Liet. matem. rink, **45**, *spec. nr.*, 2005, 321–325

Efficient decision procedure for Belief modality

Adomas BIRŠTUNAS (VU)
e-mail: adomo@takas.lt

Abstract. This paper defines decision algorithm for subclass of $B^{KD45}D^{KD}I^{KD}$ logic which is based on known algorithm for temporal $B^{KD45}D^{KD}I^{KD}$ logic ([2]). BDI logics are widely used in agent based systems. Such usage of BDI logic can be found in [1]. The original decision algorithm uses loop-check technique for BEL and temporal operators. Applied loop-check technique is not optimized and therefore loop-check takes most of the time used in decision algorithm. Some examples of efficient loop-check applications for logic KT, S4 and some subclasses of intuitionistic logic can be found in [4]. Another efficient loop-check can be found in work [3]. We concentrate on our attitude on loop-check optimization for BEL operator. This paper defines decision algorithm modification, which uses efficient loop-check for BEL operator, but do not effect performance of other parts of algorithm. We define optimization only for BEL operator and therefore we omit temporal operators in this paper.

Keywords: BDI logic, loop-check, sequent calculus..

1. Definition of researched subclass

Researched subclass of $B^{KD45}D^{KD}I^{KD}$ logic are described as follows:

Let P be a set of all propositional symbols. If p is a primitive preposition ($p \in P$), then p is formula, if ϕ, ψ are formulas, then $\neg\phi$, $\phi \vee \psi$, $BEL(\phi)$ are also formulas. Formulas $\phi \wedge \psi$, $\phi \supset \psi$, $\phi \Leftrightarrow \psi$ are abbreviations of $\neg(\neg\phi \vee \neg\psi)$, $\neg\phi \vee \psi$, $(\phi \supset \psi) \wedge (\psi \supset \phi)$ respectively.

We use common Kripke 'possible worlds' semantics. In other words, we will operate with all BDI logic formulas those do not contain $INTEND$ or $DESIRE$ modalities. $INTEND$, $DESIRE$ are omitted because of the lack of the space.

Decision algorithms are based on sequent calculus for $B^{KD45}D^{KD}I^{KD}$ logic. We will use not all inference rules, because some of them are redundant then working with subclass of $B^{KD45}D^{KD}I^{KD}$ logic (see [2] for all inference rules including ones for temporal operators). Sequent calculus deals with sequents which has multi-sets on both sides of $\rightarrow$. During decision algorithms we will use inference rules $\neg L$, $\vee L$, $\neg R$, $\vee R$, $Weak$ (see [2]) and ($BEL - KD45$):

$$\frac{\Gamma, BEL(\Gamma) \rightarrow \Theta, BEL(\Theta), BEL(\Delta)}{BEL(\Gamma) \rightarrow BEL(\Theta), BEL(\Delta)}$$

Here, Θ must be at most one formula, ϕ and ψ – formulas, Γ, Δ – finite (may be empty) sets of formulas.

2. Decision algorithms

Now we introduce several definitions used in original and modified algorithms.

DEFINITION 1. Tree node with 2 sequents S, S' we will call circle-type node if none of the rules except BEL or $Weak$ can be applied to second sequent S'. Tree node with one sequent $\widehat{S}'$ and one formula Θ we will call box-type node if all formulas in sequent $\widehat{S}'$ has top-level operator BEL and Θ is one of $\widehat{S}'$ formulas.

DEFINITION 2. If circle-type node N with second sequent S has ancestor circle-type node N^* with second sequent S^* and sequents S and S^* are the same, then we will say, that we have circle-type loop $N^* \Rightarrow N$. If box-type node $\widehat{N}$ with sequent $\widehat{S}$ has ancestor box-type node $\widehat{N}^*$ with sequent $\widehat{S}^*$ and sequents $\widehat{S}$ and $\widehat{S}^*$ are the same, then we will say, that we have box-type loop $\widehat{N}^* \Rightarrow \widehat{N}$.

Now we introduce procedures based on inference rules used during algorithms.

