B-D) $\mathbf{B}\phi \to \neg\mathbf{B}\neg\phi$,
- (B-4) $\mathbf{B}\phi \to \mathbf{B}\mathbf{B}\phi$,
- (B-5) $\neg\mathbf{B}\phi \to \mathbf{B}\neg\mathbf{B}\phi$,
- (D-K) $\mathbf{D}(\phi \to \psi) \to (\mathbf{D}\phi \to \mathbf{D}\psi)$,
- (D-D) $\mathbf{D}\phi \to \neg\mathbf{D}\neg\phi$,
- (I-K) $\mathbf{I}(\phi \to \psi) \to (\mathbf{I}\phi \to \mathbf{I}\psi)$,
- (I-D) $\mathbf{I}\phi \to \neg\mathbf{I}\neg\phi$,
- $EX\ true\ \&\ AX\ true$,
- $AG(\xi \to (\neg\psi \& EX\xi)) \to (\xi \to \neg A(\phi \cup \psi)$,
- $AG(\xi \to (\neg\psi \& EX\xi)) \to (\xi \to \neg AF\psi)$,
- $AG(\xi \to (\neg\psi \& (\phi \to AX\xi))) \to (\xi \to \neg E(\phi \cup \psi))$,
- $AG(\xi \to (\neg\psi \& AX\xi)) \to (\xi \to \neg EF\psi)$,
- $AG(\phi \to \psi) \to (EX\phi \to EX\psi)$,

and rules:

$$\frac{\phi, \quad \phi \to \psi}{\psi}, \qquad \frac{\phi}{\mathbf{B}\,\phi}, \qquad \frac{\phi}{\mathbf{D}\,\phi}, \qquad \frac{\phi}{\mathbf{I}\,\phi}, \qquad \frac{\phi}{AG\,\phi}.$$

Here

- $EF\phi \equiv E(true \cup \phi)$,
- $AG\phi \equiv \neg EF \neg\phi$,
- $AF\phi \equiv A(true \cup \phi)$,
- $EG\phi \equiv \neg AF \neg\phi$,
- $EX(\phi \vee \psi) \equiv EX\phi \vee EX\psi$,
- $AX\phi \equiv \neg EX \neg\phi$,
- $E(\phi \cup \psi) \equiv \psi \vee (\psi \& EXE(\phi \cup \psi))$,
- $A(\phi \cup \psi) \equiv \psi \vee (\psi \& AXA(\phi \cup \psi))$,
- modal operator AX stands for $\circ$ operator.

A.S. Rao, M. Georgeff in the work [45] presents decision procedure for multiagent branching time BDI logic. They provide sound and complete axiomatizations and presents tableau-based decision procedure for formula satisfability and validity. According them, complexity of these decision procedures is not greater than the complexity of used temporal logics.

In [39], N. NIDE and T. Shiro presented sequent calculus for BDI logic, which uses branching time logic with until operators for time representation.

Definition 5.1.2 Sequent calculus rule (BEL) is:

$$\frac{\Gamma, \mathbf{B}\Gamma \to \Theta, \mathbf{B}\Theta, \mathbf{B}\Delta}{\mathbf{B}\Gamma \to \mathbf{B}\Theta, \mathbf{B}\Delta} \quad (BEL)$$

- Θ is empty or only one formula.
- Γ - set of the formulas obtained from $\mathbf{B}\Gamma$ by removing the most outer $\mathbf{B}$ occurence.

Rule (BEL) is the same rule as the sequent calculus $KD45_{init}$ modal rule $(\Box)$ for $KD45$ logic used in Chapter 3. Here we use different notation, because $\mathbf{B}$ is only one modality of the modalities used in the BDI logic.

Definition 5.1.3 Sequent calculus rules (INT) and (DES) are:

$$\frac{\Gamma \to \Theta}{\mathbf{I}\Gamma \to \mathbf{I}\Theta}\ (INT) \qquad\qquad \frac{\Gamma \to \Theta}{\mathbf{D}\Gamma \to \mathbf{D}\Theta}\ (DES)$$

- Θ is empty or only one formula.
- Γ - set of the formulas obtained from $\mathbf{I}\Gamma$ or $\mathbf{D}\Gamma$ by removing the most outer $\mathbf{I}$ or $\mathbf{D}$ occurence respectively.

Definition 5.1.4 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak)$ and modal rules (BEL), (INT), (DES), $(\circ)$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus BDI_{init} .

We say, that BDI_{init} is an initial sequent calculus for BDI logic. This sequent calculus contains non invertable (and non semi-invertable) rule $(Weak)$, and semi-invertable rules (INT), (DES), $(\circ)$. Therefore, at least some derivation tactics is necessary to get decidability.

Theorem 5.1.1 Sequent calculus BDI_{init} is sound and complete calculus for BDI logic.

Proof.

The proof was presented by N. NIDE and T. Shiro in [39].

We have sound and complete system for BDI logic based on the sequent calculus. It is a cut free sequent calculus, which uses loop-check to get decidability. Decision procedure based on the sequent calculus BDI_{init} uses inefficient (direct) loop-check (together with special derivation tactics). The authors mentioned, that the purpose was to get soundness and completeness, but efficiency was not the main goal. In this chapter, we present new sequent calculus, which is much more efficient and is equivalent to the sequent calculus BDI_{init} .

5.2 Sequent Calculus With an Efficient Loop-check for BDI Logic

In this section, new sequent calculus with an efficient loop-check for monoagent BDI logic is introduced. Introduced sequent calculus is a combination of the loop-check free sequent calculus for $KD45$ logic introduced in Chapter 3, loop-check free sequent calculus for KD logic (well known calculus) and sequent calculus with an efficient loop-check for branching time logic introduced in Chapter 4.

First of all we present weak free sequent calculus for BDI logic which contains only invertable or semi-invertable rules.

We use the proven fact (in [39]), that any two different loops (in the sequent calculus BDI_{init}) cannot overlap. This fact enables us to apply restrictions, presented for $KD45$ logic and for branching time logic, for BDI logic as well.

Sequent calculus BDI_{wf} (and BDI_{init}) do not use marked modal operator $\mathbf{B}^*$, but such a marked operator is used in the sequent calculus BDI_{elc} (introduced later). Therefore, we also use marked modal operator $\mathbf{B}^*$ in the following definitions, since they are applicable for the both calculi.

Definition 5.2.1 We say that sequent
$\Sigma, \mathbf{B}\Gamma_1, \mathbf{B}^*\Gamma_1', \mathbf{D}\Gamma_2, \mathbf{I}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}\Delta_1, \mathbf{B}^*\Delta_1', \mathbf{D}\Delta_2, \mathbf{I}\Delta_3, \circ\Delta_4$
is a primal sequent for BDI logic if:

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Definition 5.2.2 We say that sequents:

- $\mathbf{B}\Gamma, \mathbf{B}^*\Gamma' \to \mathbf{B}\Delta, \mathbf{B}^*\Delta'$,
- $\mathbf{D}\Gamma \to \mathbf{D}\Delta$,
- $\mathbf{I}\Gamma \to \mathbf{I}\Delta$,
- $\circ\Gamma \to \circ\Delta$,

are strict-primal sequents for BDI logic.

Definition 5.2.3 Sequent calculus rule $(Weak^*)$ is:

$$\frac{\mathbf{B}\Gamma_1 \to \mathbf{B}\Delta_1 \quad || \quad \mathbf{D}\Gamma_2 \to \mathbf{D}\Delta_2 \quad || \quad \mathbf{I}\Gamma_3 \to \mathbf{I}\Delta_3 \quad || \quad \circ\,\Gamma_4 \to \circ\Delta_4}{\Sigma, \mathbf{B}\Gamma_1, \mathbf{D}\Gamma_2, \mathbf{I}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}\Delta_1, \mathbf{D}\Delta_2, \mathbf{I}\Delta_3, \circ\Delta_4} \quad (Weak^*)$$

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Simple speaking, rule $(Weak^*)$ is just a rule $(Weak)$, which is restricted to be used only for primal sequent.

Definition 5.2.4 Sequent calculus rule (BEL^{or}) is:

$$\frac{\Gamma, \mathbf{B}\Gamma \to \phi_1, \mathbf{B}\phi_1, \ldots, \mathbf{B}\phi_n \quad || \quad \ldots \quad || \quad \Gamma, \mathbf{B}\Gamma \to \phi_n, \mathbf{B}\phi_1, \ldots, \mathbf{B}\phi_n}{\mathbf{B}\Gamma \to \mathbf{B}\phi_1, \ldots, \mathbf{B}\phi_n} \quad (BEL^{or})$$

- The case $n = 0$ is allowed and then rule transforms into:
  $\frac{\Gamma, \mathbf{B}\Gamma \to}{\mathbf{B}\Gamma \to} \quad (BEL^{or})$
- Γ - set of the formulas obtained from $\mathbf{B}\Gamma$ by removing the most outer $\mathbf{B}$ occurence.

This is the same rule used for $KD45$ logic (see Definition 3.2.4). In this chapter we deal with several operators, so, we use operator $\mathbf{B}$ instead of $\Box$ and rule (BEL^{or}) instead of the rule $(\Box^{or})$.

Definition 5.2.5 Sequent calculus rules (INT^{or}) and (DES^{or}) are:

$$\frac{\Gamma \to \phi_1 \quad || \quad \Gamma \to \phi_2 \quad || \quad \ldots \quad || \quad \Gamma \to \phi_n}{\mathbf{I}\Gamma \to \mathbf{I}\phi_1, \ldots, \mathbf{I}\phi_n} \quad (INT^{or})$$

$$\frac{\Gamma \to \phi_1 \quad || \quad \Gamma \to \phi_2 \quad || \quad \ldots \quad || \quad \Gamma \to \phi_n}{\mathbf{D}\Gamma \to \mathbf{D}\phi_1, \ldots, \mathbf{D}\phi_n} \quad (DES^{or})$$

- The case $n = 0$ is allowed and then rules transforms into:
  $\frac{\Gamma \to}{\mathbf{I}\Gamma \to} \quad (INT^{or})$ and $\frac{\Gamma \to}{\mathbf{D}\Gamma \to} \quad (DES^{or})$
- Γ - set of the formulas obtained from $\mathbf{I}\Gamma$ or $\mathbf{D}\Gamma$ by removing the most outer $\mathbf{I}$ or $\mathbf{D}$ occurence.

Definition 5.2.6 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak^*)$ and modal rules (BEL^{or}), (INT^{or}), (DES^{or}), $(\circ^{or})$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus BDI_{wf} .

We say, that BDI_{wf} is a weak free sequent calculus for BDI logic. This sequent calculus contains semi-invertable rules: $(Weak^*)$, (INT^{or}), (DES^{or}), $(\circ^{or})$. All the rest rules are invertable.

Theorem 5.2.1 Sequent S is derivable in the sequent calculus BDI_{init} if and only if sequent S is derivable in the sequent calculus BDI_{wf} .

Proof.

