Argumentation System with Changes of an Agent's Knowledge Base

Kenichi Okuno
Graduate School of Science & Technology
Kwansei Gakuin University
o.kenichi.gm@gmail.com

Kazuko Takahashi
School of Science & Technology
Kwansei Gakuin University
ktaka@kwansei.ac.jp

Abstract

This paper discusses a process of argumentation. We propose an algorithm for dynamic treatment of argumentation in which all lines of argumentation are executed in succession, and the agent's knowledge base can change during argumentation. We show that there exists a case in which an agent dynamically loses argumentation that would be considered won by a static analysis. We also show that the algorithm terminates, and describe acceptable arguments that are obtained after the argumentation.

1 Introduction

Argumentation is an important concept and a logical basis for legal reasoning and agent communication. Dung constructed a logical framework for argumentation [Dung, 1995]. He defined an argumentation framework as a set of arguments and a set of pairs with an attack relation, and showed that this is effective for logic programming and nonmonotonic reasoning. He also gave semantics for an argumentation theory and defined acceptable sets based on these semantics. An acceptable set is considered to be a set of arguments which both agents accept after the argumentation.

Subsequently, many studies have been undertaken based on this argumentation framework, including applications for defeasible logic programming [Garcia and Simari, 2004; Chesnevar *et al.*, 2005; Prakken, 2006; Chesnevar and Simari, 2007], belief revision [Falappa *et al.*, 2002; Paglieri and Castelfranchi, 2004; Takahashi and Sawamura, 2004], and negotiation between agents [Kraus *et al.*, 1998; Amgoud *et al.*, 2000].

Generally, argumentation proceeds as follows: two agents, a proposer and a defeater, each make arguments attacking the other's utterances in turn. A tree form is proposed to represent the structure of argumentation. The root node is a proposed formula [Amgoud *et al.*, 2000; Garcia and Simari, 2004], and each branch corresponds to a single argumentation line, namely, a sequence of arguments. Usually the constraints of loop-freeness and consistency of the proposer's utterances are imposed on each argumentation line. Each branch is considered to be argumentation in a possible world that is independent of the other branches, so relationships between branches are not considered. However, in actual argumentation, after one line of argumention ends, argumentation along another line follows. At that time, the information gained in one argumentation line is reflected in the following argumentation. Our aim is to provide a system that simulates such actual argumentation.

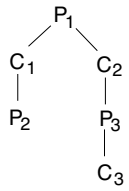


Figure 1: Structure of argumentation

Let us consider an example of argumentation. We suppose a situation in which a murderer P tells a lie, "I did not commit murder," and a policeman C argues that it is a lie. P_i and C_i show P's and C's i-th utterances, respectively.

> P_1: "I did not commit murder! There is no evidence!"
>
> C_1: "There is evidence. We found your license near the scene."
>
> P_2: "It's not evidence! I had my license stolen!"
>
> C_2: "It is you who killed the victim. Only you were near the scene at the time of the murder."
>
> P_3: "I didn't go there. I was at facility A at that time."
>
> C_3: "At facility A? No, that's impossible. Facility A does not allow a person to enter without a license. You said that you had your license stolen, didn't you?"

Figure 1 shows the structure of this argumentation. In the argumentation system proposed previously, argumentation proceeds along the left branch, and, if C has no grounds upon which to attack P_2, then the argumentation ends with P the winner.

However, in practical argumentation, C continues a counterargument in the right branch which attacks P_1 from another side. Finally, C points out the contradiction between P's utterances and wins. P's utterance P_2 gives C new information and causes C to generate C_3.

In the argumentation system proposed previously, the execution of each single argumentation line is considered separately, and successive execution of all argumentation lines is not considered, thereby excluding this example. To capture the behaviour in the example, we also have to consider the change of knowledge bases caused by exchanging arguments.

In this paper, we formalize argumentation from a more practical viewpoint. We propose a successive execution of all argumentation lines while allowing agents' knowledge bases to change. The argument is treated operationally, considering that execution of argumentation changes an agent's knowledge base. We also keep a history of the proposer's utterances. This is accumulated during the argumentation and is finally equivalent to the acceptable set of the argumentation.

This paper is organized as follows. Section 2 provides the definitions of basic concepts such as argumentation and argumentation tree. Section 3 proposes an execution algorithm for argumentation incorporating with changes of an agent's knowledge base and discusses its properties. Section 4 provides an example of this algorithm. Section 5 compares the proposed approach with related works. Finally, section 6 presents conclusions.

