Logical Aspects of Events: Quantification, Sorts, Composition and Disjointness

Ken Kaneiwa[1] **Satoshi Tojo**[2]

[1]National Institute of Informatics, Japan
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
Email: kaneiwa@nii.ac.jp
[2]Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-1292, Japan
Email: tojo@jaist.ac.jp

Abstract

An event, as opposed to an atemporal property, has its own time and location and occurs once. Although the notion of events has been found in researches on ontology, logic, linguistics, artificial intelligence and deductive databases, the different approaches to this notion do not seem to capture the various logical aspects of events. This paper proposes an event logic with expressions such as quantification over events, event sort-hierarchy, and composition and disjointness of events. In the logic, events are regarded as constants, sorts, predicates and variables in an order-sorted second-order language, which provides knowledge representation and reasoning for event assertions. In order to implement a query answering system, we present a sorted tableau calculus for the refutation of event formulas in logic.
Keywords: event-ontology, taxonomy, quantification, reasoning

1 Introduction

The notion of events is one of the entities classified in philosophical ontology, and in order to describe actions and changes in the real world, the formal representation of events has been investigated in logic, linguistics, artificial intelligence and deductive databases. In (Davidson 1980, Sadri & Kowalski 1995), event sentences are expressed by predicate formulas including event identifiers. For example, $(\exists e)Kicked\ (Shem, Shaun, e)$ states an event e such as 'Shem kicked Shaun.' Events as linguistic constructions (van Benthem 1983) have been understood to be a useful perspective for representing natural language sentences. According to Allen et al. (Allen & Ferguson 1994) events were methods used to classify useful and relevant patterns of change rather than entities in the real world. Sowa (Sowa 2000) categorized events as changes that occur in the discrete steps of a process. Landman (Landman 1991) gave semantics of events by structures, where each structure comprises a set of events and precedence, inclusion and overlap relations over events. Events in active object-oriented database (Gatziu & Dittrich 1994, Galton & Augusto 2002) have also been studied. The database system SAMOS (Gatziu & Dittrich 1994) provides event specification facilities to describe the complex events constructed by disjunction, conjunction, sequence of events, negative events, etc. Galton and Augusto (Galton & Augusto 2002) attempted to combine the two kinds of event definitions of knowledge representation and databases.

However, these approaches have the following two drawbacks. First, they do not sufficiently capture the various logical aspects of events since each approach treats only a sole aspect of events. In fact, using logical languages, event entities can be regarded differently as constants, sorts, predicates and variables in knowledge representation. Second, these approaches do not adequately define the syntax, semantics and reasoning mechanisms of event logic. Logical formalization enables a rigorous analysis of the features of events in ontology and information systems.

In this paper, by extending the order-sorted logic (Kaneiwa 2004, Kaneiwa & Mizoguchi 2004, Kaneiwa & Mizoguchi 2005, Kaneiwa & Tojo 2001), we present an event logic that provides knowledge representation and reasoning for different aspects of events. Due to the sorted expressions with variables and predicates, order-sorted logic is useful as a basic logical language to specify a diversity of events.

This paper is arranged as follows. In Section 2, we consider that the use of events leads to various kinds of logical expressions, and propose a method of representing event assertions by extending order-sorted logic. In Section 3, we introduce the basic notions of order-sorted logic. In Section 4, we define the syntax and semantics of an order-sorted event logic. Section 5 develops a sorted tableau calculus that includes additional inference rules with respect to predicate arguments and quantification, composition and disjointness over events. In Section 6, we give conclusions and discuss future work.

2 Logical aspects of events

In this section we analyze different logical aspects caused by the use of event entities, and consider an extension of expressions in order-sorted logic in order to handle the diversity of events. The notions of constants, sorts, variables and predicates can be used for suitably expressing event assertions and rules in a logical language.

2.1 Constants, sorts, variables and quantification

The content of an event e_i can be represented by a predicate formula

$$p(o_1, \ldots, o_n)$$

where p is an n-ary predicate and $o_1, \ldots, o_n$ are individuals. We call this formula $p(o_1, \ldots, o_n)$ a *(detailed) description of an event constant* e_i (or an event identifier). In knowledge representation, it is

Copyright ©2005, Australian Computer Society, Inc. This paper appeared at the Australasian Ontology Workshop (AOW 2005), Sydney, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 58. T. Meyer, M. Orgun, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

necessary to distinguish such event descriptions from the property descriptions of the formula $p(o_1, \ldots, o_n)$ (representing properties p of individuals $o_1, \ldots, o_n$). However, both types of descriptions are identically defined as sets of tuples of individuals in first-order semantics.

Any event constant can correspond to the occurrence of an event. Hence, if the predicate formula $cook(John, meat)$ is a description of 'John cooked meat' of an event constant e_1, then it is possible that numerous other event constants are described by the same formula:

$$e_1 = cook(John, meat),$$
$$e_2 = cook(John, meat),$$
$$\ldots,$$
$$e_n = cook(John, meat).$$

In line with this observation, the predicate *cook*, as a binary relation of the individuals *John* and *meat*, can play another role as a sort and unary predicate of event constants. In other words, there are sorts and unary predicates over events (called event sorts and event predicates). For example, $e_1 : cook$ expresses an event constant e_1 of the event sort *cook*, and $cook(e_1)$ implies that an event constant e_1 has a property of the event predicate *cook*. The expressions of sorts and predicates over events are conceivable as a natural extension of sorts and predicates over individuals. In (Kaneiwa & Tojo 1999, Tojo & Kaneiwa 2003), Kaneiwa and Tojo formalized an expressive logical language with event sorts (or event predicates) that contains two types of sort-hierarchies such that one hierarchy is constructed by sorts over individuals and the other is constructed by sorts (or predicates) over events. Let $\mathcal{ES}$ denote the set of event sorts. We now discuss the structural feature of events using event sorts. Let $boil, cook, act \in \mathcal{ES}$ be event sorts. Then, the declaration

$$boil \sqsubseteq_E cook \sqsubseteq_E act$$

represents a hierarchical relationship among event sorts. An event sort-hierarchy refers to a pair $(\mathcal{ES}, \sqsubseteq_E)$ of the set $\mathcal{ES}$ of event sorts and a partial order over $\mathcal{ES}$. As discussed above, each event sort indicates a set of event constants. Thus, if $cook(e_1)$ holds true, then $act(e_1)$ can be derived from $cook \sqsubseteq_E act$. Moreover, provided that the predicate formula $cook(John, meat)$ is a detailed description of an event constant e_1, a less informative description $act(John, meat)$ of the event e_1 can be inferred in the event sort-hierarchy. It should be noted that the symbol is employed not only as a binary predicate over individuals but also as a sort or unary predicate over events.