If a sequent S is derivable in the BDI_{init} , then we have a derivation tree which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (BEL), (DES), (INT), (AU-L), (AU-R), (EU-L), (EU-R), $(\circ)$, and $(Weak)$. First, we have to eliminate every rule $(Weak)$ application from the inference tree. There are the following cases of the rule $(Weak)$ application:

1. The premise sequent of the rule $(Weak)$ application is an axiom. Then we can delete such a rule $(Weak)$ application at all, since the conclusion sequent of the rule $(Weak)$ application is also an axiom.
2. Rule $(Weak)$ is consecutively applied two times. Then we can transform such a part of the tree to use only one application of the rule $(Weak)$. For this purposes we use the transformation $Trans_1$ as in the proof of the Theorem 3.2.1.

3. Rules $(Weak)$ and (BEL) are consecutively applied (in the bottom-up direction). In this case, we change the rule (BEL) application by the new rules $(Weak^{*})$ and (BEL^{or}) applications. For this purposes we use the transformation $Trans_3$ as in the proof of the Theorem 3.2.1, but we use rule $(Weak^{*})$ instead of the rule $(Weak^{*}_{KD45})$ to get a strict-primal sequent. Note that rule $(\Box)$ (used in the $Trans_3$) is the same rule (BEL).

4. Rules $(Weak)$ and (DES) are consecutively applied (in the bottom-up direction). In this case, we change the rule (DES) application by the new rules $(Weak^{*})$ and (DES^{or}) applications. We denote the main formula of the rule (DES) application by $\mathbf{D}\phi_1$ (may be empty). We denote all other not deleted modalized formulas by $\mathbf{D}\Delta = \mathbf{D}\phi_2, \ldots, \mathbf{D}\phi_k$ and all deleted modalized formulas by $\mathbf{D}\Delta' = \mathbf{D}\phi_{k+1}, \ldots, \mathbf{D}\phi_n$. Then the transformation is:

$$\dfrac{\dfrac{\dfrac{\ldots}{\Gamma \to \phi_1}}{\mathbf{D}\Gamma_2 \to \mathbf{D}\phi_1}\,(DES)}{\Sigma, \mathbf{B}\Gamma_1, \mathbf{D}\Gamma_2, \mathbf{D}\Gamma_2', \mathbf{I}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}\Delta_1, \mathbf{D}\phi_1, \ldots, \mathbf{D}\phi_n, \mathbf{I}\Delta_3, \circ\Delta_4}\,(Weak)$$

$$\Downarrow$$

$$\dfrac{\overset{\ldots}{S_1} \,||\, \dfrac{\dfrac{\dfrac{\ldots}{\Gamma_2 \to \phi_1}}{\Gamma_2, \Gamma_2' \to \phi_1}\,(Weak) \,||\ldots||\, \overset{(redundant)}{\dfrac{\ldots}{\Gamma_2, \Gamma_2' \to \phi_n}}}{\mathbf{D}\Gamma_2, \mathbf{D}\Gamma_2' \to \mathbf{D}\phi_1, \ldots, \mathbf{D}\phi_n}\,(DES^{or}) \,||\, \overset{\ldots}{S_3} \,||\, \overset{\ldots}{S_4}}{\Sigma, \mathbf{B}\Gamma_1, \mathbf{D}\Gamma_2, \mathbf{D}\Gamma_2', \mathbf{I}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}\Delta_1, \mathbf{D}\phi_1, \ldots, \mathbf{D}\phi_n, \mathbf{I}\Delta_3, \circ\Delta_4}\,(Weak^{*})$$

Here: $\dfrac{\overset{(redundant)}{\dfrac{\ldots}{\mathbf{B}\Gamma_1 \to \mathbf{B}\Delta_1}}}{S_1}$, $\dfrac{\overset{(redundant)}{\dfrac{\ldots}{\mathbf{I}\Gamma_3 \to \mathbf{I}\Delta_3}}}{S_3}$, $\dfrac{\overset{(redundant)}{\dfrac{\ldots}{\circ\Gamma_4 \to \circ\Delta_4}}}{S_1}$.

Sequents S_1; S_3; S_4; $\Gamma_2, \Gamma_2' \to \phi_2; \Gamma_2, \Gamma_2' \to \phi_3; \ldots; \Gamma_2, \Gamma_2' \to \phi_n$ are some unused or-branches of the derivation tree, because we get a derivation tree (in the calculus BDI_{init}) by choosing formula $\mathbf{D}\phi_1$ as the main for the rule (DES) application.

5. Rules $(Weak)$ and (INT) are consecutively applied (in the bottom-up direction). We use analogous transformations as in the Case 4.

6. Rule $(Weak)$ and rule R is consecutively applied (in the bottom-up direction) (rule R is one of the $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (AU-L), (AU-R), (EU-L), (EU-R)). We can change the order of these rules applications (the rule $(Weak)$ will appear above the rule R application). For this purposes we use analogous transformations to the $Trans_2$ and the $Trans_5$ as in the proof of the Theorem 3.2.1 and in the proof of the Theorem 4.2.1.

7. Rules $(Weak)$ and $(\circ)$ are consecutively applied (in the bottom-up direction). In this case, we change rule $(\circ)$ application by the new rules $(Weak^{*})$ and $(\circ^{or})$ applications. For this purposes we use transformation $Trans_6$ as in the proof of the Theorem 4.2.1, but we use rule $(Weak^{*})$ instead of the rule $(Weak^{*}_{PTL})$ to get a strict-primal sequent.

By applying such a transformations in the bottom-up direction, we eliminate all applications of the rule $(Weak)$ from the sequent S derivation tree. After these transformations, we get a derivation tree for the sequent calculus BDI_{wf} , therefore, sequent S is also derivable in the calculus BDI_{wf} .

If a sequent S is derivable in the BDI_{wf} , then we have derivation tree, which uses rules $(\neg L)$, $(\neg R)$, $(\vee L)$, $(\vee R)$, $(\& L)$, $(\& R)$, (BEL^{or}), (DES^{or}), (INT^{or}), (AU-L), (AU-R), (EU-L), (EU-R), $(\circ^{or})$, and $(Weak^{*})$ We can apply another transformations:

1. We change rule $(Weak^{*})$ application into the rule $(Weak)$ application.

2. We change rule (BEL^{or}) application into (BEL) rule application as it is done with transformation $Trans_4$ in the proof of the Theorem 3.2.1.

3. We change rule $(\circ^{or})$ application into $(\circ)$ rule application as it is done with transformation $Trans_7$ in the proof of the Theorem 4.2.1.

4. We change rule (DES^{or}) application into (DES) rule application using the following transformation. After rule (DES^{or}) application, we can get n sequents. At least one of them is derivable if the father sequent is derivable. If the initial sequent S is derivable, then we can leave only one branch from or-branches which is derivable in the BDI_{wf} . Suppose that we have chosen such a derivable branch and marked it as derivable. All other or-branches become redundant and, therefore, we mark them as redundant. Suppose that derivable branch is obtained by taking premise with $\mathbf{D}\phi_1$ (for the case $n = 0$, $\mathbf{D}\phi_1$ and ϕ_1 denotes an empty sequents) and apply transformation:

$$\frac{\begin{array}{c}(derivable)\\ \dfrac{\cdots}{\Gamma \to \phi_1}\end{array} \quad || \quad \begin{array}{c}(redundant)\\ \dfrac{\cdots}{\Gamma \to \phi_2}\end{array} \quad ||\ldots|| \quad \begin{array}{c}(redundant)\\ \dfrac{\cdots}{\Gamma \to \phi_n}\end{array}}{\mathbf{D}\Gamma \to \mathbf{D}\phi_1, \mathbf{D}\phi_2, \ldots, \mathbf{D}\phi_n}\ (DES^{or})$$

$$\Downarrow$$

$$\dfrac{\dfrac{\begin{array}{c}(derivable)\\ \dfrac{\cdots}{\Gamma \to \phi_1}\end{array}}{\mathbf{D}\Gamma \to \mathbf{D}\phi_1}\ (DES)}{\mathbf{D}\Gamma \to \mathbf{D}\phi_1, \mathbf{D}\phi_2, \ldots, \mathbf{D}\phi_n}\ (Weak)$$

5. We change rule (INT^{or}) application into (INT) rule application using analogous transformations as in the Case 4.

By applying such a transformations, we eliminate all the applications of the rules $(Weak^*)$, (BEL^{or}) (DES^{or}) (INT^{or}), $(\circ^{or})$ from the derivation tree. After these transformations, we get a derivation tree for the sequent calculus BDI_{init} , therefore, sequent S is also derivable in the BDI_{init} .

Definition 5.2.7 We say that loop $S \rightsquigarrow S'$ is a Belief type loop if at least one (BEL^{or}) (or (BEL^*), (BEL_i^{or}), (BEL_i^*)) rule is applied between sequents S and S'.

Definition 5.2.8 We say that loop $S \rightsquigarrow S'$ is a Until type loop if at least one $(\circ^{or})$ (or $(\circ^+)$) rule is applied between sequents S and S'.

Lemma 5.2.1 If $S \rightsquigarrow S'$ is a loop in a the sequent calculus BDI_{wf} , then there is no rules (DES^{or}), (INT^{or}) applications between sequents S and S'.

Proof.

The precise proof is placed in [39]. Briefly, any rule (DES^{or}) (or (INT^{or})) application decrease the number of the $\mathbf{D}$ (or $\mathbf{I}$) modal operators. There is no rule, which may increase the number of the $\mathbf{D}$ (or $\mathbf{I}$) modal operators. Therefore, we get a contradiction for loop $S \rightsquigarrow S'$ existence.

Lemma 5.2.2 Any loop $S \rightsquigarrow S'$ in the sequent calculus BDI_{wf} is either a Belief type loop or Until type loop (and not both types at the same time).

Proof.

The precise proof is placed in [39]. Briefly, any Belief type loop-starting sequent S contains some $\mathbf{B}$ modalized formula and any rule $(\circ^{or})$ may by applied only after rule $(Weak^*)$ application, which deletes all $\mathbf{B}$ modalized formulas from that premise, and we get a contradiction for loop $S \rightsquigarrow S'$ existence. Any Until type loop-starting sequent S contains some extended Æ formula and any rule (BEL^{or}) may by applied only after rule $(Weak^*)$ application which deletes all extended Æ formulas from that premise, and we get a contradiction for loop $S \rightsquigarrow S'$ existence.

Lemma 5.2.1 and Lemma 5.2.2 says, that if we have a loop it is either a loop for $KD45$ logic modality (Belief type loop), or a loop for branching time logic modality (Until type loop). We know how to eliminate $KD45$ loop-check and how to make an efficient loop-check for branching time logic. According to the Lemma 5.2.2, all our restrictions for the loop-check are valid for the sequent calculus BDI_{wf} as well.