2 Argumentation

2.1 Argumentation Framework

Definition 1 (consistent) *Let Ψ be a set of formulas in propositional logic. If there does not exist ψ that satisfies both $\psi \in \Psi$ and $\neg\psi \in \Psi$, Ψ is said to be* consistent.

The knowledge base $\mathbf{K}_\mathrm{a}$ for each agent a is a finite set of propositional formulas. Note that $\mathbf{K}_\mathrm{a}$ is not necessarily consistent and may have no deductive closure; that is, there may be a case in which $\phi, \phi \rightarrow \psi \in \mathbf{K}_\mathrm{a}$ and $\psi \notin \mathbf{K}_\mathrm{a}$ hold. An agent a participates in argumentation using elements of $\mathbf{K}_\mathrm{a}$.

Definition 2 (support) *For a nonempty set of formulas Ψ and a formula ψ, if there exist ϕ, $\phi \rightarrow \psi \in \Psi$, then Ψ is said to be a* support for ψ.

Definition 3 (argument) *Let $\mathbf{K}_\mathrm{a}$ be a knowledge base for an agent a.* An argument of a *is a pair (Ψ, ψ) where Ψ is a subset of $\mathbf{K}_\mathrm{a}$, and $\psi \in \mathbf{K}_\mathrm{a}$ such that Ψ is the empty set or a consistent support for ψ.*

For an argument $A = (\Psi, \psi)$, Ψ and ψ are said to be *grounds* and *a sentence of* A, respectively. They are denoted by $Grounds(A)$ and $Sentence(A)$, respectively. $S(A)$ denotes $Grounds(A) \cup \{Sentence(A)\}$. If $\psi \in S(A)$, it is said that *a formula ψ is contained in an argument A.*

Definition 4 (preference) *Each formula is assigned a preference value. Let $\nu(\psi)$ be the preference for a formula ψ. Then, the preference of an argument A is defined by* $\sum_{\psi \in S(A)} \nu(\psi)$.

Preferences are assigned depending on the strength, certainty, or stability of formulas. Note that a formula is assigned the same preference regardless of the knowledge base in which it is contained so as to avoid loops in the argumentation.

Definition 5 (attack) *Let $AR_{\mathbf{K}_\mathrm{a}}$ and $AR_{\mathbf{K}_\mathrm{b}}$ be sets of all possible arguments of agents a and b, respectively.*

1. *If $Sentence(A_a) \equiv \neg Sentence(A_b)$ and $\nu(A_a) \geq \nu(A_b)$, then (A_a, A_b) is said to be* a rebut *from a to b.*
2. *If $\neg Sentence(A_a) \in Grounds(A_b)$ and $\nu(A_a) \geq \nu((\emptyset, \neg Sentence(A_a)))$, then (A_a, A_b) is said to be* an undercut *from a to b.*
3. An attack *from a to b is either a rebut or an undercut from a to b.*

When (A_a, A_b) is an attack from a to b, it is said that A_a *attacks* A_b.

Let S be a set of arguments. S *attacks* an argument B if B is attacked by an argument in S. S *defends* an argument A if S attacks each argument which attacks A.

Definition 6 (acceptable arguments) *Let S be a set of arguments and $F(S)$ denote the set of arguments which S defends. If there are no arguments $A, B \in S$ such that A attacks B, the least fixed point with respect to set inclusion of F is defined to be a set of* acceptable arguments.

Based on Dung [Dung, 1995], in an argumentation framework between two agents, a proposer P makes the first argument and a defeater C makes counterarguments. Hereafter, $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{C}$ denote their knowledge bases, respectively.

Definition 7 (argumentation framework) *Let $AR_{\mathbf{K}_\mathrm{P}}$ and $AR_{\mathbf{K}_\mathrm{C}}$ be sets of all possible arguments of P and C, respectively, with preferences ν. Let $AT_{\mathbf{K}_\mathrm{P} \rightarrow \mathbf{K}_\mathrm{C}}$ and $AT_{\mathbf{K}_\mathrm{C} \rightarrow \mathbf{K}_\mathrm{P}}$ be sets of attacks from P to C and those from C to P, respectively.* An argumentation framework *between P and C, $AF(\mathbf{K}_\mathrm{P}, \mathbf{K}_\mathrm{C}, \nu)$ is defined as a quadruple $\langle AR_{\mathbf{K}_\mathrm{P}}, AR_{\mathbf{K}_\mathrm{C}}, AT_{\mathbf{K}_\mathrm{P} \rightarrow \mathbf{K}_\mathrm{C}}, AT_{\mathbf{K}_\mathrm{C} \rightarrow \mathbf{K}_\mathrm{P}} \rangle$.*