Apart from expressions of individuals, we have thus far explained the existence of constants, sorts and predicates with regard to the notion of events. Let E_1 be an event sort, e_1 be an event constant, and $o_1, \ldots, o_n$ be individuals. We express an event assertion as follows:

$$e_1 : E_1(o_1, \ldots, o_n).$$

This signifies that an event constant e_1 belongs to an event sort E_1; furthermore, $E_1(o_1, \ldots, o_n)$ is a detailed description of the event constant e_1. For example, $e_1 : cook(John, meat)$ expresses that the event description 'John cooked meat' took place as an event constant e_1.

The means of representing event rules require variables and quantifications over events. We formalize an event constant by a predicated symbol and an event variable by a predicate variable, respectively. By generalizing an event assertion $e_1 : E_1(o_1, \ldots, o_n)$, we obtain an open event assertion $X : E_1(o_1, \ldots, o_n)$, including an event variable X; by quantifying this event variable,

$$(\forall X : E_1)X : E_1(o_1, \ldots, o_n)$$

states that for every event X in E_1, the event X consisting of $(o_1, \ldots, o_n)$ occurred. Using event variables and quantifications, the following event rule can be described:

$$(\forall X : heat)X : heat(water) \rightarrow (\exists Y : boil)Y : boil(water).$$

This rule means 'if water is heated, then water will boil.' More precisely, 'for every event X of water being heated, there exists an event Y such that it boils.'

2.2 Occurrence, composition and disjointness

We take implicit or explicit account of the time and location of event assertions. Suppose that an event assertion $e_1 : E_1(o_1, \ldots, o_n)$ holds. Then, the time and location at which event e_1 occurred should also hold. Let t and l denote the time and location. These symbols can be added to the predicate arguments as follows:

$$e_1 : E_1(o_1, \ldots, o_n, t, l).$$

If the time and location are missing or omitted, then the event assertion $e_1 : E_1(o_1, \ldots, o_n)$ should be interpreted as implicitly asserting

$$(\exists x : tim)(\exists y : loc)e_1 : E_1(o_1, \ldots, o_n, x, y).$$

This is motivated by the case in which two different events e_1, e_2 have the same description in $e_1 : E_1(o_1, \ldots, o_n)$ and $e_2 : E_1(o_1, \ldots, o_n)$ but their times and locations are not exactly the same. For example, although $e_1 : twinkle(star)$ and $e_2 : twinkle(star)$ are embodied by the same description 'a star is twinkling,' the two event constants e_1, e_2 are denoted differently. Hence, these events might occur at separate times and locations, such as $e_1 : twinkle(star, t_1, l_1)$ and $e_2 : twinkle(star, t_2, l_2)$.

Regarding event assertions wherein the time and location are explicit, we attempt to deal with an event operation (a composition of two events) and an event relation (disjointness between two events). Let the following event assertions be assumed:

$$e_1 : E_1(o_1, \ldots, o_n, t_1, l_1),$$
$$e_2 : E_2(o'_1, \ldots, o'_m, t_2, l_2).$$

A prerequisite to composing the two events is that their argument components: individuals, time and location are identical; more formally, $o_1 = o'_1, \ldots, o_n = o'_m (n = m)$[1] and $t_1 = t_2$, $l_1 = l_2$. When this prerequisite is satisfied, the composed event is obtained as follows:

$$e_1 \circ e_2 : E_1 \sqcap E_2(o_1, \ldots, o_n, t_1, l_1).$$

For example, if the event assertions

$$e_1 : talking(john, mary, t, l),$$
$$e_2 : angry(john, mary, t, l).$$

hold true, then the composed event

$$e_1 \circ e_2 : talking \sqcap angry(john, mary, t, l)$$

[1] In this study, we assume that the order of argument roles is prearranged, e.g., o_1 is a subject, o_2 is an object, o_3 is a tool, etc.

expresses 'John is angrily talking with Mary.' However, the composition operation does not work for event assertions in cases where their times and locations are not specified. In this situation, a minimum requirement is that the two events occur simultaneously and at the same location even without an explicit time and location. For the simultaneity, temporal and locational relations over events are stated as follows:

$$e_1 : E_1 \approx_t e_2 : E_2,$$
$$e_1 : E_1 \approx_l e_2 : E_2.$$

The former implies that the events e_1, e_2 occurred simultaneously, and the latter implies that they occurred at the same location.

The disjointness between events indicates that their respective events cannot take place simultaneously, unlike the disjointness between classes (or sorts) in object-oriented modeling languages, e.g., UML class diagrams (Berardi, Cali, Calvanese & Giacomo 2005, Kaneiwa & Satoh 2006). The disjointness can be specified by event sorts but not by event constants. Let E_1, E_2 be event sorts. Then, the disjointness relation between E_1 and E_2 will be declared by

$$E_1 ||_E E_2.$$

For instance, we can assume the disjointness $win||_E lose$ where *win* and *lose* are event sorts, i.e., no one can win and lose at the same time. It should be noted that even if there are two disjoint events, they can take place at the same location when their times are different. As an example, the event assertions $e_1 : win(John, t_1, l)$ and $e_2 : lose(John, t_2, l)$ imply that 'at location l, John won at time t_1 but he lost at time t_2.' In some cases, *win* and *lose* may be an identical event if the agent and its counter-agent alternate. Suppose *John* and *Bob* had a boxing or sumo wrestling match. Then, $win(John, Bob)$ 'John won to Bob.' can be the exact same event as $lose(Bob, John)$ 'Bob lost to John'. However, in our formulation in the following section the arguments are sensitive to their order, and as far as the first arguments of two events coincide *win* and *lose* cannot be compatible.