Definition 5.2.9 Sequent calculus rule $(Weak^{*+})$ is:

$$\frac{\mathbf{B}\Gamma_1, \mathbf{B}^*\Gamma_1' \xrightarrow{+} \mathbf{B}\Delta_1, \mathbf{B}^*\Delta_1' \quad || \quad \mathbf{D}\Gamma_2 \xrightarrow{+} \mathbf{D}\Delta_2 \quad || \quad \mathbf{I}\Gamma_3 \xrightarrow{+} \mathbf{I}\Delta_3 \quad || \quad \circ\Gamma_4 \to \circ\Delta_4}{\Sigma, \mathbf{B}\Gamma_1, \mathbf{B}^*\Gamma_1', \mathbf{D}\Gamma_2, \mathbf{I}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}\Delta_1, \mathbf{B}^*\Delta_1', \mathbf{D}\Delta_2, \mathbf{I}\Delta_3, \circ\Delta_4}$$

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

Definition 5.2.10 Sequent calculus rule (BEL^*) is:

$$\frac{\Gamma, \Gamma_1, \mathbf{B}^*\Gamma, \mathbf{B}^*\Gamma_1 \to \phi_1, \mathbf{B}^*\phi_1, \ldots, \Box^*\phi_n || \ldots || \Gamma, \Gamma_1, \mathbf{B}^*\Gamma, \mathbf{B}^*\Gamma_1 \to \phi_n, \mathbf{B}^*\phi_1, \ldots, \mathbf{B}^*\phi_n}{\mathbf{B}\Gamma, \mathbf{B}^*\Gamma_1 \to \mathbf{B}\phi_1, \ldots, \mathbf{B}\phi_k, \mathbf{B}^*\phi_{k+1}, \ldots, \mathbf{B}^*\phi_n}$$

(BEL^*) can be applied only if $\mathbf{B}\Gamma \cup \mathbf{B}\phi_1 \cup \ldots \cup \mathbf{B}\phi_k \neq \emptyset$.

- The case $n = 0$ is allowed, and then rule transforms into:
  $\frac{\Gamma, \Gamma_1, \mathbf{B}^*\Gamma, \mathbf{B}^*\Gamma_1 \to}{\mathbf{B}\Gamma, \mathbf{B}^*\Gamma_1 \to} \quad (BEL^*)$.
- $\mathbf{B}^*\Gamma$ - set of the formulas obtained from $\mathbf{B}\Gamma$ by replacing every most outer $\mathbf{B}$ occurence with $\mathbf{B}^*$.

This is the same rule used for $KD45$ logic (see Definition 3.2.7). In this chapter, we deal with several operators, and, therefore, we use operator $\mathbf{B}$ instead of $\Box$ and rule (BEL^*) instead of the rule $(\Box^*)$.

Definition 5.2.11 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak^{*+})$ and modal rules (BEL^*), (DES^{or}), (INT^{or}), $(\circ^+)$, (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) we define sequent calculus BDI_{elc} .

We say, that BDI_{elc} is a sequent calculus with an efficient loop-check for BDI logic. This sequent calculus contains semi-invertable rules: $(Weak^{*+})$, (DES^{or}), (INT^{or}), $(\circ^+)$. All the rest rules are invertable.

All premises of the rule $(Weak^{*+})$, except one $\circ\Gamma_4 \to \circ\Delta_4$, are marked by $+$, since these premises do not contain any extended Æ formula and every Until type loop contains at least one extended Æ formula (which is ground).

Theorem 5.2.2 Sequent S is derivable in the sequent calculus BDI_{wf} if and only if sequent S is derivable in the sequent calculus BDI_{elc} .

Proof.

According to the Lemma 5.2.2, every lemma, for loops for $KD45$ logic, may be applied for Belief type loops. Therefore, we can use rule (BEL^*) instead of the rule (BEL^{or}) without loosing derivability (see the proof of the Theorem 3.2.2).

According to the Lemma 5.2.2, every lemma, for loops for branching time logic, may be applied for Until type loops. Therefore, we can use rules $(\circ^+)$, (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) instead of the rules $(\circ^{or})$, (AU-L), (AU-R), (EU-L), (EU-R) without loosing derivability (see proof of the Theorem 4.2.3 and the Corollary 4.2.1).

Theorem 5.2.3 Sequent calculus BDI_{elc} is sound and complete calculus for BDI logic.

Proof.

The proof goes straightforward from the Theorems 5.1.1, 5.2.1, 5.2.2.

Inference tree construction in the sequent calculus BDI_{elc} always terminates, because every rule application contains only the finite premises count, and every sequent in the inference tree contains only subformulas of the initial sequent, and derivation is not proceed for every loop-ending sequent.

We constructed sequent calculus for BDI logic, which contains only invertable, or semi-invertable rules. This sequent calculus deals only with Until type loops, since Belief type loops were eliminated. We use restricted loop-check for detecting Until type loops. According to provided restrictions:

- Loop-check is used only for Until type loops.
- Loop-check is used not for every sequent in the inference tree, but only for special marked sequents (marked by $\circ$).
- During loop-check, only special marked sequents (marked by $\circ$) are tested.
- Only the sequents between the current sequent and the first sequent marked by $+$ (in the top-down direction) are tested.

5.3 Multiagent BDI Logic

In this section, multiagent BDI logic is discussed. Hilbert style axiomatization, sound and complete sequent calculus for multiagent BDI logic are presented.

If environment contains not a single agent or we need several agents, then monoagent BDI logic is not applicable. We need to use multiagent case of the BDI logic. In the multiagent BDI logic we may have several agents and every agents has its own beliefs, desires and intentions. Therefore, every agent needs separate $\mathbf{B}$, $\mathbf{D}$ and $\mathbf{I}$ modal operators in calculus for BDI logic. We still need only one $\circ$ operator and until operators ($A(\phi \cup \psi), E(\phi \cup \psi)$), since all agents acts in the same environment.

Multiagent BDI logic uses (n is a finite number):

- n modal operators for Belief modality ($\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_n$),
- n modal operators for Desire modality ($\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_n$),
- n modal operators for Intend modality ($\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_n$).

The full Hilbert style system for multiagent BDI logic is the following ([52]).

Definition 5.3.1 Hilbert type calculus for multiagent BDI logic is calculus with classical non modal axioms and modal axioms as follows:

- (B-K) $\mathbf{B}_i(\phi \to \psi) \to (\mathbf{B}_i\phi \to \mathbf{B}_i\psi)$,
- (B-D) $\mathbf{B}_i\phi \to \neg\mathbf{B}_i\neg\phi$,
- (B-4) $\mathbf{B}_i\phi \to \mathbf{B}_i\mathbf{B}_i\phi$,
- (B-5) $\neg\mathbf{B}_i\phi \to \mathbf{B}_i\neg\mathbf{B}_i\phi$,
- (D-K) $\mathbf{D}_i(\phi \to \psi) \to (\mathbf{D}_i\phi \to \mathbf{D}_i\psi)$,
- (D-D) $\mathbf{D}_i\phi \to \neg\mathbf{D}_i\neg\phi$,
- (I-K) $\mathbf{I}_i(\phi \to \psi) \to (\mathbf{I}_i\phi \to \mathbf{I}_i\psi)$,
- (I-D) $\mathbf{I}_i\phi \to \neg\mathbf{I}_i\neg\phi$,
- $EXtrue\&AXtrue$,
- $AG(\xi \to (\neg\psi\&EX\xi)) \to (\xi \to \neg A(\phi \cup \psi)$,
- $AG(\xi \to (\neg\psi\&EX\xi)) \to (\xi \to \neg AF\psi)$,
- $AG(\xi \to (\neg\psi\&(\phi \to AX\xi))) \to (\xi \to \neg E(\phi \cup \psi))$,
- $AG(\xi \to (\neg\psi\&AX\xi)) \to (\xi \to \neg EF\psi)$,
- $AG(\phi \to \psi) \to (EX\phi \to EX\psi)$,

and rules:

$$\frac{\phi, \quad \phi \to \psi}{\psi}, \qquad \frac{\phi}{\mathbf{B}_i\,\phi}, \qquad \frac{\phi}{\mathbf{D}_i\,\phi}, \qquad \frac{\phi}{\mathbf{I}_i\,\phi}, \qquad \frac{\phi}{AG\,\phi}.$$

Here

- $EF\phi \equiv E(true \cup \phi)$,
- $AG\phi \equiv \neg EF\neg\phi$,
- $AF\phi \equiv A(true \cup \phi)$,
- $EG\phi \equiv \neg AF\neg\phi$,
- $EX(\phi \vee \psi) \equiv EX\phi \vee EX\psi$,

- $AX\phi \equiv \neg EX\neg\phi$,
- $E(\phi \cup \psi) \equiv \psi \vee (\psi \& EXE(\phi \cup \psi))$,
- $A(\phi \cup \psi) \equiv \psi \vee (\psi \& AXA(\phi \cup \psi))$,
- modal operator AX stands for $\circ$ operator.

Hilbert style axiomatization for multiagent BDI logic is almost the same as for monoagent BDI logic (only index of an agent is added). Therefore, sequent calculus or multiagent BDI logic follows from the sequent calculus for BDI logic (presenetd in [39]).

Definition 5.3.2 Sequent calculus rule (BEL_i) is:

$$\frac{\Gamma, \mathbf{B}_i\Gamma \to \Theta, \mathbf{B}_i\Theta, \mathbf{B}_i\Delta}{\mathbf{B}_i\Gamma \to \mathbf{B}_i\Theta, \mathbf{B}_i\Delta} \ (BEL_i) \qquad (i = 1, 2, \ldots n)$$

- Θ is empty or only one formula.
- Γ - set of the formulas obtained from $\mathbf{B}_i\Gamma$ by removing the most outer $\mathbf{B}_i$ occurence.

Definition 5.3.3 Sequent calculus rules (INT_i) and (DES_i) are:

$$\frac{\Gamma \to \Theta}{\mathbf{I}_i\Gamma \to \mathbf{I}_i\Theta} \ (INT_i) \qquad \frac{\Gamma \to \Theta}{\mathbf{D}_i\Gamma \to \mathbf{D}_i\Theta} \ (DES_i) \qquad (i = 1, 2, \ldots n)$$

- Θ is empty or only one formula.
- Γ - set of the formulas obtained from $\mathbf{I}_i\Gamma$ or $\mathbf{D}_i\Gamma$ by removing the most outer $\mathbf{I}_i$ or $\mathbf{D}_i$ occurence respectively.

Definition 5.3.4 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak)$ and modal rules (BEL_i), (INT_i), (DES_i), $(\circ)$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus BDI^n_{init} .

We say, that BDI^n_{init} is an initial sequent calculus for multiagent BDI logic. This sequent calculus contains non invertable (and non semi-invertable) rule $(Weak)$ and semi-invertable rules (INT_i), (DES_i), $(\circ)$. Therefore, at least some derivation tactics is necessary to get decidability.

Theorem 5.3.1 Sequent calculus BDI^n_{init} is sound and complete calculus for multiagent BDI logic.

Proof.

The proof for monoagent BDI logic is presented by N. NIDE and T. Shiro in [39]. Authors claim that it is the same for multiagent case (it follows from the Hilbert style axiomatizations for monoagent and multiagent BDI logics, those are very similar).

5.4 Sequent Calculus With an Efficient Loop-check for Multiagent BDI Logic

In this section, sequent calculus for multiagent BDI logic is introduced. Calculus uses efficient loop-check to get decidability. Sequent calculus for multiagent BDI logic is constructed in the way analogous to the one used for the sequent calculus for monoagent BDI logic and is based on the loop-check free sequent calculus for $KD45$ logic and sequent calculus with an efficient loop-check for branching time logic presented in the previous chapters.

Sequent calculus BDI^n_{wf} (and BDI^n_{init}) do not use marked modal operator $\mathbf{B}^*_i$, but such a marked operator is used in the sequent calculus BDI^n_{elc} (introduced later). Therefore, we also use marked modal operator $\mathbf{B}^*_i$ in the following definitions, since they are applicable for the both calculi.