2.2 Argumentation Tree

Definition 8 (move) A move *is a pair of a player (an agent) P/C and an argument A where $A \in AR_{\mathbf{K}_\mathrm{P}}/AR_{\mathbf{K}_\mathrm{C}}$. If $player$ is P/C, then it is said to be P/C's move.*

Definition 9 (move's attack) *Let $move_i = (player_i, A_i)$ and $move_j = (player_j, A_j)$. $move_i$ is said to be* an attack to *$move_j$, if (A_i, A_j) is an attack from $player_i$ to $player_j$.*

Definition 10 (argumentation line) *Let P,C denote a proposer of a formula φ and its defeater. Let also $AF(\mathbf{K}_\mathrm{P}, \mathbf{K}_\mathrm{C}, \nu)$ be an argumentation framework between P and C.* An argumentation line *D for φ on $AF(\mathbf{K}_\mathrm{P}, \mathbf{K}_\mathrm{C}, \nu)$ is a finite nonempty sequence of moves $[move_1, \ldots, move_n]$ where $move_i = (player_i, A_i)$ $(i = 1, \ldots, n)$ that satisfies:*

1. *$move_1 = (P, A_1)$ where $Sentence(A_1) = \varphi$.*
2. *If i is odd, then $player_i = P$, and if i is even, then $player_i = C$.*
3. *$move_{i+1}$ is an attack to $move_i$ for each i $(1 \leq i \leq n-1)$.*
4. *There is no attack against A_n.*
5. *$move_i \neq move_j$ for each pair of i, j $(1 \leq i \neq j \leq n)$.*

Definition 11 (win of an argumentation line) *If the last element of an argumentation line D is P's move, then it is said that P* wins *D.*

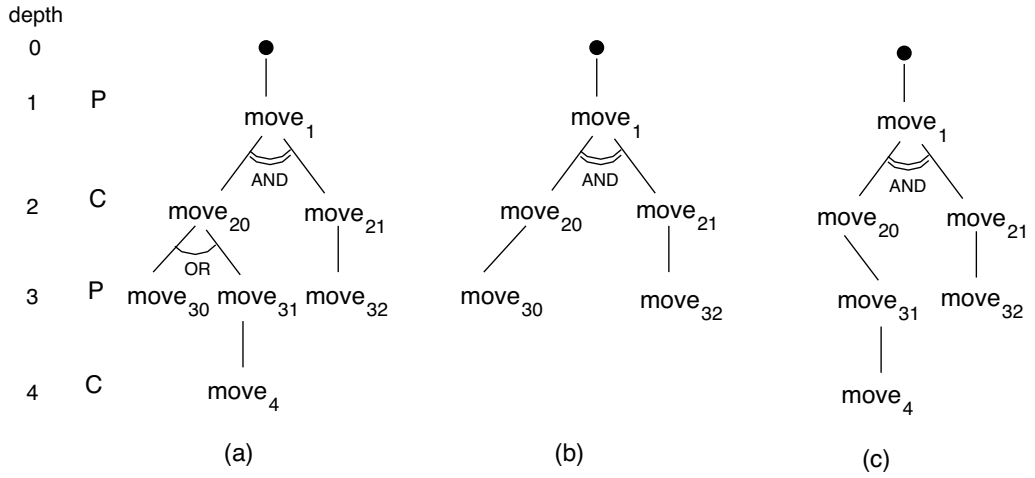


Figure 2: An argumentation tree and its candidate subtrees

Definition 12 (argumentation tree) An argumentation tree for φ on $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ *is a tree where the root node at depth 0 is empty and all the branches*[1] *starting from the node of depth one are different argumentation lines for* φ *on* $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$.

An argumentation tree can be considered to be an AND/OR tree. The nodes corresponding to C's move are OR nodes, whereas the nodes corresponding to P's move are AND nodes. This reflects the fact that P needs to defend all counterarguments made by C, whereas it is sufficient for C to select its own counterargument that leads to a win. Arguments in the nodes of depth one have φ as sentences but different grounds. Therefore, the root node is also an OR node.

Definition 13 (win branch) *If P wins an argumentation line D, the branch representing D in the argumentation tree is said to be a* win branch.