The disjointness provides us with yet another prerequisite of event composition such that the sorts of composed events must not be disjoint. Events in the event sorts *angry* and *talking* can be composed as $e_1 \circ e_2 : angry \sqcap talking$ because their sorts are not disjoint, whereas the composition $e_1 \circ e_2 : win \sqcap lose$ is inadequate due to the disjointness of the event sorts *win* and *lose*. The event composition prohibits different components in the descriptions of target event assertions. However, this condition appears to be too strong because it excludes the compositions of many event assertions if their descriptions are predicate formulas comprising various argument structures. For example, the composition of $e_1 : drive(John, car)$ and $e_2 : eat(John, food)$ is impossible since the second arguments *car* and *food* cannot be unified. In order to resolve this problem, we look to argument manipulations (proposed by Kaneiwa and Tojo (Kaneiwa & Tojo 1999)) that enable us to obtain the following event composition. First, we delete a surplus argument from the descriptions of the two events and then conclude less informative event assertions. In the example below, by deleting both the second arguments in the event descriptions, the less informative assertions $e_1 : drive(John)$ and $e_2 : eat(John)$, which state that 'John is driving' and 'John is eating' can be inferred. Hence, if the event sorts *drive* and *eat* are not disjoint, and the simultaneity of $e_1 : drive \approx_t e_2 : eat$ and $e_1 : drive \approx_l e_2 : eat$ holds, then we can obtain the event composition $e_1 \circ e_2 : drive \sqcap eat(John)$ that states 'John is eating while driving.'

Regarding temporal relationships between events, we introduce two relations over events: $\lhd_t$ (time inclusion) and $\prec_t$ (time precedence). In the above representation, these relations are declared by the following:

$$e_1 : E_1 \lhd_t e_2 : E_2,$$
$$e_1 : E_1 \prec_t e_2 : E_2.$$

The former declares that an event e_2 temporally includes an event e_1, and the latter declares that an event e_1 temporally precedes an event e_2. For example, $e_1 : boil \lhd_t e_2 : evaporate$ expresses 'evaporating for the time boiling.' More precisely, it means that an event e_2 of *evaporating* temporally includes an event e_1 of *boiling*.

We incorporate these features of events into a logical language and present a query answering system in a reasoning mechanism for event assertions. In the formalization of our logic, order-sorted logic and second-order predicate logic provide us with syntactic notions that are relevant to event representation. Sorted signatures in order-sorted logic formally clarify the sort declarations of symbols denoting events; predicate constants and predicate variables in second-order predicate logic are suitable for representing constants and variables over events.

3 Preliminaries

First, we define first-order predicate logic with sort-hierarchy, called order-sorted logic (Schmidt-Schauss 1989, Socher-Ambrosius & Johann 1996, Kaneiwa & Mizoguchi 2004).

Definition 1 (Sorted language)
A sorted first-order language contains the set $\mathcal{S}$ of individual sort symbols $s_1, s_2, \ldots$, the set $\mathcal{F}$ of function symbols $f_1, f_2, \ldots$, the set $\mathcal{P}$ of predicate symbols $p_1, p_2, \ldots$, the connectives $\wedge, \vee, \rightarrow, \neg$ and the quantifiers $\exists, \forall$.

In particular, $\mathcal{F}_n$ denotes the set of n-ary function symbols, and $\mathcal{P}_n$ denotes the set of n-ary predicate symbols. $\mathcal{V}_s$ is the set of variables $x : s, y : s, \ldots$ of sort s, and $\mathcal{V} = \bigcup_{s \in \mathcal{S}} \mathcal{V}_s$ is the set of variables of all sorts.

Definition 2 (Sorted signature)
A signature (called a sorted signature) of a sorted first-order language is a tuple $\Sigma = (\mathcal{S}, \mathcal{F}, \mathcal{P}, D)$ where

1. *$(\mathcal{S}, \sqsubseteq_S)$ is a partially ordered set of sorts including the least sort $\bot_S$ and the greatest sort $\top_S$.*
2. *If $f \in \mathcal{F}_n$, then there is a function declaration $f : \langle s_1, \ldots, s_n, s \rangle \in D$.*
3. *If $p \in \mathcal{P}_n$, then there is a predicate declaration $p : \langle s_1, \ldots, s_n \rangle \in D$.*

We denote by $s_i \sqcap s_j$ and $s_i \sqcup s_j$ the greatest lower bound for s_i and s_j and the least upper bound for s_i and s_j respectively.

Definition 3 (Sorted terms)
Let $\Sigma = (\mathcal{S}, \mathcal{F}, \mathcal{P}, D)$ be a sorted signature. The set of sorted terms is defined inductively by the following rules:

1. *Every sorted variable $x : s$ is a sorted term of sort s.*

2. *If* $f \in \mathcal{F}_n$ *with* $\langle s_1, \ldots, s_n, s\rangle \in D$ *and* $r_1 : s_1, \ldots, r_n : s_n$ *are sorted terms of sorts* $s_1, \ldots, s_n$*, then* $f(r_1 : s_1, \ldots, r_n : s_n) : s$ *is a sorted term of sort* s*.*

Definition 4 (Sorted formulas)
Let $\Sigma = (\mathcal{S}, \mathcal{F}, \mathcal{P}, D)$ *be a sorted signature. The set of sorted formulas is defined inductively by the following rules:*

1. *If* $p \in \mathcal{P}_n$ *with* $p : \langle s_1, \ldots, s_n\rangle \in D$ *and* $r_1 : s_1, \ldots, r_n : s_n$ *are sorted terms of sorts* $s_1, \ldots, s_n$*, then* $p(r_1 : s_1, \ldots, r_n : s_n)$ *is a sorted formula.*
2. *If* A *and* B *are sorted formulas, then* $\neg A$*,* $A \wedge B$*,* $A \vee B$*,* $A \rightarrow B$*,* $(\forall x{:}s)A$ *and* $(\exists x{:}s)A$ *are sorted formulas.*

Definition 5 (Σ-structure)
A sorted structure (called a Σ*-structure) for a sorted signature* Σ *is a pair* $M = (U_{ind}, I)$ *of a non-empty set* U_{ind} *(the universe of* M*) and an interpretation function* I *for* $\mathcal{S} \cup \mathcal{F} \cup \mathcal{P}$ *such that*

1. $I(s) \subseteq U_{ind}$ *where* $I(\bot_S) = \emptyset$ *and* $I(\top_S) = U_{ind}$*,*
2. $I(s_i) \subseteq I(s_j)$ *for* $s_i \sqsubseteq_S s_j \in \Sigma$*,*
3. $I(f) : I(s_1) \times \cdots \times I(s_n) \rightarrow I(s)$ *where* $f : \langle s_1, \ldots s_n, s\rangle \in D$*.*