Definition 5.4.1 We say that sequent
$\Sigma, \mathbf{B}_{1 \ldots n}\Gamma_1, \mathbf{D}_{1 \ldots n}\Gamma_2, \mathbf{I}_{1 \ldots n}\Gamma_3, \circ\Gamma_4 \to \Pi, \mathbf{B}_{1 \ldots n}\Delta_1, \mathbf{D}_{1 \ldots n}\Delta_2, \mathbf{I}_{1 \ldots n}\Delta_3, \circ\Delta_4$
is primal sequent for multiagent BDI logic if:

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.
- $\mathbf{B}_{1 \ldots n}\Gamma_1 = \mathbf{B}_1\Gamma^1_1, \mathbf{B}^*_1\Gamma'^1_1, \ldots, \mathbf{B}_n\Gamma^n_1, \mathbf{B}^*_n\Gamma'^n_1$;
  $\mathbf{B}_{1 \ldots n}\Delta_1 = \mathbf{B}_1\Delta^1_1, \mathbf{B}^*_1\Delta'^1_1, \ldots, \mathbf{B}_n\Delta^n_1, \mathbf{B}^*_n\Delta'^n_1$.
- $\mathbf{D}_{1 \ldots n}\Gamma_2 = \mathbf{D}_1\Gamma^1_2, \ldots, \mathbf{D}_n\Gamma^n_2$; $\quad \mathbf{D}_{1 \ldots n}\Delta_2 = \mathbf{D}_1\Delta^1_2, \ldots, \mathbf{D}_n\Delta^n_2$.
- $\mathbf{I}_{1 \ldots n}\Gamma_3 = \mathbf{I}_1\Gamma^1_3, \ldots, \mathbf{I}_n\Gamma^n_3$; $\quad \mathbf{I}_{1 \ldots n}\Delta_3 = \mathbf{I}_1\Delta^1_3, \ldots, \mathbf{I}_n\Delta^n_3$.

Definition 5.4.2 We say that sequents:

- $\mathbf{B}_i\Gamma, \mathbf{B}^*_i\Gamma' \to \mathbf{B}_i\Delta, \mathbf{B}^*_i\Delta'$,
- $\mathbf{D}_i\Gamma \to \mathbf{D}_i\Delta$,
- $\mathbf{I}_i\Gamma \to \mathbf{I}_i\Delta$,
- $\circ\Gamma \to \circ\Delta$,

are strict-primal sequents for multiagent BDI logic.

Definition 5.4.3 Sequent calculus rule $(Weak^*_i)$ is:

$$\frac{S^1_1 || \ldots || S^n_1 \quad || \quad S^1_2 || \ldots || S^n_2 \quad || \quad S^1_3 || \ldots || S^n_3 \quad || \quad \circ\,\Gamma_4 \to \circ\Delta_4}{\Sigma, \mathbf{B}_{1 \ldots n}\Gamma_1, \mathbf{D}_{1 \ldots n}\Gamma_2, \mathbf{I}_{1 \ldots n}\Gamma_3, \circ\Gamma^4 \to \Pi, \mathbf{B}_{1 \ldots n}\Delta_1, \mathbf{D}_{1 \ldots n}\Delta_2, \mathbf{I}_{1 \ldots n}\Delta_3, \circ\Delta_4} \; (Weak^*_i)$$

- $S_1^i = \mathbf{B}_i\Gamma_1^i \to \mathbf{B}_i\Delta_1^i$; $\quad S_2^i = \mathbf{D}_i\Gamma_2^i \to \mathbf{D}_i\Delta_2^i$; $\quad S_3^i = \mathbf{I}_i\Gamma_3^i \to \mathbf{I}_i\Delta_3^i$,
  for every $i = 1, 2, \dots n$.
- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.
- $\mathbf{B}_{1\dots n}\Gamma_1 = \mathbf{B}_1\Gamma_1^1, \dots, \mathbf{B}_n\Gamma_1^n$; $\qquad \mathbf{B}_{1\dots n}\Delta_1 = \mathbf{B}_1\Delta_1^1, \dots, \mathbf{B}_n\Delta_1^n$.
- $\mathbf{D}_{1\dots n}\Gamma_2 = \mathbf{D}_1\Gamma_2^1, \dots, \mathbf{D}_n\Gamma_2^n$; $\qquad \mathbf{D}_{1\dots n}\Delta_2 = \mathbf{D}_1\Delta_2^1, \dots, \mathbf{D}_n\Delta_2^n$.
- $\mathbf{I}_{1\dots n}\Gamma_3 = \mathbf{I}_1\Gamma_3^1, \dots, \mathbf{I}_n\Gamma_3^n$; $\qquad \mathbf{I}_{1\dots n}\Delta_3 = \mathbf{I}_1\Delta_3^1, \dots, \mathbf{I}_n\Delta_3^n$.

Definition 5.4.4 Sequent calculus rule (BEL_i^{or}) is:

$$\frac{\Gamma, \mathbf{B}_i\Gamma \to \phi_1, \mathbf{B}_i\phi_1, \dots, \mathbf{B}_i\phi_n \quad || \dots || \quad \Gamma, \mathbf{B}_i\Gamma \to \phi_n, \mathbf{B}_i\phi_1, \dots, \mathbf{B}_i\phi_n}{\mathbf{B}_i\Gamma \to \mathbf{B}_i\phi_1, \dots, \mathbf{B}_i\phi_n} \quad (BEL_i^{or})$$

$(i = 1, 2, \dots, n)$

- The case $n = 0$ is allowed and then rule transforms into:
  $\frac{\Gamma, \mathbf{B}_i\Gamma \to}{\mathbf{B}_i\Gamma \to} \quad (BEL_i^{or})$
- Γ - set of the formulas obtained from $\mathbf{B}_i\Gamma$ by removing the most outer $\mathbf{B}_i$ occurence.

This is the same rule (BEL^{or}) used for monoagent BDI logic, which is adapted for the multiagent case.

Definition 5.4.5 Sequent calculus rules (INT_i^{or}) and (DES_i^{or}) are:

$$\frac{\Gamma \to \phi_1 \quad || \quad \Gamma \to \phi_2 \quad || \quad \dots \quad || \quad \Gamma \to \phi_n}{\mathbf{I}_i\Gamma \to \mathbf{I}_i\phi_1, \dots, \mathbf{I}_i\phi_n} \quad (INT_i^{or}) \qquad (i = 1, 2, \dots, n)$$

$$\frac{\Gamma \to \phi_1 \quad || \quad \Gamma \to \phi_2 \quad || \quad \dots \quad || \quad \Gamma \to \phi_n}{\mathbf{D}_i\Gamma \to \mathbf{D}_i\phi_1, \dots, \mathbf{D}_i\phi_n} \quad (DES_i^{or}) \qquad (i = 1, 2, \dots, n)$$

- The case $n = 0$ is allowed and then rules transforms into:
  $\frac{\Gamma \to}{\mathbf{I}_i\Gamma \to} \quad (INT_i^{or})$ and $\frac{\Gamma \to}{\mathbf{D}_i\Gamma \to} \quad (DES_i^{or})$
- Γ - set of the formulas obtained from $\mathbf{I}_i\Gamma$ or $\mathbf{D}_i\Gamma$ by removing the most outer $\mathbf{I}_i$ or $\mathbf{D}_i$ occurence.

This is the same rules (DES^{or}) and (INT^{or}) used for monoagent BDI logic adapted for multiagent case.

Definition 5.4.6 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak_i^*)$ and modal rules (BEL_i^{or}), (INT_i^{or}), (DES_i^{or}), $(\circ^{or})$, (AU-L), (AU-R), (EU-L), (EU-R) we define sequent calculus BDI_{wf}^n .

We say, that BDI^n_{wf} is a weak free sequent calculus for multiagent BDI logic. This sequent calculus contains semi-invertable rules: $(Weak^*_i)$, (INT^{or}_i), (DES^{or}_i), $(\circ^{or})$. All the rest rules are invertable.

Theorem 5.4.1 Sequent S is derivable in the sequent calculus BDI^n_{init} if and only if sequent S is derivable in the sequent calculus BDI^n_{wf} .

Proof.

The proof is the same as for monoagent case (see the Theorem 5.2.1). In this case, we have to change rule $(Weak^*)$ application with the rule $(Weak^*_i$ application, rule (BEL) application with the rule (BEL_i) application, rule (DES) application with the rule (DES_i) application, rule (INT) application with the rule (INT_i) application, rule (BEL^{or}) application with the rule (BEL^{or}_i) application, rule (DES^{or}) application with the rule (DES^{or}_i) application, rule (INT^{or}) application with the rule (INT^{or}_i) application. Rule $(Weak^*_i)$ application have more redundant or-branches then rule $(Weak^*)$ application, but it do not impact the given proof.

Lemma 5.4.1 If $S \rightsquigarrow S'$ is a loop in the sequent calculus BDI^n_{wf} , then there is no rules (DES^{or}_i), (INT^{or}_i) applications between sequents S and S'.

Proof.

Rule (DES^{or}_i) (or (INT^{or}_i)) application decrease the number of the $\mathbf{D}_i$ (or $\mathbf{I}_i$) modal operators.

Suppose that we have a loop $S \rightsquigarrow S'$. Suppose that sequents S_1, S'_2 are inside a loop and S_1 is a conclusion and S_2 is a premise of the rule (DES^{or}_i) (or (INT^{or}_i)) application. Then S_1 contains only formulas of the shape $\mathbf{D}_i\phi$ (or $\mathbf{I}_i\phi$). Suppose, that $\mathbf{D}_i\psi$ (or $\mathbf{I}_i\psi$) is such of them, that its length is greater or equal then any other formula in the sequent S_1. Therefore, formula $\mathbf{D}_i\psi$ (or $\mathbf{I}_i\psi$) is a subformula in the sequent S_1, but it is not a subformula in the sequent S_2. Since any rule premise contains only subformulas of the conclusion, formula $\mathbf{D}_i\psi$ (or $\mathbf{I}_i\psi$) is a subformula in the sequent S, but it is not a subformula in the sequent S'. We get a contradiction, because $S \rightsquigarrow S'$ is a loop and S' contains all formulas from S.

Lemma 5.4.2 Any loop $S \rightsquigarrow S'$ in the sequent calculus BDI^n_{wf} is either a Belief type loop or Until type loop (and not both types at the same time).

Moreover, any Belief type loop $S \rightsquigarrow S'$ in the sequent calculus BDI^n_{wf} is either a loop for one agent's belief modality or a loop for another agent's belief modality (and not a loop for the both agents at the same time).

Proof.

Suppose that we have an Until type loop $S \rightsquigarrow S'$. Suppose that sequents S_1, S_2' are inside a loop and S_1 is a conclusion and S_2 is a premise of the rule (BEL_i^{or}) application. Since every sequent inside Until type loop contains at least one extended Æ formula (not subformula), and S_1 does not contain extended Æ formulas (because it is a conclusion of the rule (BEL_i^{or})), we get a contradiction. Therefore, no rule (BEL_i^{or}) application exists inside Until type loop.