Definition 14 (candidate subtree) A candidate subtree *is a subtree of an argumentation tree that selects only one child node for each node corresponding to C's move in the original tree, and selects all child nodes for each node corresponding to P's move.*

Definition 15 (solution subtree) A solution subtree *is a candidate subtree in which P* wins *all of the argumentation lines in the tree.*

Each candidate subtree corresponds to a proposer's selection of an argument, and the solution subtree indicates the case in which P takes a winning strategy. In Figure 2, (a) is an argumentation tree, (b) and (c) are its candidate subtrees, and (b) is the solution subtree.

Definition 16 (static win of an argumentation tree) *If an argumentation tree has a solution subtree, then P* statically wins *the argumentation tree; otherwise,* statically loses *it.*

[1]Here, a branch means a path from the designated node to a leaf node.

3 Argumentation with Changes of Knowledge Base

3.1 Execution of Argumentation

We now present the dynamic treatment of argumentation.

When P statically wins an argumentation line, C is considered to consent to the result, because there are no more counterarguments to the last argument. Therefore, C's knowledge base is revised when argumentation of a win branch is finished. However, a counterargument to a move in a middle node of the argumentation line may still remain. Here, P will need also to defeat this counterargument using a revised knowledge base.

Definition 17 (justified) *Let* $[move_1, \ldots, move_n]$ *be a win branch in an argumentation tree. If* $move_j$ *has no AND-branch and* $player_j$ *is P where* $1 \leq j \leq n$*, then an argument corresponding to* $move_j$ *is said to be* justified *after the execution of the win branch.*

For example, assume that argumentation of a win branch $[move_1, move_{20}, move_{30}]$ in Figure 2(a) is executed. P's argument $move_{30}$ is justified but $move_1$ is not, since there is another possible counterargument $move_{21}$ to $move_1$.

Definition 18 (change of a knowledge base) *Let* $\mathbf{K}_{\mathrm{C}}$ *be a knowledge base and* $move_i = (player_i, A_i)$*. The change of* $\mathbf{K}_{\mathrm{C}}$ *by* $move_i$*, denoted by* $\mathbf{K}_{\mathrm{C}} * move_i$*, is defined as follows: if* $player_i = P$*, then* $(\mathbf{K}_{\mathrm{C}} \setminus \{\neg Sentence(A_i)\}) \cup S(A_i)$*; if* $player_i = C$*, then* $\mathbf{K}_{\mathrm{C}}$.

$\mathbf{K}_{\mathrm{P}}$ is never changed, whereas $\mathbf{K}_{\mathrm{C}}$ may be changed after accepting P's argument, and C may attempt a counterargument based on the revised knowledge base. As a result, the formulas appearing in C's arguments may be inconsistent. However, the formulas appearing in P's arguments within the set of the executed argumentation lines are consistent.

Definition 19 (history) *The set of formulas contained in P's preceding arguments is said to be* a history of P's arguments.

Definition 20 (dynamic argumentation line) *Let P,C denote a proposer of a formula φ and its defeater. Let also $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ be an argumentation framework between P and C.* A dynamic argumentation line D *for φ on $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ with a history $\mathbf{H}$ is defined as the extension of the (static) argumentation line by adding the following additional condition.*

6. $\mathbf{H} \cup S(A_i)$ is consistent if i is odd.

If no misleading is involved, *a dynamic argumentation line* for φ on $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ with a history $\mathbf{H}$ is said to be just *an argumentation line* on $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ by omitting φ and $\mathbf{H}$.

When the argumentation procedure allows change in the agents' knowledge bases, a win branch of a candidate subtree is selected and the change of C's knowledge base by the corresponding argumentation line is performed. The argumentation tree is reconstructed using the revised knowledge base. This process is repeated, and, when the proposal is contained in C's knowledge base while its negation is not contained, P wins the argumentation tree. We show below this Argumentation Procedure with Knowledge Change (*APKC*).

Argumentation Procedure with Knowledge Change (*APKC*)

Let $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ be an argumentation framework, φ be a proposed formula, and $\mathbf{H}$ be a history of P's arguments.
[STEP0(initialization)]
Set $\mathbf{H} = \emptyset$.
[STEP1((re)construction of a tree)]
Construct an argumentation tree for $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$ on φ with $\mathbf{H}$.
[STEP2(execution of an argumentation line)]
if $\varphi \in \mathbf{K}_{\mathrm{C}}$ and $\neg\varphi \notin \mathbf{K}_{\mathrm{C}}$
then terminate with success.
else if there is no win branch
then terminate with failure.
else
select a win branch $[move_1, \ldots, move_n]$
where $move_i = (player_i, A_i)$.
if there exists a justified argument in these moves,
then set i to be the minimum number of such a move.
else set $i = 0$.
set $\mathbf{K}_{\mathrm{C}} = (\ldots(\mathbf{K}_{\mathrm{C}} * move_n) * move_{n-1}) * \ldots * move_{i+1}$.
set $\mathbf{H} = \mathbf{H} \cup \{S(A_i) \mid 1 \leq i \leq n, i \text{ is } odd\}$.
goto STEP1.