Procedure A. Apply $\neg R$, $\neg L$, $\vee R$, $\vee L$ to first sequent of the circle-type node S as far as possible. Delete all sequents which are sequent calculus axioms. **End.**

All sequents $S'_1, S'_2, \ldots$ obtained by Procedure A satisfies such conditions:

- S'_i consist only of formulas with top-level operator BEL or are atomic propositions and all formulas are sub-formulas of formulas in sequent S.
- Sequent S is invalid iff at least one of $S'_1, S'_2, \ldots$ is invalid. (1)

Procedure B. Apply Procedure B1 for sequent S'. For every pair $< \widehat{S}'_i, \Theta_i >$ got during Proc. B1 apply Proc. B2 (B1, B2 definitions are placed below). **End.**

All sequents $S''_1, S''_2, \ldots$ obtained by Procedure B satisfies:

- Sequent S' is invalid iff all of $S''_1, S''_2, \ldots$ are invalid. (2)

Procedure B1. Let S' be sequent obtained by Proc. A. Delete all formulas in S' leaving only those top-level operator is BEL. Choose one of the formulas from the right side of $\rightarrow$ and mark chosen formula as Θ. Result of this procedure are all different pairs $< \widehat{S}'_i, \Theta_i >$(empty marked formula is included). **End.**

Procedure B2. Let $< \widehat{S}'_i, \Theta_i >$ be a pair obtained by Proc. B1. For $\widehat{S}'$ apply rule $BEL - KD45$ with Θ'_i as the main formula of the rule and get new sequent S''. This procedure generates one sequent for every pair $< \widehat{S}'_i, \Theta'_i >$. **End.**

Original algorithm (here named Algorithm 1) uses A and B, but newly introduced decision algorithm uses B1 and B2 instead of B. In this work we introduce informal Algorithm 1 (fully described in [2]), because of the lack of the space.

Algorithm 1. For every second sequent S' of the circle-type node N' apply Proc. B and get $S''_1, S''_2, \ldots$. For every S''_i apply Proc. A and get sequents $S'''_{i,1}, S'''_{i,2}, \ldots$. For every $S'''_{i,j}$ create circle-type node $N'''_{i,j}$ with sequents S''_i, $S'''_{i,j}$. Join node N' with all nodes $N'''_{i,j}$. If some node ends circle-type loop mark it with letter 'c'. Repeat these actions for newly got not marked nodes as far as possible.

You will get a tree of circle-type nodes with the root node containing initial sequent S. All marked leaf-node's second sequents are invalid, not marked – valid. Delete valid branches from tree by going down from leaves to root and applying conditions (1), (2). Initial sequent S is valid iff all nodes are deleted. **End.**

Proof of decision algorithm completeness is placed in [2].

We introduce modified algorithm (here named Algorithm 2) which uses efficient loop-check. Main idea of algorithm is to check box-type loops instead of circle-type.

For this reason we insert new box-type nodes between every pair of joined circle-type nodes without adding additional circle-type nodes. So, Algorithm 2 constructs tree consisted of circle and box type nodes.

Algorithm 2.

1. Apply Proc. A for initial sequent S_0, get sequents $S'_1, S'_2, \ldots$. For S'_i create new circle-type node N'_i with sequents S_0, S'_i.
2. For every circle-type node of $N'_1, N'_2, \ldots$ apply the following (we will write N' for taken circle-type node, and S' for the second sequent of this node):
   2.1. Apply Proc. B1 for S' and get pairs $< \widehat{S}'_1, \Theta_1 >, < \widehat{S}'_2, \Theta_2 >, \ldots$. For every pair $< \widehat{S}'_i, \Theta_i >$ create new box-type node $\widehat{N}'_i$ with sequent $\widehat{S}'_i$ and marked formula Θ_i. Join node N' with nodes $\widehat{N}'_1, \widehat{N}'_2, \ldots$.
   2.2. For every pair $< \widehat{S}'_i, \Theta_i >$ (box-type node $\widehat{N}'_i$) apply the following:
      2.2.1. If exist box-type loop $\widehat{N}^* \Rightarrow \widehat{N}'_i$ mark node $\widehat{N}'_i$ with letter 'c'.
      2.2.2. If not, apply Proc. B2 for pair $< \widehat{S}'_i, \Theta_i >$ and get sequent S''_i. Then apply Proc. A for sequent S''_i and get $S'''_{i,1}, S'''_{i,2}, \ldots$. For every sequent $S'''_{i,j}$ create new circle-type nodes $N'''_{i,j}$ with sequents S''_i, $S'''_{i,j}$. Join node $\widehat{N}'_i$ with nodes $N'''_{i,j}$. For every created node $N'''_{i,j}$ recursively apply Step 2.
3. Delete valid branches from constructed tree. Delete all circle-type leaf-nodes from the tree. Let that circle-type node N has box-type children $\widehat{N}'_1, \widehat{N}'_2, \ldots$. If at least one of $\widehat{N}'_i$ is a leaf-node and is not marked with 'c', delete all nodes $\widehat{N}'_i$ (with all nodes above them). Repeat this step until all tree leaf-nodes will be marked with 'c' or no node will be left.
4. Sequent S_0 is valid iff all nodes are deleted during Step 3.