Suppose that we have a Belief type loop $S \rightsquigarrow S'$. Suppose that sequents S_1, S_2' are inside a loop and S_1 is a conclusion and S_2 is a premise of the rule $(\circ^{or})$ application. Since every sequent inside Belief type loop $S \rightsquigarrow S'$ contains at least one $\mathbf{B}_i$ modalized formula, and S_1 do not contain $\mathbf{B}_i$ modalized formula (because it is a conclusion of the rule $(\circ^{or})$), we get a contradiction. Therefore, no rule $(\circ^{or})$ application exists inside Belief type loop.

Suppose that we have a Belief type loop $S \rightsquigarrow S'$ and sequent S contains formula having the shape $\mathbf{B}_w\phi$, and $w \neq i$ ($S \rightsquigarrow S'$ is also a Belief type loop for agent w). Suppose that sequents S_1, S_2 are inside a loop and S_1 is a conclusion and S_2 is a premise of the rule (BEL_i^{or}) application (and $i \neq w$). Since every sequent inside Belief type loop $S \rightsquigarrow S'$ contains at least one formula having the shape $\mathbf{B}_w\phi$, and S_1 do not contain formula having the shape $\mathbf{B}_w\phi$ (because it is a conclusion of the rule (BEL_i^{or}) and $i \neq w$), we get a contradiction. Therefore, any Belief type loop $S \rightsquigarrow S'$ in the sequent calculus BDI_{wf}^n is either a loop for one agent's belief modality or a loop for another agent's belief modality.

Lemma 5.4.1 and Lemma 5.4.2 says, that if we have a loop it is either a loop for $KD45$ logic modality (Belief type loop for other particular agent), or a loop for branching time logic modality (Until type loop). We now how to eliminate $KD45$ loop-check and how to make an efficient loop-check for branching time logic. According to the Lemma 5.4.2, all our restrictions for the loop-check are valid for the sequent calculus BDI_{wf}^n as well.

Definition 5.4.7 Sequent calculus rule $(Weak_i^{*+})$ is:

$$\frac{S_1^1||\ldots||S_1^n \quad || \quad S_2^1||\ldots||S_2^n \quad || \quad S_3^1||\ldots||S_3^n \quad || \quad \circ\Gamma_4 \to \circ\Delta_4}{\Sigma, \mathbf{B}_{1\ldots n}\Gamma_1, \mathbf{D}_{1\ldots n}\Gamma_2, \mathbf{I}_{1\ldots n}\Gamma_3, \circ\Gamma^4 \to \Pi, \mathbf{B}_{1\ldots n}\Delta_1, \mathbf{D}_{1\ldots n}\Delta_2, \mathbf{I}_{1\ldots n}\Delta_3, \circ\Delta_4}$$

- $S_1^i = \mathbf{B}_i\Gamma_1^i, \mathbf{B}_i^*\Gamma_1'^i \xrightarrow{+} \mathbf{B}_i\Delta_1^i, \mathbf{B}_i^*\Delta_1'^i$;
  $S_2^i = \mathbf{D}_i\Gamma_2^i \xrightarrow{+} \mathbf{D}_i\Delta_2^i$;
  $S_3^i = \mathbf{I}_i\Gamma_3^i \xrightarrow{+} \mathbf{I}_i\Delta_3^i$, for every $i = 1, 2, \ldots n$.

- Σ, Π - finite (may be empty) sets of propositional variables, $\Sigma \cap \Pi = \emptyset$.

- $\mathbf{B}_{1\ldots n}\Gamma_1 = \mathbf{B}_1\Gamma_1^1, \mathbf{B}_1^*\Gamma_1'^1, \ldots, \mathbf{B}_n\Gamma_1^n, \mathbf{B}_n^*\Gamma_1'^n$;

  $\mathbf{B}_{1\ldots n}\Delta_1 = \mathbf{B}_1\Delta_1^1, \mathbf{B}_1^*\Delta_1'^1, \ldots, \mathbf{B}_n\Delta_1^n, \mathbf{B}_n^*\Delta_1'^n$.

- $\mathbf{D}_{1\ldots n}\Gamma_2 = \mathbf{D}_1\Gamma_2^1, \ldots, \mathbf{D}_n\Gamma_2^n$; $\mathbf{D}_{1\ldots n}\Delta_2 = \mathbf{D}_1\Delta_2^1, \ldots, \mathbf{D}_n\Delta_2^n$.
- $\mathbf{I}_{1\ldots n}\Gamma_3 = \mathbf{I}_1\Gamma_3^1, \ldots, \mathbf{I}_n\Gamma_3^n$; $\mathbf{I}_{1\ldots n}\Delta_3 = \mathbf{I}_1\Delta_3^1, \ldots, \mathbf{I}_n\Delta_3^n$.

Definition 5.4.8 Sequent calculus rule (BEL_i^*) is:

$$\frac{\Gamma, \Gamma_1, \mathbf{B}_i^*\Gamma, \mathbf{B}_i^*\Gamma_1 \to \phi_1, \mathbf{B}_i^*\phi_1, \ldots, \mathbf{B}_i^*\phi_n || . . || \Gamma, \Gamma_1, \mathbf{B}_i^*\Gamma, \mathbf{B}_i^*\Gamma_1 \to \phi_n, \mathbf{B}_i^*\phi_1, \ldots, \mathbf{B}_i^*\phi_n}{\mathbf{B}_i\Gamma, \mathbf{B}_i^*\Gamma_1 \to \mathbf{B}_i\phi_1, \ldots, \mathbf{B}_i\phi_k, \mathbf{B}_i^*\phi_{k+1}, \ldots, \mathbf{B}_i^*\phi_n}$$

$(i = 1, 2, \ldots n)$
(BEL_i^*) can be applied only if $\mathbf{B}_i\Gamma \cup \mathbf{B}_i\phi_1 \cup \ldots \cup \mathbf{B}_i\phi_k \neq \emptyset$.

- The case $n = 0$ is allowed, and then rule transforms into:
  $\dfrac{\Gamma, \Gamma_1, \mathbf{B}_i^*\Gamma, \mathbf{B}_i^*\Gamma_1 \to}{\mathbf{B}_i\Gamma, \mathbf{B}_i^*\Gamma_1 \to}$ (BEL_i^*).
- $\mathbf{B}_i^*\Gamma$ - set of the formulas obtained from $\mathbf{B}_i\Gamma$ by replacing every most outer $\mathbf{B}_i$ occurence with $\mathbf{B}_i^*$.

This is the same rule (BEL^*) used for monoagent BDI logic adapted for multiagent case.

Definition 5.4.9 The sequent calculus with a loop-axiom and with an axiom $\phi, \Gamma \to \phi, \Delta$, logical rules, rule $(Weak_i^{*+})$ and modal rules (BEL_i^*), (INT_i^{or}), (DES_i^{or}), ($\circ^+$), (AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$) we define sequent calculus BDI_{elc}^n .

We say, that BDI_{elc}^n is a sequent calculus with an efficient loop-check for multiagent BDI logic. This sequent calculus contains semi-invertable rules: $(Weak_i^{*+})$, (INT_i^{or}), (DES_i^{or}), ($\circ^+$). All the rest rules are invertable.

All premises of the rule $(Weak_i^{*+})$, except one $\circ\Gamma_4 \to \circ\Delta_4$, are marked by $+$, since these premises do not contain any extended Æ formula and every Until type loop contains at least one extended Æ formula (which is ground).

Theorem 5.4.2 Sequent S is derivable in the sequent calculus BDI_{elc}^n if and only if sequent S is derivable in the sequent calculus BDI_{wf}^n .

Proof.
The proof goes straightforward from the Lemma 5.4.2 and the proof of the Theorem 5.2.2 for monoagent BDI logic.

Theorem 5.4.3 Sequent calculus BDI_{elc}^n is sound and complete calculus for multiagent BDI logic.

Proof.

The proof goes straightforward from the Theorems 5.3.1, 5.4.1, 5.4.2.

Inference tree construction in the sequent calculus BDI^n_{elc} always terminates, because every rule application contains only the finite premises count, and every sequent in the inference tree contains only subformulas of the initial sequent, and derivation is not proceed for every loop-ending sequent.

We constructed sequent calculus for multiagent BDI logic, which contains only invertable, or semi-invertable rules. This sequent calculus deals only with Until type loops, since all Belief type loops were eliminated. We use restricted loop-check for detecting Until type loops. According to the restrictions, only special marked (by $+$) sequents are used for the loop-check (loop-check is performed only for such a sequents, and only such a sequents are tested). Moreover, these restrictions allows us to test only the sequents, those are ancestors of the current sequent, but are not the ancestors of any special marked (by $+$) sequent.

Example 5.4.1 Suppose we have a sequent S:
$\mathbf{B}_1 A(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{I}_1 \circ A(\mathbf{B}_2 A(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi) \to A(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \circ\mathbf{B}_2\psi$.

Sequent S with indexed formulas is:
$\mathbf{B}_1 A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{I}_1 \circ A^2_{\emptyset}(\mathbf{B}_2 A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi) \to A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \circ\mathbf{B}_2\psi$.

The following short formula notation is used to denote formulas:

- $G_1' = A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi)$, $G_1 = A^1_{\emptyset}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi)$,
- $G_2 = A^2_{\emptyset}(\mathbf{B}_2 G_1' \cup \mathbf{I}_1\psi) = A^2_{\emptyset}(\mathbf{B}_2 A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi)$.

Sequent $S = \mathbf{B}_1 G_1', \mathbf{I}_1 \circ G_2 \to G_1', \circ\mathbf{B}_2\psi$ inference tree may be the following:

$$\dfrac{\dfrac{\dfrac{S_2}{\mathbf{B}_1 G_1' \xrightarrow{+}} \quad || \quad \dfrac{\dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\uparrow}\,(BEL_2^*)}{\xrightarrow{+} \mathbf{B}_2\psi} \quad || \quad \dfrac{S_3}{\mathbf{I}_1 \circ G_2 \xrightarrow{+}} \quad || \quad \dfrac{S_5}{\to \circ G_1', \circ\mathbf{B}_2\psi}}{\mathbf{B}_1 G_1', \mathbf{I}_1 \circ G_2 \to \mathbf{B}_2\psi, \circ G_1', \circ\mathbf{B}_2\psi}\,(Weak_i^{*+})}{}$$

$$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{S_1}{\mathbf{B}_1 G_1' \xrightarrow{+} \mathbf{B}_1\phi} \quad || \quad \dfrac{\dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\uparrow}\,(BEL_2^*)}{\xrightarrow{+} \mathbf{B}_2\psi} \quad || \quad \dfrac{S_3}{\mathbf{I}_1 \circ G_2 \xrightarrow{+}} \quad || \dfrac{\dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\xrightarrow{\circ} \mathbf{B}_2\psi}\,(BEL_2^*)}{\to \circ\mathbf{B}_2\psi}\,(\circ^+)}{\mathbf{B}_1 G_1', \mathbf{I}_1 \circ G_2 \to \mathbf{B}_1\phi, \mathbf{B}_2\psi, \circ\mathbf{B}_2\psi}\,(Weak_i^{*+})}{\mathbf{B}_1 G_1', \mathbf{I}_1 \circ G_2 \to A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \circ\mathbf{B}_2\psi}\,\text{(AU-R}^+\text{)}}{\mathbf{B}_1 G_1', \mathbf{I}_1 \circ G_2 \to G_1', \circ\mathbf{B}_2\psi}$$