4 An order-sorted event logic

Based on the specification in Section 1, we define the syntax and semantics of an event logic with quantifiers over events, event sorts and composition and disjointness of events. A sorted event language contains the set $\mathcal{E}$ of event constant symbols $e_1, e_2, \ldots$, the set $\mathcal{ES}$ of event sort symbols $E_1, E_2, \ldots$, the binary event predicate symbols $\lhd_t, \prec_t, \approx_t, \approx_l$, the binary event function symbol $\circ$ (event composition) and the symbols of a sorted first-order language (the set $\mathcal{S}$ of individual sort symbols $s_1, s_2, \ldots$, the set $\mathcal{F}$ of function symbols, the connectives $\wedge, \vee, \rightarrow, \neg$ and the quantifiers $\exists, \forall$). $\mathcal{EV}_{\langle s_1, \ldots, s_n\rangle}$ is the set of event variables $X, Y, \ldots$ of predicate declaration $\langle s_1, \ldots, s_n\rangle$, and $\mathcal{EV}$ is the set of event variables of all predicate declarations.

Definition 6 (Event sort-hierarchy)
An event sort-hierarchy with disjointness is a tuple $(\mathcal{ES}, \sqsubseteq_E, ||_E)$ *where* $\mathcal{ES}$ *is the set of event sorts (including* $\bot_E$ *and* $\top_E$*),* $\sqsubseteq_E$ *(event subsort relation) is a partial order on* $\mathcal{ES}$ *and* $||_E$ *(event disjoint relation) is a binary relation on* $\mathcal{ES}$*.*

The event disjoint relation $E_i||_E E_j$ implies that events in the event sorts E_i, E_j do not take place simultaneously.

Definition 7 (Event signature) *A sorted signature (called an event signature) of a sorted event language is a tuple* $\Sigma_{ev} = (\mathcal{S}, \mathcal{F}, \mathcal{E}, \mathcal{ES}, \mathcal{T}, \mathcal{L}, D^+)$ *such that*

1. $\Sigma = (\mathcal{S}, \mathcal{F}, \mathcal{E}, D)$ *is a sorted signature with* $D \subseteq D^+$*, where* $(\mathcal{S}, \sqsubseteq_S)$ *is a partially ordered set of sorts and* D *is a set of function declarations* $f : \langle s_1, \ldots, s_n, s\rangle$ *and predicate declarations* $e : \langle s_1, \ldots, s_n\rangle$*.*
2. $\mathcal{T}$ *is the set of times (denoted* $t_1, t_2, \ldots$*), and* $\mathcal{L}$ *is the set of locations (denoted* $l_1, l_2, \ldots$*).*
3. $\prec_t$ *(time precedence) is a linear order on* $\mathcal{T}$*, and* $\lhd_t$ *(time inclusion) is a partial order on* $\mathcal{T}$*.*
4. *If* $e \in \mathcal{E}$*, then there are* $|e| = (t, l)$*,* $e : E \in D^+$ *and* $e{:}\langle s_1, \ldots, s_n\rangle \in D$*.*

In the event signature, each event constant is declared as a constant of an event sort E, and at the same time it is declared as an n-ary predicate of $\langle s_1, \ldots, s_n\rangle$. Thus, the event constant is restricted by two different kinds of sort declarations:

(i) $e{:}E$ as in a set of events (an event sort)

(ii) $e{:}\langle s_1, \ldots, s_n\rangle$ as in the set of n-ary predicates of $\langle s_1, \ldots, s_n\rangle$

Moreover, $|e|$ denotes the time and location at which the event e occurs, and $|e|_t = t_k$ and $|e|_l = l_k$ if $|e| = (t_k, l_k)$ in Σ_{ev}. In the following, event expressions in a sorted event language: *event term* and *event formula* are introduced. The function '$|\ |$' is expanded to event variables and event compositions in the definition of event terms.

Definition 8 (Event terms)
Let $\Sigma_{ev} = (\mathcal{S}, \mathcal{F}, \mathcal{E}, \mathcal{ES}, \mathcal{T}, \mathcal{L}, D^+)$ *be an event signature. The set of event terms* $t^e : E$ *is defined inductively by the following rules:*

1. *Every event constant* $e : E$ *with* $e : E \in D^+$ *and* $e{:}\langle s_1, \ldots, s_n\rangle \in D$ *is an event term of* $\langle s_1, \ldots, s_n\rangle$*.*
2. *Every event variable* $X{:}E$ *with* $X \in \mathcal{EV}_{\langle s_1, \ldots, s_n\rangle}$ *is an event term of* $\langle s_1, \ldots, s_n\rangle$*, and* $|X| = (x, y)$*.*
3. *If* $t_1^e : E_1$ *and* $t_2^e : E_2$ *are event terms of* $\langle s_1, \ldots, s_n\rangle$ *and* $\langle s_1, \ldots, s_m\rangle$ $(n \leq m)$ *and* $|t_1^e|$ *and* $|t_2^e|$ *are unifiable, then* $t_1^e \circ t_2^e : E_1 \sqcap E_2$ *is an event term of* $\langle s_1, \ldots, s_n\rangle$*, and* $|t_1^e \circ t_2^e| = mgu(|t_1^e|, |t_2^e|)$*.*

In Statement 2 of Definition 8, the pair (x, y) indicates a pair of variables of times and locations such that x and y can be substituted with any time t_k and location l_k.

The event formulas are constructed by two kinds of atomic formulas: *n-ary predicate over individuals* and *relation of event terms*, and by logical connectives, first-order quantifiers and second-order quantifiers.