End.

3. Efficiency of used loop-check

At this point nothing better can be found in Algorithm 2 then in Algorithm 1. To show improvement of performance we have to prove some lemmas. First we will show that algorithms are equivalence. After we will show, that loop-check used in Algorithm 2 is more efficient then loop-check used in Algorithm 1.

LEMMA 1. *If $N \Rightarrow N'$ is a loop (circle or box type) in a tree then all sequents inside a loop nodes has the same formulas with top-level operator BEL.*

Proof. Loop begins and ends with the same sequent and during Algorithm 2 steps we do not delete any formula with top-level operator BEL.

If we eliminate all box-type nodes we will get some subtree of the tree constructed according to Algorithm 1 (tree construction in Algorithm 2 and Proc. B definition based on B1, B2). Let we have a tree constructed during Algorithm 1 before deletion (Step 3). Insert corresponding box-type node between every joined circle-type nodes

to get tree constructed according Algorithm 2 (but without box-type loop-check at all). Lemmas (their proofs) bellow will deal with such a tree.

LEMMA 2. *If sequent is invalid according to Algorithm* 1 *then it will be invalid according to Algorithm* 2 *too.*

Proof. If $N \Rightarrow N'$ is circle-type loop in a tree and box-type nodes $\widehat{N}, \widehat{N}'$ are children of N and N', then there exists box-type loop $\widehat{N} \Rightarrow \widehat{N}'$ (box-type node's sequent can have only formulas with top-level operator BEL and Lemma 1). Every tree branch contains circle-type loop for Algorithm 1 if sequent is invalid, then every such tree branch will contain box-type loop too, and we get a proof.

During Step 3 of Algorithm 1 we can delete box-type node $\widehat{N}'$ in two cases: a) if all children of node $\widehat{N}'$ are deleted, b) if $\widehat{N}'$ has father N' which is deleted because some other child of N' was deleted or it is true for other ancestor of $\widehat{N}'$.

It is important to mention that in case b) sub-tree above $\widehat{N}'$ do not impact $\widehat{N}'$ deletion at all. The next lemma says that we can use only case b) for deletion box-type node which ends some box-type loop.

LEMMA 3. *Let we have box-type loop* $\widehat{N} \Rightarrow \widehat{N}'$ *in constructed tree and node* $\widehat{N}'$ *was deleted during Step* 3. *Then there exist another (different from* $\widehat{N}'$*) child of* $\widehat{N}'$ *father which was deleted or it is true for some other ancestor of* $\widehat{N}'$.

Proof. Where can be two cases of box-type loop $\widehat{N} \Rightarrow \widehat{N}'$ – then nodes $\widehat{N}$ and $\widehat{N}'$ has the same marked Θ formula (I) and then has different one (II).

I) Box-type nodes $\widehat{N}, \widehat{N}'$ has identical children. We will check too sub-cases: all children of $\widehat{N}'$ are leaves, and then some of them are not.

If all $\widehat{N}'$ children are leaves, then at least one are marked with 'c' (otherwise we cannot get box-type loop, because all children of $\widehat{N}$ will be leaves). If there exist $\widehat{N}'$ child marked with letter 'c', then $\widehat{N}'$ can not be deleted according to a).

