# Reference Data Abstraction and Causal Relation based on Algebraic Expressions

Susumu Yamasaki and Mariko Sasakura
*Department of Computer Science, Okayama University, Tsushima-Naka, Okayama, Japan*

Keywords: Reference Data, Algebraic Structure, 3-Valued Model Theory.

Abstract: This paper is related to algebraic aspects of referential relations in distributed systems, where the sites as states are assumed to contain pages, and each page as reference data involves links to others as well as its own contents. The links among pages are abstracted into causal relations in terms of algebraic expressions. As an algebra for the representation basis of causal relations, more abstract Heyting algebra (a bounded lattice with Heyting implication) is taken rather than the Boolean algebra with classical implication, where the meanings of negatives are different in the two algebras. A standard form may be obtained from any Heyting algebra expression, which may denote causal relations with Heyting negatives. If the evaluation domain is taken from the 3-valued, then the algebraic expressions are abstract enough to represent referential links of pages in a distributed system, where the link may be interpreted as active, inactive and unknown. There is a critical problem to be solved in such a framework as theoretical basis. The model theory is relevant to nonmonotonic function or reasoning in AI, with respect to the mapping associated with the causal relations, such that fixed point theory cannot be always routines. This paper presents a method to inductively construct models of algebraic expressions conditioned in accordance to reference data characters. Then we examine the traverse of states with models of algebraic expressions clustering at states, for metatheory regarding searching the reference data in a distributed system. With abstraction from state transitions, an algebraic structure is refined such that operational aspect of traversing may be well formulated.

## 1 INTRODUCTION

As in a distributed system, we virtually assume sites as states where data and contents are involved in, which are associated with abstract state machine. Regarding existing data and contents at a state, they contain static informations, however, references may have another aspect to be captured as dynamic in the sense of being linked with others and having search effects. It can be also regarded as relational, in the sense of linkage. As traverses from a state to another, the state transition behaviors (like actions) should be formalized abstractly for a whole distributed system. The traverses of states have been examined as state transitions from algebraic views. As backgrounds, we have seen actions in abstract state machine structures and traverses.

(a) The action is formulated as a key role in strategic reasoning of abstract state machine, as well as concretized actions as programs in dynamic logic, acting and sensing failures are discussed as advanced works (Spalazzi and Traverso, 2000).

(b) Actions are also captured in logical systems from the viewpoints of sequential process, as in the papers (Giordano et al., 2000; Hanks and McDermott, 1987).

(c) Procedural action is expressed by denotational approach in the book (Mosses, 1992). The procedural method is in accordance with operational implementation for programs to be executed. The actions may be abstracted, with functional programs (Bertolissi et al., 2006).

(d) As regards transitions, abstract state machine is discussed, in the paper (Reps et al., 2005). Regarding structure of streams possibly caused by abstract state transitions, there is the note (Rutten, 2001).

In reasoning with semantics for AI programs and data representations, logical approaches are often taken:
(1) Logics with knowledge (Reiter, 2001) are classical. Based on beliefs and intentions, modal operations have been applied to mental states (Dragoni et al.,

Yamasaki, S. and Sasakura, M.
Reference Data Abstraction and Causal Relation based on Algebraic Expressions.
DOI: 10.5220/0009825602070214
In *Proceedings of the 9th International Conference on Data Science, Technology and Applications (DATA 2020)*, pages 207-214
ISBN: 978-989-758-440-4

1985).
(2) From model theoretic views, the argumentation may be expressed by means of 3-valued logic to the semantics for defeasible reasonings to implement (Governatori et al., 2004).
(3) Mobile ambients (Cardelli and Gordon, 2000; Merro and Nardelli, 2005) have been formulated with environments to make communication reasonable.
(4) The papers (Dam and Gurov, 2002; Kozen, 1983) are classical enough to formulate the proof systems with fixed points and their approximations.
(5) Compared with default or defeasible logic in AI programming, defeasibility is beforehand assumed in the given rules, but the causal relation consisting of rules by Heyting algebra expressions is simpler without treating ambiguity of rules containing default negation.