Definition 9 (Event formulas)
Let $\Sigma_{ev} = (\mathcal{S}, \mathcal{F}, \mathcal{E}, \mathcal{ES}, \mathcal{T}, \mathcal{L}, D^+)$ *be an event signature. The set of event formulas is defined inductively by the following rules:*

1. *If* $t^e : E$ *is an event term of* $\langle s_1, \ldots, s_m\rangle$ *and* $r_1 : s_1, \ldots, r_n : s_n$ $(n \leq m)$ *are sorted terms of sorts* $s_1, \ldots, s_n$*, then* $t^e : E(r_1 : s_1, \ldots, r_n : s_n)$ *is an event formula.*
2. *If* $t_1^e : E_1$ *and* $t_2^e : E_2$ *are event terms, then* $t_1^e : E_1 \prec_t t_2^e : E_2$*,* $t_1^e : E_1 \lhd_t t_2^e : E_2$*,* $t_1^e : E_1 \approx_t t_2^e : E_2$ *and* $t_1^e : E_1 \approx_l t_2^e : E_2$ *are event formulas.*
3. *If* A *and* B *are event formulas, then* $\neg A$*,* $A \wedge B$*,* $A \vee B$*,* $A \rightarrow B$*,* $(\forall x{:}s)A$ *and* $(\exists x{:}s)A$ *are event formulas.*
4. *If* A *is an event formula, then* $(\forall X : E)A$ *and* $(\exists X{:}E)A$ *are event formulas.*

We use the abbreviation $t_1^e : E_1 \approx t_2^e : E_2$ for $t_1^e : E_1 \approx_t t_2^e : E_2$ and $t_1^e : E_1 \approx_l t_2^e : E_2$. The semantics of our order-sorted event logic is defined by sorted structures restricted on the conditions of an event signature Σ_{ev}. Let $\vec{d} = (d_1, \ldots, d_n, t, l)$, each d_i of which is a member of the set U_{ind} of individuals. We define $sb(\vec{d}) = \{(d_1, \ldots, d_i, t, l) \mid 1 \leq i \leq n\}$ in order to

Let A, B be event formulas.

$$\frac{A \wedge B}{A; B}\ (\wedge 1) \qquad \frac{\neg(A \wedge B)}{\neg A \mid \neg B}\ (\wedge 2) \qquad \frac{A \vee B}{A \mid B}\ (\vee 1) \qquad \frac{\neg(A \vee B)}{\neg A; \neg B}\ (\vee 2)$$

$$\frac{A \to B}{\neg A \mid B}\ (\to 1) \qquad \frac{\neg(A \to B)}{A; \neg B}\ (\to 2) \qquad \frac{\neg\neg A}{A}\ (\neg\neg)$$

Let A be an event formula. If $s' \sqsubseteq_s s \in \Sigma$, then

$$\frac{(\forall x{:}s)A[x{:}s]}{A[r{:}s']}\ (\forall 1) \qquad \frac{\neg(\forall x{:}s)A[x{:}s]}{\neg A[c{:}s']}\ (\forall 2) \qquad \frac{(\exists x{:}s)A[x{:}s]}{A[c{:}s']}\ (\exists 1) \qquad \frac{\neg(\exists x{:}s)A[x{:}s]}{\neg A[r{:}s']}\ (\exists 2)$$

where $r{:}s'$ is any sorted term of sort s' and $c{:}s'$ is a new constant of sort s'.

Figure 1: Tableau rules in order-sorted logic

give semantics that corresponds to eliminating arguments, and define $\mathit{arg\text{-}tim}(sb(\vec{d})) = \{(d_1, \ldots, d_i, t) \mid (d_1, \ldots, d_i, t, l) \in sb(\vec{d})\}$ without location l.

Definition 10 (Σ_{ev}-structure)
A sorted structure (called a Σ_{ev}-structure) for an event signature Σ_{ev} is a tuple $M^+ = (U_{ind}, U_t, U_l, I^+)$ of non-empty sets U_{ind}, U_t and U_l with $U_{ind} \cap U_t \cap U_l = \emptyset$ and an interpretation function I^+ for $\mathcal{S} \cup \mathcal{F} \cup \mathcal{E} \cup \mathcal{ES} \cup \mathcal{T} \cup \mathcal{L} \cup \{\prec_t, \lhd_t\}$ such that

1. *$M = (U_{ind}, I)$ with $I \subseteq I^+$ is a Σ-structure,*
2. *$(U_t, I^+(\prec_t)) = (\mathcal{T}, \prec_t)$ and $(U_l, I^+(\lhd_t)) = (\mathcal{L}, \lhd_t)$,*
3. *$I^+(\circ)(sb(\vec{d_1}), sb(\vec{d_2})) = sb(\vec{d_1}) \cap sb(\vec{d_2})$,*
4. *$I^+(E) \subseteq \{sb(\vec{d}) \mid \vec{d} \in \bigcup_{\{s_1,\ldots,s_n\} \subseteq \mathcal{S}} I^+(s_1) \times \cdots \times I^+(s_n) \times U_t \times U_l\}$,*
5. *$I^+(E_i) \subseteq I^+(E_j)$ for $E_i \sqsubseteq_E E_j$ in Σ_{ev},*
6. *$I^+(\bot_E) = \emptyset$, and $I^+(E_i \sqcap E_j) = (I^+(E_i) \cap I^+(E_j)) \cup \{I^+(\circ)(sb(\vec{d_i}), sb(\vec{d_j})) \mid sb(\vec{d_i}) \in I^+(E_i),\ sb(\vec{d_j}) \in I^+(E_j)\}$,*
7. *$\mathit{arg\text{-}tim}(sb(\vec{d_i})) \cap \mathit{arg\text{-}tim}(sb(\vec{d_j})) = \emptyset$ for $sb(\vec{d_i}) \in I^+(E_i)$ and $sb(\vec{d_j}) \in I^+(E_j)$ where $E_i \|_E E_j$ in Σ_{ev},*
8. *$I^+(e) \in I^+(E)$ where $e{:}E \in D^+$,*
9. *$I^+(e) = sb(\vec{d})$ with $\vec{d} \in I^+(s_1) \times \cdots \times I^+(s_n) \times \{I^+(t)\} \times \{I^+(l)\}$ where $|e| = (t, l)$ and $e : \langle s_1, \ldots, s_n \rangle \in D$.*

A variable assignment α over individuals is a mapping from sorted variables to individuals (in U_{ind}) where $\alpha(x{:}s) \in I^+(s)$. Let $I^+(E_{\langle s_1,\ldots,s_n\rangle}) = I^+(E) \cap \{sb(\vec{d}) \mid \vec{d} \in \bigcup_{1 \le i \le n} I^+(s_1) \times \cdots \times I^+(s_i) \times U_t \times U_l\}$. A variable assignment Ω over events is a mapping from event variables to subsets of $\bigcup_{i \in \mathbb{N}} (U_{ind}^i \times U_t \times U_l)$ where $\Omega(X{:}E) \in I^+(E_{\langle s_1,\ldots,s_n\rangle})$ for $X \in \mathcal{EV}_{\langle s_1,\ldots,s_n\rangle}$.