Note that there are multiple ways to select a win branch, and *APKC* can decide the success or failure of a certain execution.

The change of $\mathbf{K}_{\mathrm{C}}$ is nonmonotonic. Therefore, in the reconstructed tree, new nodes may be added at the leaves of branches other than the executed one, and several existing nodes are removed. The nodes corresponding to P's moves that contradict $\mathbf{H}$ will also disappear.

3.2 Properties of APKC

First, we prove the termination of *APKC*, by showing that the nodes decreases in number at every resonstuction of a tree, and that each reconstruction terminates.

Lemma 1 *Let $T_0, \ldots, T_k$ be a sequence of argumentation trees, each of which is constructed for the revised knowledge base. If a move at depth more than zero in T_i ($0 \leq i \leq k-1$) is executed, then it does not appear either in $T_{i+1}, \ldots, T_k$.*

Proof
(i) C's move:
When P executes a move that is a parent node of C's move, some formulas included in the argument of C's move are removed from $\mathbf{K}_{\mathrm{C}}$. Therefore, the argument does not exist in $AR_{\mathbf{K}_{\mathrm{C}}'}$, where $\mathbf{K}_{\mathrm{C}}'$ is a revised knowledge base of $\mathbf{K}_{\mathrm{C}}$. A formula ψ is removed from $\mathbf{K}_{\mathrm{C}}$ only when a move with sentence $\neg\psi$ is executed. Therefore, if ψ is removed from $\mathbf{K}_{\mathrm{C}}$, $\neg\psi$ must be in the history $\mathbf{H}$. It follows that the move with sentence ψ is not included in the argumentation line. In contrast, assume that a formula ψ is added to $\mathbf{K}_{\mathrm{C}}$. It means that the move with sentence ψ is included in the argumentation line. This is a contradiction. Therefore, ψ is not added to $\mathbf{K}_{\mathrm{C}}$ again. Hence, if a formula is removed from $\mathbf{K}_{\mathrm{C}}$, it is never added again. Therefore, C's executed move never appears in any subsequent reconstructed argumentation trees.
(ii) P's move:
When P's move at a depth greater than zero is executed, C's move at its parent node is executed. C's executed move never appears in any reconstructed argumentation trees from (i). Therefore, P's executed move does not appear either. Q.E.D.

Lemma 2 *The (re)construction of an argumentation tree terminates.*

Proof
$AR_{\mathbf{K}_{\mathrm{P}}} \cup AR_{\mathbf{K}_{\mathrm{C}}}$ is a finite set. For each node corresponding to a move, the number of child nodes is finite, because the moves are taken from $AR_{\mathbf{K}_{\mathrm{P}}} \cup AR_{\mathbf{K}_{\mathrm{C}}}$. The length of a branch is finite because the same move never appears more than once in a branch. Therefore, the number of possible argumentation trees is finite. It follows that the reconstruction step terminates. Q.E.D.

Theorem 1 (termination) *The algorithm* APKC *terminates.*

Proof
From lemma 2, the reconstruction step of an argumentation tree terminates, and the number of nodes in an argumentation tree is finite. From lemma 1, the number of nodes appearing in an argumentation tree decreases at every execution of a move.

In *APKC*, reconstruction of a tree is performed every time more than one move is executed. Therefore, for an arbitrary win branch, either the length of the branch is finally decreased to one or the branch is finally changed to one that is not a win branch. In the former case, C's knowledge base is revised upon the execution of a move at depth one in the tree, that is, P's move, where the sentence is a proposal φ. Then the algorithm terminates with the result $\varphi \in \mathbf{K}_{\mathrm{C}}$, $\neg\varphi \notin \mathbf{K}_{\mathrm{C}}$. In the latter case, it terminates with P's defeat. Q.E.D.

A set of justified arguments is a subset of $\mathbf{H}$, and the justified arguments increase in number along with the execution of win branches. When *APKC* terminates with success, it is equivalent to $\mathbf{H}$. Therefore, we have the following theorem.