Following such backgrounds of algebraic and logical perspectives, this paper is motivated to examine algebraic approach to the representations of reference data. Through simple structure of reference data, algebraic expressions are studied and captured as causal relations, whose model theories are required in Heyting algebra. As regards traverses through states, semiring is characterized for abstract state transition. The theoretical aspect of Heyting algebra expressions in this paper contains a specific property in postfix modal operator of modal mu-calculus (S. Yamasaki, in *C0MPLEXIS* 2020):

(a) Reference data in distributed systems are analyzed. A simple inductive structure of reference links is abstracted from the view of deepening algebraic aspects.
(b) Heyting algebra expressions are adopted for application to abstractly represent referential structures. The model theory of algebraic expressions is hard, since the mapping associated with expressions is nonmonotonic such that classical fixed point theory cannot be always adopted. Some prefixpoint may be inductively constructed as a model. This paper makes refinements in the constructions of models for given algebraic expressions. The model theory can be applied to classical logic, but it is discussed in non-classical framework over a 3-valued domain. In addition to non-classical discussions, we just mention a classical tool to get negation by failure: Qeuries for algebraic expressions are approximately realized, by negation as failure rule being sound with respect to models.
(c) State transitions are abstracted into semiring structure, caused by models of algebraic expressions. In accordance with finite state automata, star semiring may be given, on one hand (S. Yamasaki, in *COMPLEXIS* 2017). From the view on nondeterministic alternation of traverses, more complex semiring may be defined, on the other hand.

The paper is organized as follows. Section 2 is concerned with simple observation of structures in reference data of distributed systems. In Section 3, Heyting algebra expressions, and the standardization of expressions are summarized, such that model theory problems may be mentioned and solved. Section 4 presents an algebraic structure in terms of semiring, by constructing the models of algebraic expression for concatenation and alternation regarding state transitions. Concluding remarks and related topics are described in Section 5.

## 2 STRUCTURE FOR REFERENCE DATA

### 2.1 Reference Data

A distributed system is assumed to consist of sites, where:

(i) the site contains pages, and
(ii) the page denotes references which are linked to others.

With abstraction from the system containing data with references to others, assume an abstract and simple system, where

(a) the sites are denoted as states,
(b) the pages in a site are defined in each state, and
(c) the reference data is organized in each page by a recursive way, as illustrated below.

| Reference name | Contents |
|---|---|
| | Reference name$_1$ |
| | … |
| | … |
| | Reference name$_n$ |

Note that the content and recursive references are separated such that the page is viewed as following Frege-ontology. It is formally in Backus-Naur Form described:

$$\begin{aligned}
&Syst ::= null_{Syst} \mid s : P;Syst\\
&P ::= null_P \mid p;P\\
&p ::= r;\ Con;\ ref\\
&ref ::= null_{ref} \mid r;\ ref
\end{aligned}$$

where:

(a) $null_{Syst}$, $null_P$ and $null_{ref}$ are the empty strings on the domain of systems, pages and references, respectively.

(b) *Syst* is a system variable, and $s$ is a state variable.

(c) *Con* is a variable denoting a content.

(d) the semicolon ";" denotes a concatenation operation.

(e) $p$ and $r$ are page and reference variables, respectively, such that $P$ and $ref$ denote a sequence of pages and a sequence of references, respectively.

Rather than the content of each page, the relation among references may be compiled in a page, that is, the page involves a reference followed by a sequence of references.

We now pay attention to the case assumption of the page to be active, inactive or unknown, in terms of the page to be linked, not linked or unknown in a distributed system of sites. To a reference (variable) $r$, the 3-valued domain may be taken to make assignments:

| Reference activity | Values |
|---:|:---:|
| Active (linked) | 1 |
| Unknown | 1/2 |
| Inactive (not linked) | 0 |

A reference followed by a sequence of references (possibly the empty) may suggests a causal relation between the sequence (as a cause) and the reference (as an effect). The relation is modeled by some programing, as well.

*Related Programming*:
A "logic program" with respect to its Herbrand base may be regarded as containing the predicate $pr$ (with or without "$\sim$" as a procedure) preceded by the conjunction of:

$$pr_1, \ldots, pr_m \text{ (as a procedural body)}$$

for $pr_1, \ldots, pr_m$ (predicates or their negations). The program may be dealt with in 3-valued logic, where the negation is interpreted as default negation.

## 2.2 Causal Relation in Terms of Algebraic Expressions

Heyting algebra (HA) $(A, \bigvee, \bigwedge, \bot, \top)$ equipped with the partial order $\sqsubseteq$ and an implication $\Rightarrow$ is assumed as follows:

(i) $\bot$ and $\top$ are the least and the greatest elements of the algebra (set) $A$, respectively, with respect to the partial order $\sqsubseteq$.