Inference tree fragment for the sequent S_1 is

$$
\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\mathbf{B}_1^*G_1 \xrightarrow{+} \mathbf{B}_1^*\phi} \quad || \quad \dfrac{\overset{\text{(final)}}{\psi, \mathbf{B}_2^*\psi \to}}{\mathbf{B}_2\psi \xrightarrow{+}}\,(BEL_2^*)}{\mathbf{B}_2\psi, \mathbf{B}_1^*G_1 \xrightarrow{+} \phi, \mathbf{B}_1^*\phi}\,(Weak_i^{*+})}{\begin{matrix}\uparrow\\ \uparrow\end{matrix}} \qquad \overset{\oplus}{\mathbf{B}_1\phi, \circ A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_1^*G_1' \to \phi, \mathbf{B}_1^*\phi}}{\dfrac{A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_1^*G_1' \to \phi, \mathbf{B}_1^*\phi}{\dfrac{\dfrac{G_1', \mathbf{B}_1^*G_1' \to \phi, \mathbf{B}_1^*\phi}{\mathbf{B}_1G_1' \xrightarrow{+} \mathbf{B}_1\phi}\,(BEL_1^*)}{S_1}}}\,(\text{AU-L}^+)
$$

Inference tree fragment for the sequent S_2 is

$$
\dfrac{\dfrac{\overset{\text{(final)}}{\mathbf{B}_1^*G_1 \xrightarrow{+}} \quad || \quad \dfrac{\overset{\text{(final)}}{\psi, \mathbf{B}_2^*\psi \to}}{\mathbf{B}_2\psi \xrightarrow{+}}\,(BEL_2^*)}{\mathbf{B}_2\psi, \mathbf{B}_1^*G_1 \xrightarrow{+}}\,(Weak_i^{*+}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\mathbf{B}_1^*\phi, \mathbf{B}_1^*G_1' \xrightarrow{+}} \quad || \quad \dfrac{\overset{\text{(final)}}{\psi, \mathbf{B}_2^*\psi \to}}{\mathbf{B}_2\psi \xrightarrow{+}}\,(BEL_2^*)}{\phi, \mathbf{B}_2\psi, \mathbf{B}_1^*\phi, \mathbf{B}_1^*G_1' \to}\,(Weak_i^{*+}) \qquad \dfrac{\overset{\text{(final)}}{\mathbf{B}_1^*\phi, \mathbf{B}_1^*G_1' \xrightarrow{+}} \quad || \quad \dfrac{S_4}{\circ G_1' \to}}{\phi, \mathbf{B}_1^*\phi, \circ G_1', \mathbf{B}_1^*G_1' \to}\,(Weak_i^{*+})}{\phi, A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_1^*\phi, \mathbf{B}_1^*G_1' \to}\,(\text{AU-L}^+)}{\dfrac{\phi, G_1', \mathbf{B}_1^*\phi, \mathbf{B}_1^*G_1' \to}{\mathbf{B}_1\phi, \mathbf{B}_1^*G_1' \xrightarrow{+}}\,(BEL_1^*)} \qquad || \qquad \dfrac{S_4}{\circ G_1' \to}}{\begin{matrix}\uparrow\\ \uparrow\\ \uparrow\\ \uparrow\end{matrix}}}{\mathbf{B}_1\phi, \circ G_1', \mathbf{B}_1^*G_1' \to}\,(Weak_i^{*+})}{}}{\dfrac{A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_1^*G_1' \to}{\dfrac{\dfrac{G_1', \mathbf{B}_1^*G_1' \to}{\mathbf{B}_1G_1' \xrightarrow{+}}\,(BEL_1^*)}{S_2}}}\,(\text{AU-L}^+)
$$

Inference tree fragment for the sequent S_3 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\psi \to}}{\mathbf{I}_1\psi \xrightarrow{+}}\,(INT_1^{or}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\psi, \mathbf{B}_2^*\psi, \mathbf{B}_2^*G_1 \xrightarrow{+}} \qquad \dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\phi, \mathbf{B}_1^*\phi \to}}{\uparrow}\,(BEL_1^*)}{\mathbf{B}_1\phi \xrightarrow{+}} \quad || \quad \overset{\text{(final)}}{\mathbf{B}_2^*\psi, \mathbf{B}_2^*G_1' \xrightarrow{+}} \quad || \quad \dfrac{S_4}{\circ G_1' \to}}{\psi, \mathbf{B}_1\phi, \circ G_1', \mathbf{B}_2^*\psi, \mathbf{B}_2^*G_1' \to}\,(Weak_i^{*+})}{\psi, A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_2^*\psi, \mathbf{B}_2^*G_1' \to}\,(\text{AU-L}^+)}{\psi, G_1', \mathbf{B}_2^*\psi, \mathbf{B}_2^*G_1' \to}}{\uparrow}\,(BEL_2^*) \\ \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \\ \uparrow\end{matrix}}{\mathbf{B}_2\psi, \mathbf{B}_2^*G_1' \xrightarrow{+}} \qquad \dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\phi, \mathbf{B}_1^*\phi \to}}{\uparrow}\,(BEL_1^*)}{\mathbf{B}_1\phi \xrightarrow{+}} \quad || \quad \overset{\text{(final)}}{\mathbf{B}_2^*G_1' \xrightarrow{+}} \quad || \quad \dfrac{S_4}{\circ G_1' \to}}{\mathbf{B}_1\phi, \circ G_1', \mathbf{B}_2^*G_1' \to}\,(Weak_i^{*+})}{A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_2^*G_1' \to}\,(\text{AU-L}^+)\ (\circ\text{-loop})\oplus}{\mathbf{B}_2A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \xrightarrow{+}}\,(BEL_2^*) \quad || \quad \dfrac{G_2 \xrightarrow{\circ}}{\circ G_2 \to}\,(\circ^+)}{\mathbf{B}_2A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \circ G_2 \to}\,(Weak_i^{*+})}{\uparrow}}{A^2_\emptyset(\mathbf{B}_2A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi) \xrightarrow{\circ}}\,(\text{AU-L}^+)}{\circ A^2_\emptyset(\mathbf{B}_2A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi) \to}\,(\circ^+)}{\mathbf{I}_1 \circ A^2_\emptyset(\mathbf{B}_2A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \cup \mathbf{I}_1\psi) \xrightarrow{+}}\,(INT_1^{or})}{\dfrac{\mathbf{I}_1 \circ G_2 \xrightarrow{+}}{S_3}}$$

Inference tree fragment for the sequent S_4 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\psi, \mathbf{B}_2^*\psi \to}}{\mathbf{B}_2\psi \to}\,(BEL_2^*) \qquad \dfrac{\dfrac{\overset{\text{(final)}}{\phi, \mathbf{B}_1^*\phi \to}}{\mathbf{B}_1\phi \xrightarrow{+}}\,(BEL_1^*) \quad || \quad \dfrac{\overset{(\circ\text{-loop})\oplus}{\dfrac{G_1 \xrightarrow{\circ}}{\circ G_1 \to}\,(\circ^+)}}{\circ A^1_\emptyset(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \to}}{\mathbf{B}_1\phi, \circ A^1_\emptyset(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \to}\,(Weak_i^{*+})}{A^1_\emptyset(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \xrightarrow{+\circ}}\,(\text{AU-L}^+)}{\circ A^1_{\{2\}}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi) \to}\,(\circ^+)}{\dfrac{\circ G_1' \to}{S_4}}$$

Inference tree fragment for the sequent S_5 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\to \phi, \mathbf{B}_1^*\phi}}{\uparrow}(BEL_1^*)}{\overset{+}{\to} \mathbf{B}_1\phi} \quad || \quad \dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\overset{+}{\to} \mathbf{B}_2\psi}(BEL_2^*)}{\to \mathbf{B}_1\phi, \mathbf{B}_2\psi}(Weak_i^{*+}) \qquad \dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\overset{+}{\to} \mathbf{B}_2\psi}(BEL_2^*) \quad || \quad \dfrac{\overset{\text{(}\circ\text{-loop)}}{\overset{\circ}{\to} G_1}}{\to \circ G_1}(\circ^+)}{\to \mathbf{B}_2\psi, \circ G_1}(Weak_i^{*+})}{\begin{array}{c}\uparrow\\ \uparrow\\ \uparrow\end{array}}}{\dfrac{\overset{\circ}{\to} A^1_{\emptyset}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi)}{\to \circ A^1_{\emptyset}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi)}(\circ^+)}(\text{AU-R}^+)}{}}{}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{\text{(final)}}{\to \phi, \mathbf{B}_1^*\phi}}{\uparrow}(BEL_1^*)}{\overset{+}{\to} \mathbf{B}_1\phi} \quad || \quad \dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\overset{+}{\to} \mathbf{B}_2\psi}(BEL_2^*)}{\to \mathbf{B}_1\phi, \mathbf{B}_2\psi}(Weak_i^{*+}) \qquad \dfrac{\dfrac{\overset{\text{(final)}}{\to \psi, \mathbf{B}_2^*\psi}}{\overset{+}{\to} \mathbf{B}_2\psi}(BEL_1^*) \quad || \quad \begin{array}{c}\uparrow\\ \uparrow\\ \uparrow\\ \uparrow\\ \uparrow\end{array}}{\to \mathbf{B}_2\psi, \circ A^1_{\emptyset}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi)}(Weak_i^{*+})}{\dfrac{\dfrac{\overset{+\circ}{\to} A^1_{\emptyset}(\mathbf{B}_1\phi \cup \mathbf{B}_2\psi), \mathbf{B}_2\psi}{\overset{+\circ}{\to} G_1, \mathbf{B}_2\psi}}{\dfrac{\to \circ G_1', \circ\mathbf{B}_2\psi}{S_5}}(\circ^+)}(\text{AU-R}^+)}{}$$

Since we use short formula notation (using G_1, G_1', G_2) to denote formulas, there are used nodes without any rule application. These nodes just rewrites the same sequent using other notation (short notation instead of the full notation, or conversely) and, in fact, they are not the part of the inference tree.

We write $\oplus$ to denote axioms, we write '(final)' to denote final sequents and we write '($\circ$-loop)' to denote loop-ending sequents. There are a lot of leafs those contain marked modalized formulas ($\mathbf{B}_1^*\phi$ or $\mathbf{B}_2^*\psi$). Technically, most of them are not final sequents, because rule $(Weak_i^{*+})$ may be applied. In order to make it readable, we mark such a branches by 'final' (and we skip last rule $(Weak_i^{*+})$ application). We may bypass redundant rule $(Weak_i^{*+})$ applications, if we use conatenated rules ($(Weak_i^{*+})$ with (BEL_i^*), (DES_i^{or}), (INT_i^{or}) and $(\circ^+)$) instead of the given ones as it is done in the previous chapters. For the BDI logics, concatenated rules are simple, but very large, and we do not use them just to make it more clear.

We presented full sequent S inference tree. It is obviously, that some tree branches may be left unfinished, since we can show that sequent S is non derivable in the sequent calculus BDI^n_{elc} even if some branches are left unfinished.