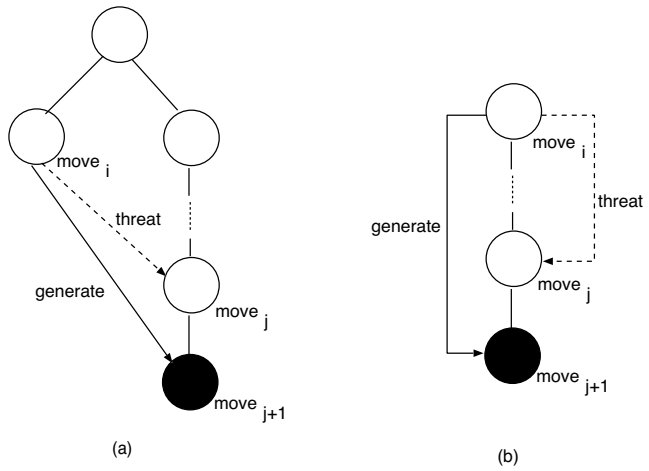


Figure 3: A threat

Theorem 2 (acceptable arguments) *When APKC terminates with success, then the resulting* $\mathbf{H}$ *is a set of acceptable arguments which includes the proposed formula* φ*.*

The set of acceptable arguments may be different depending on the execution order. In addition, *APKC* terminates with success for some execution orders and fail for other execution orders. A winning strategy for P will be to take the former order.

A dynamic win/loss of an argumentation tree can be defined based on *APKC*.

Definition 21 (dynamic solution subtree) *Let* CT *be a candidate subtree of an initial argumentation tree. For any order of execution of win branches of* CT*, the algorithm* APKC *terminates with success, and if more than one selection of candidate subtrees is generated by the addition of P's move, and at least one of them satisfies this condition, then* CT *is said to be* a dynamic solution subtree*.*

Definition 22 (dynamic win of an argumentation tree) *If an argumentation tree has a dynamic solution subtree, then P* dynamically wins *the argumentation tree; otherwise, P* dynamically loses *it.*

In the reconstruction of a tree, a node which does not appear in the initial tree may be generated. We introduce the concept of a *threat* to explain this situation.

Definition 23 (threat) *Let* $move_i$ *and* $move_j$ *be moves in an argumentation tree on* $AF(\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{C}}, \nu)$*. If* $\mathbf{K}_{\mathrm{C}} * move_i$ *results in the creation of a new move* $move_{j+1}$*, which is an attack to* $move_j$*, then* $move_i$ *is said to be* a threat to $move_j$ *(Figure 3(a)). If no win branch contains* $move_{j+1}$*, then* $move_i$ *is said to be* a strict threat to $move_j$*.*

Intuitively, a strict threat is a move that provides information advantageous for a defeater. A move may be a threat to a move in the same branch (Figure 3(b)), however, we do not consider such a case here, since we assume that the argumentation line, once executed, is not changed.

An agent can statically win an argumentation tree even if the initial argumentation tree contains a strict threat, because such a threat will not be detected in the initial argumentation tree. It follows that an agent who statically wins does not always dynamically win.

Theorem 3 (relationship of static win and dynamic win) *If P dynamically wins an argumentation tree* T*, then P statically wins* T*, but not vice versa.*

Proof
The former part of the statement is trivial by the definition. As for the latter part, the next section presents an example which shows a case P statically wins but dynamically loses. Q.E.D.

4 An Example

Let us consider an example shown in Section 1. The knowledge bases of a proposer and a defeater are shown below. The number attached to each formula shows its preference. We assume that the facts and rules are all represented in the knowledge base, and the agents have no other knowledge.

$$\mathbf{K}_{\mathrm{P}} = \left\{ \begin{array}{l} \neg m[1],\ \neg e[2],\ (\neg e \rightarrow \neg m)[1],\ \neg(la \rightarrow e)[1], \\ ls[1],\ (ls \rightarrow \neg(ls \rightarrow e))[1],\ \neg n[1],\ a[2], \\ (a \rightarrow \neg n)[1] \end{array} \right\}$$

$$\mathbf{K}_{\mathrm{C}} = \left\{ \begin{array}{l} e[1],\ la[1],\ (la \rightarrow e)[2],\ m[2],\ n[2], \\ (n \rightarrow m)[1],\ \neg a[1],\ (ls \rightarrow \neg a)[1] \end{array} \right\}$$

The propositions have the following meanings:

m:	P commits murder.
e:	There is evidence.
la:	P's license was left at the scene of the murder.
ls:	P's license was stolen.
n:	P was near the scene when the murder was committed.
a:	P was at facility A when the murder was committed.