(ii) the *join* $\bigvee$ and *meet* $\bigwedge$ are defined for any two elements of $A$.

(iii) as regards the implication $\Rightarrow$,

$$c \sqsubseteq (a \Rightarrow b) \text{ iff } a \bigwedge c \sqsubseteq b.$$

The element $a \Rightarrow \bot$ is denoted as "*not a*" (a negative) for $a \in A$, where $not\ \bot = \top$ and $not\ \top = \bot$. As is well known, we note some algebraic properties on the HA with parentheses of operation-priority representation:

$$((a \Rightarrow b) \bigwedge (b \Rightarrow c)) \sqsubseteq (a \Rightarrow c),$$
$$a \sqsubseteq not\ (not\ a),$$

as well as

$$(a \bigwedge (a \Rightarrow b)) \Rightarrow b$$

are always equivalent to the top element, such that

$$not\ (a \bigvee b) = not\ a \bigwedge not\ b.$$

Therefore the expression $a \Rightarrow b$ may be regarded as representing a causal relation between the cause denoted $a$ and the effect denoted $b$. The implication $\Rightarrow$ is more abstract than the classical (e.g. propositional) logic one.

The causal relation is now taken into consideration, for the denotation of reference data, by more general form of the HA expression. The expression $F$ (over the underlined set $A$ of the algebra) of the following form is regarded as a causal relation to abstract reference data:

$$\bigwedge_j (l_1^j \bigwedge \cdots \bigwedge l_{n_j}^j \Rightarrow l^j)$$

where $l_i^j$ denotes $a$ or $a \Rightarrow \bot$ (*not a*) for $a \in A$.

Assume in the following that (a) the implication is based on Heyting algebra, and (b) the evaluation of *not a* (with respect to the value of $a$ for $a \in A$) follows the rule:

| $a$ | *not a* |
|---:|---:|
| 1 | 0 |
| 1/2 | 0 |
| 0 | 1 |

where $0 \sqsubseteq 1/2 \sqsubseteq 1$.

*Transformation of Expressions*:
In an Heyting algebra $(A, \bigvee, \bigwedge, \bot, \top)$, any expression $Ex_1$ derives some expression $Ex_2$ of the form:

$$\bigwedge_j (l_1^j \bigwedge \cdots \bigwedge l_{n_j}^j \Rightarrow l^j),$$

where $l_i^j$ and $l^j$ are an expression $a$ or *not* $a$ (denoting $a \sqsubseteq \perp$), for $a \in A$, such that

$$Ex_2 \sqsubseteq Ex_1.$$

By the method in a language system (S. Yamasaki, in *COMPLEXIS* 2020), we may have got such a standardization to transform a given expression $Ex_1$ to $Ex_2$. If there is some model of $Ex_2$ in 3-valued domain, then it may be also the model of $Ex_1$. In this sense, the expression $Ex_2$ is worthwhile being obtained, as a standard form.

## 3 MODELS OF ALGEBRAIC EXPRESSIONS

With respect to a denotation of pages in a state (site) containing refernce data, the expression $F$ (over the underlined set $A$ of the algebra) of the form:

$$\bigwedge_j (l_1^j \wedge \ldots \wedge l_{n_j}^j \Rightarrow l^j)$$

is represented as a set of the rules

$$\{l_1^j \wedge \ldots \wedge l_{n_j}^j \Rightarrow l^j \mid j = 1, 2, \ldots\}.$$

It may be regarded as a set of causal relations of the form $l_1^j \wedge \ldots \wedge l_{n_j}^j \Rightarrow l^j$, with the HA implication $\Rightarrow$, where the outer meet is assumed in the evaluation of the set (the whole expression). The set of rules is also referred to by the same name $F$ in the following.

### 3.1 Conditioned Algebraic Expressions in 3-Valued Domain

To have a theory of HA expressions applicable to denotations of reference data, we here have restrictions on the set of rules:

(a) Given a set, the left hand of $\Rightarrow$ for $a$ or *not* $a$ (with its right hand) is unique, if the rule "$\ldots \Rightarrow a$" or "$\ldots \Rightarrow$ *not* $a$" exists.