It is evident, that sequent S is non derivable in sequent calculus BDI^n_{elc} . There are a lot of or-branches in the sequent S inference tree, but most of them are very simple, since rule $(Weak_i^{*+})$ contain several premises, but sum of premises lengths is less or equal then conclusion lenght. Moreover, almost every rule $(Weak_i^{*+})$ premise is marked by $+$, and, therefore, loop-check is restriced on these premises.

5.5 Complexity Results for Sequent Calculus BDI^n_{elc}

In this section, complexity results for the introduced sequent calculus BDI^n_{elc} are presented. Sequent calculus for multiagent BDI logic was based on the loop-check free sequent calculus for $KD45$ logic and the sequent calculus with an efficient loop-check for branching time logic. Complexity results given in this section are obtained by summarizing results obtained in the Chapter 3 and in the Chapter 4.

We use different modalized formulas counts and formulas modality depth (see Definition 2.1.10) to evaluate sequent calculus BDI^n_{elc} complexity.

For the transparency, we ignore rule $(Weak^{*+}_i)$ applications in calculations at all, because rule $(Weak^{*+}_i)$ may be applied only together with one of the modal rules: (BEL^*_i), (DES^{or}_i), (INT^{or}_i), $(\circ^+)$.

In this section, if S is an initial sequent, then

- m_b denotes the count of the different modalized subformulas of the shape $\mathbf{B}_i\phi$ ($\mathbf{B}^*_i\phi$),
- m_d denotes the count of the different modalized subformulas of the shape $\mathbf{D}_i\phi$,
- m_i denotes the count of the different modalized subformulas of the shape $\mathbf{I}_i\phi$,
- m_a denotes the count of the different *Æ* subformulas (formulas having the shape $A^\beta_\alpha(\phi \cup \psi)$ or $E^\beta_\alpha(\phi \cup \psi)$),
- $m_\circ$ denotes the count of the different modalized subformulas having the shape $\circ\phi$, those are not extended *Æ* formulas,
- $m = m_a + m_\circ$,
- k' denotes logical operators count,
- $k = k' + 3 \cdot m_a$ (maximum count of the logical and until ((AU-L$^+$), (AU-R$^+$), (EU-L$^+$), (EU-R$^+$)) rules applied consecutively),
- d denotes initial sequent S modality depth.

As in the previous section, we have to mention, that m_a counts only *Æ* subformulas (not every extended *Æ* formula) and $m_\circ$ does not include extended *Æ* subformulas.

Lemma 5.5.1 Suppose that a sequent S has modality depth d and contains m different modalized subformulas (*Æ* subformulas or subformulas having the shape $\circ\phi$, those are not extended *Æ* formulas), then any weak $\circ$-loop $S \rightsquigarrow S'$ in the sequent calculus BDI^n_{elc} contains at most $(\frac{m}{d})^{d-1}$ rule $(\circ^+)$ applications.

Proof.

Since every weak $\circ$-loop $S \rightsquigarrow S'$ in the sequent calculus BDI^n_{elc} is an Until type rule, then, according to the Lemma 4.3.4., $\circ$-loop $S \rightsquigarrow S'$ in the sequent calculus BDI^n_{elc} contains at most $(\frac{m}{d})^{d-1}$ rule $(\circ^+)$ applications.

Lemma 5.5.2 Suppose that a sequent S contains, k' logical operators, m_d different modalized subformulas having the shape $\mathbf{D}_i\phi$ and m_i different modalized subformulas having the shape $\mathbf{I}_i\phi$, then there may be at most $k' + (m_d + m_i)$ rule (DES^{or}_i), (INT^{or}_i) and logical rules applied consecutively.

Proof.

The proof goes straightforward from the fact, that premise of any rule ((DES^{or}_i), (INT^{or}_i) or logical rule), contains less operators ($\mathbf{D}_i$, $\mathbf{I}_i$ or logical) then conclusion.

Lemma 5.5.3 Suppose that a sequent S has modality depth d and contains

- m_a different Æ subformulas (formulas having the shape $A^\beta_\alpha(\phi \cup \psi)$ or $E^\beta_\alpha(\phi \cup \psi)$),
- $m_\circ$ different modalized subformulas having the shape $\circ\phi$, those are not extended Æ formulas, and $m = m_a + m_\circ$,
- k' logical operators, and $k = k' + 3 \cdot m_a$,
- m_b different modalized subformulas having the shape $\mathbf{B}_i\phi$ ($\mathbf{B}^*_i\phi$),
- m_d different modalized subformulas having the shape $\mathbf{D}_i\phi$,
- m_i different modalized subformulas having the shape $\mathbf{I}_i\phi$.

Then any branch in a sequent S inference tree (in the sequent calculus BDI^n_{elc}) has height $\leq (k + m_d + m_i) \cdot m_b \cdot (\frac{m}{d})^d = (k' + 3 \cdot m_a + m_d + m_i) \cdot m_b \cdot (\frac{m_a + m_\circ}{d})^d$.

Proof.

The proof goes straightforward from the proves of the Lemmas 3.3.1, 4.3.3, 5.5.2.

Since sequent length $l > k' + m_a + m_\circ + m_b + m_d + m_i$, height of the inference tree in the sequent calculus BDI^n_{elc} is even less then for the one obtained for the sequent calculus PTL_{rlc} or BDI^n_{elc} ($m_a + m_\circ$ is a smaller part of the initial sequent length). In other words, complexity of the sequent calculus BDI^n_{elc} is not greater then complexity of the sequent calculus PTL_{rlc} .

Chapter 6

Conclusion

In this thesis, there are presented some new sequent calculi for BDI logics and their fragments. Presented calculi are either loop-check free sequent calculi (for $KD45$ logic) or sequent calculi with an efficient loop-check (for branching time and BDI logics). BDI logics and their fragments are widely used in multiagent agent systems implementation. Results, presented in this thesis, are very useful for such a multiagent systems implementation.

Results obtained for the fragments of the BDI logics (namely $KD45$ logic and branching time logic) are important not only for agent systems based on the BDI logics. $KD45$ logic is used to represent agents belief modality, which is one of the most popular modalities used in various agent systems. Branching time temporal logic is even more important, since temporal logics (Linear and branching time logics) are used almost in any agent system.

These are the main contribution of the thesis:

1. Created loop-check free sequent calculus for $KD45$ logic, which is used to represent agents beliefs in BDI architecture. Or-rules usage instead of the simple backtracking and some proven properties for $KD45$ logic enables us to use a new approach to the inference tree construction. If particular conditions are hold, some sequent on the inference tree, irrespective of the fact that the sequent is derivable or not by itself, may be treated as non derivable and its derivation are not proceeded. In this case, sequent derivability depends not only on the sequent itself, but on the other or-branches of the inference tree. This approach together with used marked modal operators enables us to construct loop-check free sequent calculus for $KD45$ logic.

2. Created sequent calculus with an efficient loop-check for branching time logic. Temporal logics are widely used, but complex logics. Inference trees in the introduced sequent calculus have the same size, but applied loop-check was strongly

restricted. Usual loop-check tests all ancestor sequents till the root. The main idea of the used restrictions is to use not all but several special marked sequents. Loop-check is performed only for these special sequents and only such a sequents are tested. Moreover, during inference tree construction, some sequents are marked by $+$. During loop-check, only the marked sequents placed above the sequents marked by $+$ must be tested. Indexes are used to determine sequents, those must be marked by $+$. These restrictions leads to the efficient loop-check.

3. Created sequent calculus with an efficient loop-check for monoagent BDI logic. Since BDI logic is a combination of some other modal logics, we applied obtained results for the fragments of the BDI logics to get efficient sequent calculus for BDI logic.

4. Created sequent calculus with an efficient loop-check for multiagent BDI logic. All intermediate results were summarized to construct full system for multiagent BDI logic, based on the sequent calculus with an efficient loop-check. Introduced sequent calculus is an efficient, sound and complete system applicable for the multiagent systems implementation.

Bibliography

[1] P. Abate, R.Gore, The Tableau WorkBench (TWB),
www [date: 2009-12-05]: http://twb.rsise.anu.edu.au,
Prover for propositional modal logic $KD45$,
www [date: 2009-12-05]:
http://twb.rsise.anu.edu.au/propositional_modal_logic_kd45.

[2] P. Abate, R.Gore, The Tableau WorkBench (TWB), Electronic Notes in Theoretical Computer Science, 2003

[3] M. Baaz, A. Leitsch, R. Zach, Completeness of a First order Temporal Logic with Time Gaps, Theoretical Computer Science, vol. 160, 1996, pp. 241–270.

[4] C.C. Bark, I.W. Geoffrey, Using Decision Trees for Agent Modeling: Improving Prediction Performance, User Modeling and User-Adapted Interaction archive, vol. 8, issue 1-2, 1998, pp. 131–152.

[5] A. Birštunas, Efficient decision procedure for Belief modality, Lithuanian Mathematical Journal, vol 45, spec. issue, 2005, pp. 321–325.

[6] A. Birštunas, Sequent calculus usage for BDI agent implementation, Lithuanian Mathematical Journal, vol 46, spec. issue, 2006, pp. 232–237.

[7] A. Birštunas, Efficient loop-check for $KD45$ logic, Lithuanian Mathematical Journal, vol 46, No. 1, 2006, pp. 44–53, Springer, New York.

[8] A. Birštunas, Efficient loop-check for multimodal $KD45_n$ logic, Lithuanian Mathematical Journal, vol 47, spec. issue, 2007, pp. 351–355.

[9] A. Birštunas, PSPACE complexity of modal logic $KD45_n$, Lithuanian Mathematical Journal, vol 48, No. 2, 2008, pp. 174–187, Springer, New York.

[10] A. Birštunas, Restrictions for loop-check in sequent calculus for temporal logic, Lithuanian Mathematical Journal, LMD works, vol 48-49, 2008, pp. 269–274.

[11] A. Birštunas, Restrictions for loop-check in sequent calculus for temporal logic with until operator, Lithuanian Mathematical Journal, LMD works, vol 50, 2009, pp. 247–252.

[12] A. Bolotov, M. Fisher, A resolution method for CTL branching-time temporal logic, in: Proc. of the 4th International Workshop on Temporal Representation and Reasoning (TIME '97), 1997, pp. 20–27

[13] H. Bordini, J.F. Hübner, BDI agent programming in AgentSpeak using Jason, in: CLIMA VI (Tutorial Paper), 2005, pp. 143–164.

[14] M.M. Cheikhrouhou, BDI-oriented agents for network management, in: GLOBECOM'99, vol 3, 1999, pp. 1964–1968.

[15] L. Chuchang, M.A. Ozols, M.A. Orgun, A fibred belief logic for multi-agent systems, Lecture Notes in Computer Science, vol. 3809, 2005, pp. 29–38.

[16] S. Coffey, D. Gaertner, Using pheromones, broadcasting and negotiation for agent gathering tasks, in: CLIMA VI, 2005, pp. 267–273.

[17] M. Dastani, F. Dignum, J.J. Meyer, 3APL: A Programming Language for cognitive agents, ERCIM News, European Research Consortium for Informatics and Mathematics, No. 53, spec. issue on Cognitive Systems, 2003.

[18] M. Dastani, J. Dix, The first contest on multi-agent systems based on computational logic, in: CLIMA VI, 2005, pp. 261–266.