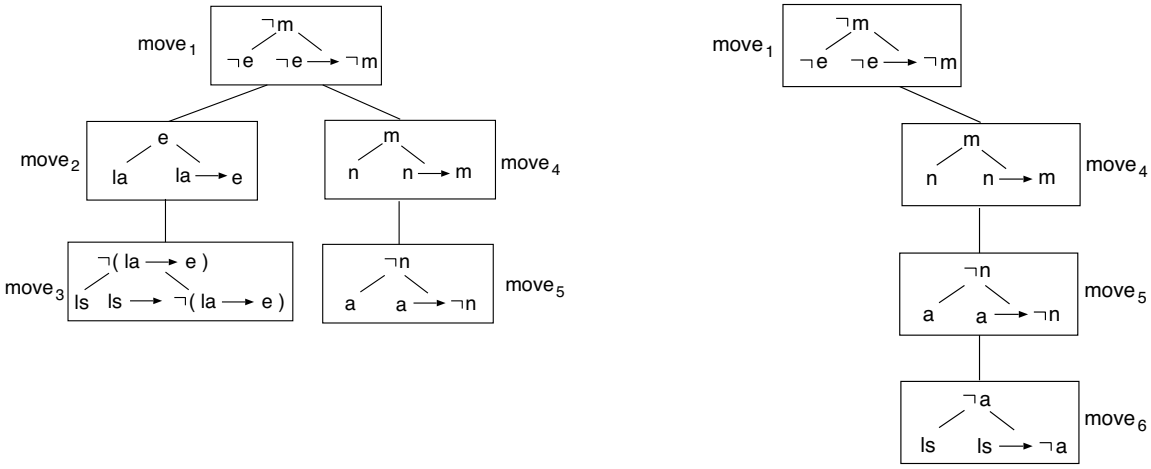
APKC first constructs an initial argumentation tree. For simplicity, only the part relevant to the discussion is shown in Figure 4(a). P statically wins, since all leaves of this tree are P's moves.

The argumentation is executed along the left branch, and C's knowledge base is revised. The following three formulas are added to $\mathbf{K}_{\mathrm{C}}$ and $ls \rightarrow e$ is removed from $\mathbf{K}_{\mathrm{C}}$.

$$\neg(la \rightarrow e), \quad ls, \quad ls \rightarrow \neg(la \rightarrow e)$$

As a result, the left branch is removed. Moreover, P's argument in $move_3$ causes a new fact ls to be added to $\mathbf{K}_{\mathrm{C}}$. It follows that P can provide a new counterargument in $move_6$, which is not defeated, to the leaf of the right branch. The reconstructed argumentation tree is shown in Figure 4(b). It contains only one argumentation line. P loses this argumentation line, since its leaf node is C's move. Therefore, *APKC* terminates in failure. In this case, $move_3$ is a strict threat to $move_5$. Note that if the argumentation along the right branch is executed first, $move_6$ is not added.

This example is a case in which P statically wins but dynamically loses the argumentation tree.



(a) an initial argumentation tree (b) a reconstructed argumentation tree

Figure 4: The argumentation trees

5 Related Works

García applied argumentation to defeasible logic programming. He considered argumentation to be an explanation and proposed a model in which argumentation is evaluated when a claim is accepted [Garcia *et al.*, 2007]. In his model, evaluation of argumentation is a dialectical proof procedure that is performed by traversing a constructed dialectical tree. Moguillansky discussed revision of the knowledge base represented in the form of defeasible logic programming [Moguillansky *et al.*, 2008]. These works both examined reconstruction of the tree with the revised knowledge base, but their goal was to construct undefeated argumentation by selecting suitable defeasible rules and not to consider the effect of the execution of argumentation.

Several works regard argumentation as a dialogue exchanging information between agents [Kraus *et al.*, 1998; Amgoud *et al.*, 2000; Amgoud and Cayrol, 2002; Paglieri and Castelfranchi, 2004]. An argument is regarded as a communication protocol between agents. In most models, an agent rejects a proposal if it contradicts his or her knowledge base and accepts it otherwise, and in the end agreement may be achieved. In these models, an agent's behaviour is determined by the arguments he or she receives, but his or her knowledge base never changes during the argumentation.