(b) For each $a \in A$, there is no case that both the rules "$\ldots \Rightarrow a$" and "$\ldots \Rightarrow$ *not* $a$" are defined.

(c) The model of a given expression (a set) is considered in 3-valued domain.

We present *prefixed point as model* of the expression $F$, over the 3-valued domain $\{0, 1/2, 1\}$. With the set $A$ for a *conditioned* expression $F$, a mapping

$$\Psi_F : 2^A \times 2^A \to 2^A \times 2^A,$$
$$\Psi_F(I_1, J_1) = (I_2, J_2),$$

can be defined with *order* of componentwise subset inclusion.

*The Mapping* $\Psi_F$:

Note that the left hand part is unique for each right hand of the implication $\Rightarrow$. Because the conditioned form is assumed, in order to denote reference data structure.

Assume $(I_1, J_1)$ for a given set of rules, where $I_1$ is regarded as the set of elements assigned to 1, and $J_1$ is as the set of elements assigned to 0. For each $a \in A$, within the rules of $F$:

(1) In case that there is a rule (in the set)

$$b_1 \wedge \ldots \wedge b_n \wedge \textit{not } c_1 \wedge \ldots \wedge \textit{not } c_m \Rightarrow a:$$

if any $b_i$ is in $I_1$ $(1 \le i \le n)$, and any $c_j$ is in $J_1$ $(1 \le j \le m)$, then $a \in I_2$.

(2)(a) In case that there is no rule, whose right hand of the implication $\Rightarrow$ is $a$ (that is, $a$ may be only in the left hands of rules): $a \in J_2$.

(b) In case that there is a rule

$$b_1 \wedge \ldots \wedge b_n \wedge \textit{not } c_1 \wedge \ldots \wedge \textit{not } c_m \Rightarrow a:$$

if some $b_i$ is in $J_1$ $(1 \le i \le n)$, or some $c_j$ is not in $J_1$ $(1 \le j \le m)$, then $a \in J_2$.

(c) In case that there is a rule

$$d_1 \wedge \ldots \wedge d_l \wedge \textit{not } e_1 \wedge \ldots \wedge \textit{not } e_k \Rightarrow \textit{not } a:$$

if any $d_i$ is not in $J_1$ $(1 \le i \le l)$, and any $e_j$ is in $J_1$ $(1 \le j \le k)$, then $a \in J_2$.

If $\Psi_F(I, J) \subseteq_c (I, J)$ (with the componentwise subset inclusion $\subseteq_c$) and $I \cap J = \emptyset$, then $(I, J)$ can be a model of $F$, that is, $F$ is evaluated as 1.

Note: If $I \cap J = \emptyset$, then $I_0 \cap J_0 = \emptyset$ for $(I_0, J_0) = \Psi_F(I, J)$. It is because of the restriction of the expression $F$. That is, both $a$ and *not* $a$ are not definable. Since the mapping $\Psi_F$ is not monotonic, the method by (pre-)fixpoint of $\Psi_F$ is not always available as a modelling of the given expression $F$.

**Proposition 1.** *Assume a pair* $(I, J) \in 2^A \times 2^A$ *for a given expression (a set of rules) with the element set* $A$. *If* $\Psi_F(I, J) \subseteq_c (I, J)$, *then the pair* $(I, J)$ *is a model of* $F$.

*Proof.* Let $\Psi_F(I, J) = (I_0, J_0)$. Following the definition of the mapping $\Psi_F$, we make the exhaustive examination. For any $a$ occurring in $F$, there are three types of rules.
(i) In case of (1), if the left hand of the rule (where the left hand may be the empty) is evaluated as 1 by $(I, J)$, then the right hand $a \in I_0$ is in $I$, evaluated as 1.
(ii) In case of (2): (a) if no $a$ may occur in the right hand of a rule, $a \in J_0$ is evaluated as 0 for the pair $(I, J)$ to consistently be a model.
(b) if the left hand of the rule is evaluated as 0 (with case of (b)), then the right hand $a \in J_0 \subseteq J$ is in $J$,

evaluated as 0. (c) if the left hand of the rule (where the left hand may be the empty) is evaluated as not 0, then the right hand *not* $a$ ($a \in J_0$) is evaluated as 1. Because $a \in J$.

Thus all the rules are evaluated as 1, with respect to the relations between left and right hands of the implication. This concludes the proposition. □

In what follows, we suppose the set $A$ (for HA expressions) and the expression $F$ in a set of rules.