If there exist $\widehat{N}'$ child which is not leaf, then let that loop $\widehat{N} \Rightarrow \widehat{N}'$ goes through $\widehat{N}$ child N_1 and it's child $\widehat{N}_{1,1}$: $\widehat{N} \rightarrow N_1 \rightarrow \widehat{N}_{1,1} \Rightarrow \widehat{N}'$. Let circle-type node N_1' be a child of $\widehat{N}'$ and equal to N_1 (such child exist because children of $\widehat{N}, \widehat{N}'$ are identical). N_1' cannot be a leaf (otherwise we cannot have a loop). Box-type node $\widehat{N}'$ can be deleted according to a) only if all its children are deleted, including N_1'. Let that $\widehat{N}'_{1,1}, \widehat{N}'_{1,2}, \ldots$ are children of N_1'. N_1' can be deleted if at least one of $\widehat{N}'_{1,1}, \widehat{N}'_{1,2}, \ldots$ was deleted. Let deleted child to be $\widehat{N}'_{1,i}$. If $\widehat{N}'_{1,i}$ equals to $\widehat{N}_{1,1}$, then we have box-type loop $\widehat{N} \Rightarrow \widehat{N}'_{1,i}$ and can apply the same arguments for $\widehat{N}'_{1,i}$ as to $\widehat{N}'$, till reach the leaves of the tree. If $\widehat{N}'_{1,i}$ is different from $\widehat{N}_{1,1}$, then circle-type node N_1 has child (different from $\widehat{N}_{1,1}$) which is equal to $\widehat{N}'_{1,i}$. So, $\widehat{N}_{1,1}$ can be deleted according to b) and some ancestor of node $\widehat{N}'$ can be deleted according to b).

II) In this case father of $\widehat{N}$ has another child $\widehat{N}^*$ (different from $\widehat{N}$) which is the same as $\widehat{N}'$. So, if during Step 3 node $\widehat{N}'$ is deleted, then it can be deleted because $\widehat{N}$ was deleted according to b).

LEMMA 4. *If sequent is valid according to Algorithm* 1 *then it will be valid according to Algorithm* 2 *too.*

Proof. If we check box-type loops, every box-type loop ending node wont be deleted according to a) (Algorithm 2 definition), but can be deleted according to b) (Lemma 3). If sequent is valid, then all nodes will be deleted from the tree including box-type loop-check, therefore sequent is valid according to Algorithm 2.

DEFINITION 3. If box-type loop $\widehat{N} \Rightarrow \widehat{N}'$ consist of $n+1$ box-types nodes, then we will say, that box-type loop $\widehat{N} \Rightarrow \widehat{N}'$ has length n.

LEMMA 5. *Every box-type loop* $\widehat{N} \Rightarrow \widehat{N}'$ *has length* 1.

Proof. All box-type nodes has only sequents those all formulas has BEL as their top level operator. According to Lemma 1 all sequents inside a loop has the same formulas with top-level operator BEL. So, all box-type nodes inside loop $\widehat{N} \Rightarrow \widehat{N}'$ has identical sequents and therefore loop has length 1.

We have to notice that such fact is wrong for circle-type loops.

Lemma 5 says that box-type loop-check is more efficient then previously used circle-type loop-check, because for loop detection we can check only the last box-type node, not all nodes placed below.

References

1. M. Wooldridge, *Reasoning about Rational Agents*, The MIT Press (2000).
2. N. Nide, S. Takata, Deducton systems for BDI logic using sequent calculus, *Proc. of AAMAS'02*, 928–935 (2002).
3. M. Mouri, Constracting counter-models for modal logic K4 from refutation trees, *Bulletin of the Section of Logic*, **31** (2), 81–90 (2002).
4. A. Heuerding, M. Seyfried, 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.), *LNCS*, **1071**, 210–225 (1996).

REZIUMĖ

A. Birštunas. Efektyvi sprendimo procedūra modaliniam operatoriui Belief

Pateiktas sprendimo radimo algoritmas BDI logikos formulių klasei, kuris naudoja efektyvų ciklų radimo mechanizmą modaliniam operatoriui Belief.