Definition 11 (Σ_{ev}-interpretation)
A Σ_{ev}-interpretation is a tuple $\mathcal{I} = (M^+, \alpha, \Omega)$ where M^+ is a Σ_{ev}-structure, α is a variable assignment over individuals and Ω is a variable assignment over events. The denotation $[\![\ ]\!]_{\alpha,\Omega}$ for sorted terms and event terms is defined by:

1. *$[\![x{:}s]\!]_{\alpha,\Omega} = \alpha(x{:}s)$,*
2. *$[\![f(r_1 : s_1, \ldots, r_n : s_n)]\!]_{\alpha,\Omega} = I^+(f)([\![r_1 : s_1]\!]_{\alpha,\Omega}, \ldots, [\![r_n{:}s_n]\!]_{\alpha,\Omega})$,*
3. *$[\![e{:}E]\!]_{\alpha,\Omega} = I^+(e)$,*
4. *$[\![X{:}E]\!]_{\alpha,\Omega} = \Omega(X{:}E)$,*
5. *$[\![t^e{:}\bot_E]\!]_{\alpha,\Omega} = \emptyset$,*
6. *$[\![t_1^e \circ t_2^e : E_1 \sqcap E_2]\!]_{\alpha,\Omega} = I^+(\circ)([\![t_1^e : E_1]\!]_{\alpha,\Omega}, [\![t_2^e : E_2]\!]_{\alpha,\Omega})$.*

We define $\mathit{arg}(sb(\vec{d})) = \{(d_1, \ldots, d_i) \mid (d_1, \ldots, d_i, t, l) \in sb(\vec{d})\}$ as without time t and location l, and define $\mathit{tim}(sb(\vec{d})) = t$ and $\mathit{loc}(sb(\vec{d})) = l$ if $\vec{d} = (d_1, \ldots, d_n, t, l)$.

Definition 12 (Satisfiability) *Let $\mathcal{I} = (M^+, \alpha, \Omega)$ be a Σ_{ev}-interpretation and F be an event formula. The satisfiability relation $\mathcal{I} \models F$ is defined by the following rules:*

1. *$\mathcal{I} \models t^e : E(r_1 : s_1, \ldots, r_n : s_n)$ iff $([\![r_1 : s_1]\!]_{\alpha,\Omega}, \ldots, [\![r_n{:}s_n]\!]_{\alpha,\Omega}) \in \mathit{arg}([\![t^e{:}E]\!]_{\alpha,\Omega})$.*
2. *$\mathcal{I} \models t_1^e : E_1 \prec_t t_2^e : E_2$ iff $(\mathit{tim}([\![t_1^e : E_1]\!]_{\alpha,\Omega}), \mathit{tim}([\![t_2^e{:}E_2]\!]_{\alpha,\Omega})) \in I^+(\prec_t)$.*
3. *$\mathcal{I} \models t_1^e{:}E_1 \lhd_t t_2^e{:}E_2$ iff $(\mathit{tim}([\![t_1^e{:}E_1]\!]_{\alpha,\Omega}), \mathit{tim}([\![t_2^e{:}E_2]\!]_{\alpha,\Omega})) \in I^+(\lhd_t)$.*
4. *$\mathcal{I} \models t_1^e : E_1 \approx_t t_2^e : E_2$ iff $\mathit{tim}([\![t_1^e : E_1]\!]_{\alpha,\Omega}) = \mathit{tim}([\![t_2^e{:}E_2]\!]_{\alpha,\Omega})$.*
5. *$\mathcal{I} \models t_1^e : E_1 \approx_l t_2^e : E_2$ iff $\mathit{loc}([\![t_1^e : E_1]\!]_{\alpha,\Omega}) = \mathit{loc}([\![t_2^e{:}E_2]\!]_{\alpha,\Omega})$.*
6. *$\mathcal{I} \models \neg A$ iff $\mathcal{I} \not\models A$.*
7. *$\mathcal{I} \models A \wedge B$ iff $\mathcal{I} \models A$ and $\mathcal{I} \models B$.*
8. *$\mathcal{I} \models A \vee B$ iff $\mathcal{I} \models A$ or $\mathcal{I} \models B$.*
9. *$\mathcal{I} \models A \to B$ iff $\mathcal{I} \models A$ implies $\mathcal{I} \models B$.*
10. *$\mathcal{I} \models (\forall x{:}s)A$ iff for all $d \in I^+(s)$, $\mathcal{I}[x{:}s/d] \models A$.*

Argument rules: Let $t^e : E$ be an event term of $\langle s_1, \ldots, s_n \rangle$ and let $1 < m \leq n$.

$$\frac{t^e : E(r_1 : s_1, \ldots, r_m : s_m)}{t^e : E(r_1 : s_1, \ldots, r_{m-1} : s_{m-1})} \ (a1) \qquad \frac{\neg t^e : E(r_1 : s_1, \ldots, r_{m-1} : s_{m-1})}{(\forall x : s_m) \neg t^e : E(r_1 : s_1, \ldots, r_{m-1} : s_{m-1}, x : s_m)} \ (a2)$$

where $x : s_m$ is a new variable.

Event composition rules: Let $t_1^e \circ t_2^e : E_1 \sqcap E_2$ be an event term of $\langle s_1, \ldots, s_m \rangle$ and $\vec{t}$ be a sequence $r_1 : s_1, \ldots, r_n : s_n$ of sorted terms $(n \leq m)$.

$$\frac{t_1^e \circ t_2^e : E_1 \sqcap E_2(\vec{t})}{t_1^e : E_1(\vec{t}); t_2^e : E_2(\vec{t})} \ (c1) \qquad \frac{\neg t_1^e \circ t_2^e : E_1 \sqcap E_2(\vec{t}) \quad t_1^e : E_1 \approx t_2^e : E_2}{\neg t_1^e : E_1(\vec{t}) \mid \neg t_2^e : E_2(\vec{t})} \ (c2)$$

$$\frac{t_1^e \circ t_2^e : E_1 \sqcap E_2(\vec{t}) \quad \neg(t_1^e : E_1 \approx t_2^e : E_2)}{t_1^e \circ t_2^e : \bot_E(\vec{t})} \ (c3)$$