Amgoud formalized a negotiation system in an argumentation framework [Amgoud *et al.*, 2007]. She considered the knowledge base for each agent separately and its revision by exchanging arguments. The significant difference between her work and ours is that in her approach only a single argumentation line is considered, so only threats to the same branch are taken into account, whereas in our approach all argumentation lines are considered successively, so threats to the other branches are examined. Dunne proposed "dispute tree" on which successive execution of all argumentation lines are considered [Dunne and Bench-Capon, 2003]. However, the revision of agents' knowledge base which means that the executed move may add new information to the opponent's knowledge base is not considered.

Recently, Cayrol presented interesting research on the revision of an argumentation theory [Cayrol *et al.*, 2008]. She investigated how acceptable arguments are changed when an argument is added. The aim of her research is a formal analysis under changes to argumentation, and the contents of the additional arguments and reasons for the addition are beyond her scope. In contrast, we focus specifically on the effect of knowledge gained by executing argumentation.

6 Conclusion

We have proposed an argumentation system in which multiple argumentation lines are executed in succession, and an agent's knowledge base can change during argumentation. This system can accommodate practical phenomena ignored by argumentation systems so far proposed.

We have implemented the system in Prolog, for the core algorithm, and Java, for the interface. In our implementation, a win branch is selected not automatically but rather manually by a user. The system displays the process of argumentation and shows the final judgment of dynamic win or loss.

In the future, we will extend this system to cases that involve the type of threat shown in Figure 3(b). We will also consider improvements to an agent's ability to use logical reasoning to derive new facts from added information.

References

[Amgoud *et al.*, 2000] L.Amgoud, S.Parsons, and N.Maudet: Arguments, dialogue, and negotiation, ECAI2000, pp.338-342, 2000.

[Amgoud and Cayrol, 2002] L.Amgoud, C.Cayrol: Inferring from inconsistency in preference-based argumenta-

tion frameworks, J. of Automated Reasoning, pp.125-169, 2002.

[Amgoud *et al.*, 2007] L.Amgoud, Y.Dimopolos and P.Moraitis: A general framework for argumentation-based negotiation, ArgMAS2007, pp.1-17, 2007.

[Cayrol *et al.*, 2008] C.Cayrol, F.D.de St-Cyr, and M-C Lagasquie-Shiex: Revision of an argumentation system. pp.124-134, KR2008, 2008.

[Chesnevar *et al.*, 2005] C.I.Chesñevar, A.Maguitman and R.Loui: Logical models of argument. ACM Computing Surveys, 32(4), pp.337-383, 2005.

[Chesnevar and Simari, 2007] C.I.Chesñevar and G.R.Simari: A lattice-based approach to computing warranted beliefs in skeptical argumentation frameworks. IJCAI2007, pp.280-285, 2007.

[Dunne and Bench-Capon, 2003] P.E.Dunne and T.J.M.Bench-Capon: Two party immediate response disputes: properties and efficiency. Artificial Intelligence, 149(2), pp.221-250, 2003.

[Dung, 1995] P.M.Dung: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence, 77, pp.321-357, 1995.

[Falappa *et al.*, 2002] M.Falappa, G.Kern-Isberner and G.R.Simari: Explanations, belief revision and defeasible reasoning, Artificial Intelligence, 141(1-2), pp.1-28, 2002.

[Garcia and Simari, 2004] A.García, and G.Simari: Defeasible logic programming: an argumentative approach. Theory and practice of logic programming, 4(1), pp.95-138, 2004.

[Garcia *et al.*, 2007] A.García, C.Chesñevar, N.Rotstein, and G.Simari: An abstract presentation of dialectical explanations in defeasible argumentation, ArgNMR07, pp.17-32, 2007.

[Kraus *et al.*, 1998] S.Kraus, K.Sycara and A.Evenchik: Reaching agreements through argumentation: a logical model and implementation, Artificial Intelligence, 104(1-2), pp.1-69, 1998.

[Moguillansky *et al.*, 2008] M.O.Moguillansky, et al.: Argument theory change applied to defeasible logic programming, AAAI2008, pp.132-137, 2008.

[Prakken, 2006] H.Prakken: Combining skeptical epistemic reasoning with credulous practical reasoning. COMMA 2006, pp.311-322, 2006.

[Paglieri and Castelfranchi, 2004] F.Paglieri and C.Castelfranchi: Revising beliefs through arguments: bridging the gap between argumentation and belief revision in MAS, ArgMAS2004, pp.78-94, 2004.

[Takahashi and Sawamura, 2004] T.Takahashi and H.Sawamura: A logic of multiple-valued argumentation, AAMAS2004, pp.789-805, 2004.