[19] A. Cau, B. Moszkowski, H. Zedan, Interval Temporal Logic, 2009, www [date: 2009-12-07]:http://www.cse.dmu.ac.uk/STRL/ITL//.

[20] C. Dixon, F. Gago, M. Fisher, W. Hoek, Using temporal logics of knowledge in the formal verification of security protocols, Technical report ULCS-03-022.

[21] C. Dixon, M. Fisher, A Bolotov, Clausal resolution in a logic of rational agency, Artificial Intelligence archive, vol. 139, 2002, pp. 47–89.

[22] E.A. Emerson, Temporal and modal logic, in J. van Leeuwen (eds.), Handbook of Theoretical Computer Science, vol. B Formal Models and Semantics, 1990, pp. 995–1072.

[23] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, F. Orejas, A cut-free and invariant-free sequent calculus for PLTL, Lecture Notes in Computer Science, vol. 4646, 2007, pp. 481–495, Springer, Berlin/Heidelberg.

[24] R. Goldblatt, Mathematical Modal Logic: a View of its Evolution, Journal of Applied Logic, vol. 1, No. 5-6, 2003, pp. 309-392.

[25] R. Gore, Tableau Methods for Modal and Temporal Logics, in M. D'Agostino, R. Gabbay, R. Hähnle, J. Posegga (eds.), Handbook for Tableau Methods, 1999, pp. 297–396.

[26] R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi, Reasoning About Knowledge, The MIT Press, 1995.

[27] M. Fitting, Proof Methods for Modal and Intuitionistic Logics, 1983, D. Reidel, Dordrecht, Holland: Synthese Library.

[28] J.Y. Halpern, Y.O. Moses, Knowledge and common knowledge in a distributed environment, Journal of the Association for Computing Machinery, 37, 1990, pp. 549–587.

[29] A. Heuerding, M. Seyfried, and H. Zimmermann, Efficient loop-check for backward proof search in some non-classical propositional logics, in: P. Miglioli, U. Moscato, D. Mundici, M. Ornaghi (eds.), Tableaux 96, LNCS 1071, 1996, pp. 210–225.

[30] J.M. Howe, Two loop detection mechanisms: a comparison, in:
Proc. of TABLEAUX'97, LNAI 1227, 1997, pp. 188-200.

[31] N. Jennings, M. Wooldridge, Applications of intelligent agents, in: N. Jennings, M. Wooldridge (eds.), Agent Technology: Foundations, Applications, and Markets, 1998, pp. 3–28, Springer.

[32] R.A. Kowalski, F. Sadri, From logic programming towards multi-agent systems, Annals of Mathematics and Articial Intelligence, 25(3-4), 1999, pp. 391–419.

[33] S. Kraus, V.S. Subrahmanian, Multiagent reasoning with probability, time, and beliefs, International journal of intelligent systems, vol. 10, issue 5, 2007, pp. 459–499.

[34] F. Laroussinie, Ph. Schnoebelen, A hierarchy of temporal logics with past, Theoretical Computer Science, vol. 148, 1995, pp 303–324 .

[35] B. van Linder, W. van der Hoek, J.J. Ch. Meyer, Formalising motivational attitudes of agents: On preferences, goals and commitments, Intelligent Agents II, vol. 1037, 1996, pp. 17—32.

[36] J.J. Meyer, F. de Boer, R.van Eijk, K. Hindriks, W. van der Hoek, On programming KARO agents, Journal of KGPL, 9(2), 2001, pp.245–256.

[37] F. Massacci, Single step tableaux for modal logics. Computational Properties, Complexity and Methadology, Journal of Automated Reasoning, 24(3), 2000, pp. 319–364.

[38] M. Mouri, Constructing counter-models for modal logic K4 from refutation trees, Bull. Section of Logic, vol 31 No. 2, 2002, pp. 81–90.

[39] N. Nide and S. Takata, Deduction systems for BDI logic using sequent calculus, in: Proc. AAMAS'02, 2002, pp. 928–935.

[40] R. Pliuskevicius, A. Pliuskeviciene, Decision procedure for a fragment of mutual Belief logic with quantified agent variables, LNAI 3900, in: Proc. CLIMA VI, 2006, pp. 57–72.

[41] R. Pliuskevicius, A. Pliuskeviciene, Termination of derivations in a fragment of transitive distributed knowledge logic, Informatica, vol. 1, issue 4, 2008, pp. 597–616.

[42] M. Reynolds, Towards a CTL* Tableau, Lecture Notes in Computer Science, vol. 3821, 2005, pp. 384–395.

[43] O. Rana, M. Winikoff, L. Padgham, J. Harland, Applying conflict management strategies in BDI agents for resource management in computational grids, in: Proc. of the Australasian Conference on Computer Science, 2002, Melbourne, Australia, ACM Press.

[44] A.S. Rao, M. Georgeff, BDI agents: from theory to practice, in: Proc. of the First International Conference on Multi-Agent Systems (ICMAS-95), 1995, pp. 312–319, S. Francisco, CA.

[45] A.S. Rao, M. Georgeff, Decision Procedures for BDI Logics, Journal of Logic and Computation, 8(3), 1998, pp. 293–343

[46] A.S. Rao, AgentSpeak(L): BDI Agents speak out in a logical computable language, in: MAAMAW'96: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038, 1996, Springer.

[47] J. Sakalauskaite, Sequent calculi with analytic cut for logics of time and knowledge with perfect recall, Lithuanian Mathematical Journal, vol. 44, No. 2, 2004, pp.209–224.

[48] F. Shvarts, Gentzen style systems for K45 and KD45, in: Lecture Notes in Computer Science 363, 1989, pp. 245–256.

[49] A. Schmidt, D. Tishkovsky, U. Hustadt, Interactions between knowledge, action and commitment within agent dynamic logic, Journal Studia Logica, 2005, pp. 381–415.

[50] J.A. Torres, L.P. Nedel, R.H. Bordini, Using the BDI architecture to produce autonomous characters in virtual worlds, in Proc. of 4th International Workshop on Intelligent Virtual Agents, 2003, pp. 197–201.

[51] W. Wajs, M. Swiecicki, Neural Network Model of Autonomous Agent for Decision Support System, Modeiling and Simulation MODSIM 97, 1997.

[52] M. Wooldridge, Reasoning about Rational Agents, The MIT Press, 2000.

[53] M. Wooldridge, N. Jennings, Intelligent agents: theory and practice, The Knowledge Engineering Review, vol. 10, no. 2, 1995, pp. 115–152, Cambridge University Press.

[54] Ch. Yong, I.Y. Suk, Multi-agent Web Information Retrieval: Neural Network Based Approach, emphIDA 1999, pp. 499–512.

Appendix A

Decision Algorithm for $KD45$ Logic

In this appendix, we present decision algorithm, which is based on the loop-check free sequent calculus $KD45_{lcf}$. Algorithm determines whether sequent is derivable or not. We use pseudocode to present this algorithm.

We use deep first search algorithm to create and examine inference tree in the sequent calculus $KD45_{lcf}$. We use Stack to store nodes of the inference tree. Every node of the constructed tree contains: sequent, it's depth in the tree, operator (with possible values: AND, OR) and flag (with possible values: YES, NO, UNKNOWN). We can define structure of the node with the following pseudocode:

```
1. Node {
2.     Sequent Sequent;
3.     Depth Integer;
4.     Operator {AND, OR};
5.     Flag {YES, NO, UNKNOWN};
6. }
```

Suppose that sequents $S_1, S_2, \ldots, S_k$ are premises and S is conclusion of the inference rule application. Sequents $S_1, S_2, \ldots, S_k$ have Depth equal to the sequent S Depth $+1$. If applied inference rule is $(\Box^{LCF})$ when Operator is set to OR, otherwise Operator is set to AND. So, Operator defines the type of the branches. Flag is set to YES, if sequent on this node is derivable, Flag is set to NO, if it is non derivable, and Flag is set to UNKNOWN, if we still do not know whether it is derivable or not.

Now we introduce pseudocode of the algorithm which is based on the sequent calculus $KD45_{lcf}$:

```
1. function deriveSequent(Sequent S) {
2.     Node N = new Node();
3.     N.Sequent = S;
4.     N.Depth = 0;
5.     N.Operator = AND;
6.    N.Flag = UNKNOWN;
7.    do {
8.        N = performOneNode(N);
9.    } while (N.Depth > 0);
10.   writeResult(N.Flag);
11. }
```

In lines 2-6, we create the root node of the tree.

In lines 7-9, we check nodes of the tree till we reach the root node (with a result set in it).

In line 10, we write obtained result.

The main work is performed in the function performOneNode. This function gets the node with a sequent as a parameter. If the node contains untested sequent (Flag = UNKNOWN), then it applies suitable rule, creates child nodes and puts them in to the Stack (creates new branches of the inference tree). If the node contains already tested sequent (Flag = YES, or NO), then it makes backtracking (throws nodes from the Stack).

Pseudocode of this function is listed bellow:

```
1. function Node performOneNode(Node N) {
2.    if (N.Flag == UNKNOWN) {
3.        if (isAxiom(N)) {
4.            N.Flag = YES;
5.            return N;
6.        } else {
7.            Rule R = getPossibleRule(N);
8.            if (R != null) {
9.                Stack.put(N);
10.               Node[] newN = applyRule(N.Sequent, R);
11.               for (int i = 0; i < newN.length; i ++) {
12.                   newN[i].Depth = N.Depth + 1;
13.                   newN[i].Operator = getOperatorType(R);
14.                   newN[i].Flag = UNKNOWN;
15.                   Stack.put(newN[i]);
16.               }
17.               Node M = Stack.get();
18.               return M;
19.           } else { // - rules cannot be applied
20.               N.Flag = NO;
21.               return N;
22.           }
23.         }
24.     } else if (N.Flag == YES) {
25.         if (N.Operator == AND) {
26.             Node M = Stack.get();
27.             return M;
28.         } else { // if (N.Operator = OR)
29.             Node M;
30.             do {
31.                 M = Stack.get();
33.             } while(M.Depth == N.Depth);
33.             M.Flag = YES;
34.             return M;
35.         }
36.     } else { // if (N.Flag == NO)
37.         if (N.Operator == AND) {
38.             Node M;
39.             do {
40.                 M = Stack.get();
41.             } while(M.Depth == N.Depth);
42.             M.Flag = NO;
43.             return M;
44.         } else { // if (N.Operator = OR)
45.             Node M = Stack.get();
46.             return M;
47.         }
48.     }
49. }
```

In lines 2-23, we check sequents derivability.

In line 3, we check if it is an axiom.

In lines 4-5, we return positive result, because it is an axiom, and, therefore, node sequent is derivable.

In lines 7-18, we creates new branches of the inference tree.

Function getPossibleRule returns the rule, which can be applied for the sequent, or null if there is no such a rule. All loop-check is performed within this function, because if no rule can be applied, then we know, that some loop exist.

Function applyRule applies selected rule and returns obtained sequents.

Function getOperatorType returns OR, if rule was $(\Box^{LCF})$, and AND, otherwise.

In lines 20-21, we return negative result, because no rules can be applied and we treat such a node sequent as non derivable.

In lines 24-35, we backtrack then child node was derivable.
In lines 36-48, we backtrack then child node was non derivable.