Event disjointness rules: Let $t_1^e : E_1$, $t_2^e : E_2$ be event terms of $\langle s_1, \ldots, s_n \rangle$ and $\langle s_1, \ldots, s_m \rangle$ and $\vec{t}$ be a sequence $r_1 : s_1, \ldots, r_k : s_k$ of sorted terms $(k \leq m, n)$. If $E_1 ||_E E_2$ in Σ_{ev}, then

$$\frac{t_1^e : E_1(\vec{t}) \quad t_1^e : E_1 \approx_t t_2^e : E_2}{\neg t_2^e : E_2(\vec{t})} \ (d1) \qquad \frac{t_1^e : E_1(\vec{t}) \quad t_2^e : E_2(\vec{t}) \quad t_1^e : E_1 \approx_t t_2^e : E_2}{t_1^e \circ t_2^e : \bot_E(\vec{t})} \ (d2)$$

Event quantification rules: Let $X \in \mathcal{EV}_{\langle s_1, \ldots, s_m \rangle}$ and let A be an event formula. If $E' \sqsubseteq_E E \in \Sigma_{ev}$, then

$$\frac{(\forall X : E) A[X : E]}{A[t^e : E']} \ (\forall X1) \quad \frac{\neg(\forall X : E) A[X : E]}{\neg A[e : E']} \ (\forall X2) \quad \frac{(\exists X : E) A[X : E]}{A[e : E']} \ (\exists X1) \quad \frac{\neg(\exists X : E) A[X : E]}{\neg A[t^e : E']} \ (\exists X2)$$

where $t^e : E'$ is any event term of $\langle s_1, \ldots, s_n \rangle$ $(n \leq m)$ and $e : E'$ is a new event constant of $\langle s_1, \ldots, s_k \rangle$ $(k \leq m)$.

Figure 2: Tableau rules for event formulas

11. $\mathcal{I} \models (\exists x : s) A$ *iff for some* $d \in I^+(s)$, $\mathcal{I}[x : s/d] \models A$.
12. $\mathcal{I} \models (\forall X : E) A$ *iff for all* $sb(\vec{d}) \in I^+(E_{\langle s_1, \ldots, s_n \rangle})$, $\mathcal{I}\{X : E/sb(\vec{d})\} \models A$.
13. $\mathcal{I} \models (\exists X : E) A$ *iff for some* $sb(\vec{d}) \in I^+(E_{\langle s_1, \ldots, s_n \rangle})$, $\mathcal{I}\{X : E/sb(\vec{d})\} \models A$.

An event formula F is satisfiable if for some Σ_{ev}-interpretation $\mathcal{I}$, $\mathcal{I} \models F$ holds, and F is unsatisfiable otherwise. F is valid if for every Σ_{ev}-interpretation $\mathcal{I}$, $\mathcal{I} \models F$ holds.

5 Tableau calculus

We extend the order-sorted tableau calculus in Figure 1 to event formulas by adding the tableau rules in Figure 2. This extended calculus provides a refutation method for finding contradictions of a set of formulas.

The argument rules eliminate the final argument in a positive event formula and complement the sorted variable in a negative event formula. Using event composition rules, a compound event assertion derives its corresponding two event assertions that involve the same time and location and that are not disjoint. Through the disjointness of two event sorts, an event assertion concludes the negation of its disjoint assertion in the event disjointness rules. The event quantification rules are based on quantification rules of predicate variables, which substitute subsort event terms for event variables.

Let Γ be a set of event formulas and A be an event formula. In order to prove the validity of A in Γ, our tableau calculus is used to decide whether or not $\Gamma \cup \{\neg A\}$ is refutable. In the following examples, the answers to the queries A are obtained in a manner such that the answer is *yes* if $\Gamma \cup \{\neg A\}$ refutable; otherwise, the answer is *no*.

Signature 1
$e : \langle food \rangle, e : boil,$
$boil \sqsubseteq_E cook, roast \sqsubseteq_E cook,$
$chicken \sqsubseteq_S meat, beef \sqsubseteq_S meat,$
$meat \sqsubseteq_S food$

Fact 1
$\underline{e : boil(c : chicken)}_{(0)}$

Query 1a
?-$X : cook(y : meat)$

Answer 1a
yes,
$X : cook = e : boil, y : meat = c : chicken$

Query 1b
?-$X : roast(y : beef)$

Answer 1b
no

Derivation 1a
$\Gamma_0^0 = \{\neg(\exists X : cook)(\exists y : meat) X : cook(y : meat)\}$
$\cup \Gamma_{Fact1}$

$\Gamma_1^0 = \Gamma_0^0 \cup \{\neg(\exists y{:}meat)e{:}boil(y{:}meat)\}$ (by $\exists X2$)
$\Gamma_2^0 = \Gamma_1^0 \cup \{\underline{\neg e{:}boil(c{:}chicken)}_{(0)}\}$ (by $\exists 2$)

In Signature 1, an event constant e is declared as a predicate of $\langle food\rangle$ and as a constant of event sort *boil*. In the event sort-hierarchy, the event sorts *boil* and *roast* are subsorts of *cook*. Regarding Fact 1, 'chicken was boiled', the answer to Query 1a 'Was meat cooked?' is *yes*. This is because the refutation is successful in cases, by applying the tableau calculus to $\{e{:}boil(c{:}chicken)\} \cup \{\neg(\exists X{:}cook)(\exists y{:}meat)X{:}cook(y{:}meat)\}$, a contradiction $\{e{:}boil(c{:}chicken), \neg e : boil(c : chicken)\}$ appears in Γ_2^0 of Derivation 1a. Moreover, the answer to Query 1b 'Was beef roasted?' will be *no* since the refutation fails.

Further, Signature 2 defines event constants e_1, e_2 as predicates of $\langle person, food\rangle$ and $\langle person, car\rangle$, and constants of event sorts *eat* and *drive* as follows:

Signature 2
$e_1{:}\langle person, food\rangle, e_1{:}eat,$
$e_2{:}\langle person, car\rangle, e_2{:}drive,$
$|e_1| = |e_2|, man \sqsubseteq_S person,$
$drive \sqcap eat \sqsubseteq_E eat$, $drive \sqcap eat \sqsubseteq_E drive$

Fact 2
$\underline{e_1{:}eat(john{:}man, c_1{:}food)}_{(0)}$
$\underline{e_2{:}drive(john{:}man, c_2{:}car)}_{(1)}$

Query 2
?-$X{:}drive \sqcap eat(john{:}man)$

Answer 2
yes,
$X{:}drive \sqcap eat = e_1 \circ e_2{:}drive \sqcap eat$

Derivation 2
$\Gamma_0^0 = \{\neg(\exists X{:}drive \sqcap eat)$
$X{:}drive \sqcap eat(john{:}man)\} \cup \Gamma_{Fact2}$
$\Gamma_1^0 = \Gamma_0^0 \cup \{\neg e_1 \circ e_2{:}drive \sqcap eat(john{:}man)\}$ (by $\exists X2$)
$\Gamma_2^0 = \Gamma_1^0 \cup \{e_1{:}drive \approx e_2{:}eat\}$ (by Deri.1)
$\Gamma_3^0 = \Gamma_2^0 \cup \{\neg e_1{:}drive(john{:}man)\}$ (by $c2$)
$\Gamma_4^0 = \Gamma_3^0 \cup \{(\forall x{:}car)$
$\neg e_1{:}drive(john{:}man, x{:}car)\}$ (by $a2$)
$\Gamma_5^0 = \Gamma_4^0 \cup \{\underline{\neg e_1{:}drive(john{:}man, c_1{:}car)}_{(0)}\}$ (by $\forall 1$)
$\Gamma_3^1 = \Gamma_2^0 \cup \{\neg e_2{:}eat(john{:}man)\}$ (by $c2$)
$\Gamma_4^1 = \Gamma_3^1 \cup \{(\forall y{:}food)$
$\neg e_2{:}eat(john{:}man, y{:}food)\}$ (by $a2$)
$\Gamma_5^1 = \Gamma_4^1 \cup \{\underline{\neg e_2{:}eat(john{:}man, c_2{:}food)}_{(1)}\}$ (by $\forall 1$)

Fact 2 contains event assertions, such as 'John is eating food' and as 'John is driving a car,' whose events occurred simultaneously and in the same location (by $|e_1| = |e_2|$). From these assertions, the answer to Query 2 'Is John eating while driving?' is *yes*. This result is obtained from Derivation 2, where the refutation is successful, i.e., $\{e_1 : drive(john : man, c_1 : car), \neg e_1 : drive(john : man, c_1 : car)\}$ and $\{e_2 : eat(john : man, c_2 : food), \neg e_2 : eat(john : man, c_2 : food)\}$ are contradictions in Γ_5^0 and Γ_5^1, derived from non-deterministic rules.

6 Conclusion and future work

In this paper, we have presented a sorted event logic, that provides rich knowledge representation of various roles of events in terms of constants, sorts, predicates and variables, together with the inference mechanisms. By our logic, event sort-hierarchies are an extension of sort-hierarchies in order-sorted logic, and the expressions of event constants and event variables are based on predicate constants and predicate variables in second-order predicate logic. In semantics, sorted structures are rather complicatedly elaborated to capture the meaning of event assertions. For such an expressive event language, we have developed a tableau calculus that provides new inferences and contradictions pertaining to event formulas. The logical aspects of events presented in this paper can be regarded as a common specification of events (whereas event ontology philosophically distinguishes events in the classification of entities). Thus, we have shown an ontological framework to represent what events are in information systems.

Currently, reasoning and description concerning temporal information in event assertions still remain immature. As an approach to this, we are now to plan to study a combination of our event logic with temporal logic. For instance, temporal relations implicitly included in events lead to interesting temporal reasoning in event assertions such as upward, downward, rightward and leftward heredities.

Acknowledgment

This research has been partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (17700164).

References

Allen, J. F. & Ferguson, G. (1994), 'Actions and events in interval temporal logic', *Journal of Logic and Computation* **4**(5), 531–579.

Berardi, D., Cali, A., Calvanese, D. & Giacomo, G. D. (2005), 'Reasoning on UML class diagrams', *Artificial Intelligence* **168**(1-2), 70–118.

Davidson, D. (1980), The logical form of action sentences, *in* 'Essays on actions and events', Clarendon Press, Oxford, pp. 105–148.

Galton, A. & Augusto, J. C. (2002), Two approaches to event definition, *in* 'Proceedings of DEXA 2002, Lecture Notes in Computer Science', Vol. 2453, pp. 547–556.

Gatziu, S. & Dittrich, K. R. (1994), Events in an Active Object-Oriented Database System, *in* 'Proceedings of the 1st International Workshop on Rules in Database Systems', Workshops in Computing, Springer, pp. 23–29.

Kaneiwa, K. (2004), 'Order-sorted logic programming with predicate hierarchy', *Artificial Intelligence* **158**(2), 155–188.

Kaneiwa, K. & Mizoguchi, R. (2004), Ontological knowledge base reasoning with sort-hierarchy and rigidity, *in* 'Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR2004)', pp. 278–288.

Kaneiwa, K. & Mizoguchi, R. (2005), An order-sorted quantified modal logic for meta-ontology., *in* 'Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX2005)', LNCS 3702, Springer–Verlag, pp. 169–184.

Kaneiwa, K. & Satoh, K. (2006), Consistency checking algorithms for restricted UML class diagrams, *in* 'Proceedings of the Fourth International Symposium on Foundations of Information and Knowledge Systems (FoIKS2006)', Springer–Verlag. accepted.

Kaneiwa, K. & Tojo, S. (1999), Event, property, and hierarchy in order-sorted logic, *in* 'Proceedings of the 1999 Int. Conf. on Logic Programming', The MIT Press, pp. 94–108.

Kaneiwa, K. & Tojo, S. (2001), An order-sorted resolution with implicitly negative sorts, *in* 'Proceedings of the 2001 Int. Conf. on Logic Programming', Springer-Verlag, pp. 300–314. LNCS 2237.

Landman, F. (1991), *Structures for Semantics*, Kluwer.

Sadri, F. & Kowalski, R. A. (1995), Variants of the event calculus, *in* L. Sterling, ed., 'Proceedings of the 12th International Conference on Logic Programming', MIT Press, pp. 67–82.

Schmidt-Schauss, M. (1989), *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Springer-Verlag.

Socher-Ambrosius, R. & Johann, P. (1996), *Deduction Systems*, Springer-Verlag.

Sowa, J. F. (2000), *Knowledge Representation: logical, philosophical and computational foundations*, Brooks/Cole.

Tojo, S. & Kaneiwa, K. (2003), Toward a proper semantics for the logic of occurrence, *in* 'Proceedings of the Context '03 Workshop on Barwise and Situation Theory'.

van Benthem, J. (1983), *The logic of time: a model-theoretic investigation into the varieties of temporal ontology and temporal discourse*, Vol. 156 of *Synthese Library*, Reidel.