We have got a procedure with respect to construction of some model $(I,J)$, if $\Psi_F(I,J) \subseteq_c (I,J)$.

*Predicates of Success and Failure for Query*:
With respect to query $a$ to be an effect for the expression, the predicates of success $Suc_F(a)$ and failure $Fail_F(a)$ may be inductively defined for $a \in A$ and a given expression (a set of rules) $F$ as follows.

(1) If there is a rule

$$b_1 \bigwedge \ldots \bigwedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that $Suc_F(b_i)$ for any $1 \le i \le n$, and $Fail_F(c_j)$ for any $1 \le j \le m$, then $Suc_F(a)$.

(2)(a) If there is no rule, whose right hand of the implication $\Rightarrow$ is $a$, then $Fail_F(a)$.

(b) If there is a rule

$$b_1 \bigwedge \ldots \wedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that $Fail_F(b_i)$ for some $1 \le i \le n$, or not $Fail_F(c_j)$ for some $1 \le j \le m$, then $Fail_F(a)$.

(c) If there is a rule

$$d_1 \bigwedge \ldots \bigwedge d_l \bigwedge not\ e_1 \bigwedge \ldots \bigwedge not\ e_k \Rightarrow not\ a$$

such that not $Fail_F(d_i)$ for any $1 \le i \le l$, and $Fail_F(e_j)$ for any $1 \le j \le k$, then $Fail_F(a)$.

**Proposition 2.** . *Assume a pair $(I,J) \in 2^A \times 2^A$ for an expression $F$ over the set A, such that*

$$I = \{a \mid Suc_F(a)\} \text{ and } J = \{b \mid Fail_F(b)\}.$$

*Then $\Psi_F(I,J) \subseteq_c (I,J)$.*

*Proof.* Let $\Psi_F(I,J) = (I_0,J_0)$. (1) Assume that $a \in I_0$. Then there is a rule

$$b_1 \bigwedge \ldots \bigwedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that $b_i \in I$ for any $1 \le i \le n$, and $c_j \in J$ for any $1 \le j \le m$. By the assumed definitions of $I$ and $J$, $Suc_F(b_i)$ for any $1 \le i \le n$, and $Fail_F(c_j)$ for any $1 \le j \le m$. It follows that $Suc_F(a)$. That is, $a \in I$. Thus $I_0 \subseteq I$.
(2) When $a \in J_0$, then there are cases as follows.
(a) In case that there is no rule, whose right hand of the implication is $a$, $a \in J$.
(b) In case that there is a rule

$$b_1 \bigwedge \ldots \bigwedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that some $b_i$ is in $J$ $(1 \le i \le n)$, or some $c_j$ is not in $J$ $(1 \le j \le m)$: It follows that there is some $Fail_F(b_i)$ $(1 \le i \le n)$, or not $Fail_F(c_j)$ for some $1 \le j \le m$. Then $Fail_F(a)$, and $a \in J$.
(c) In case that there is a rule

$$d_1 \bigwedge \ldots \bigwedge d_l \bigwedge not\ e_1 \bigwedge \ldots \bigwedge not\ e_k \Rightarrow not\ a$$

such that any $d_i$ is not in $J$ $(1 \le i \le l)$, and any $e_j$ is in $J$ $(1 \le j \le k)$: By the definitions of $I$ and $J$, not $Fail_F(d_i)$ for any $1 \le i \le l$, and $Fail_F(e_j)$ for any $1 \le j \le k$. Therefore $Fail_F(a)$, and $a \in J$.

In the above cases, $a \in J$ on the assumption that $a \in J_0$. Therefore $J_0 \subseteq J$. This completes that

$$(I_0,J_0) \subseteq (I,J)$$

□

The significance of the above proposition is just soundness of the predicates of $Suc_F(a)$ and $Fail_F(b)$ with respect to a model $(I,J)$ of the given expression $F$, where the pair $(I,J)$ is really organized by the predicates.

## 3.2 Procedural Query

The predicate "not $Fail_F(a)$" is not so practical, where it is of use in the inductive definition of the predicate $Fail_F(a)$. It is primarily from nonmonotonicity of the mapping $\Psi_F$ associated with a given expression $F$ as causal relation. To make it more practical, we have simple predicates for queries concerning the expression $F$. The predicates $suc_F(a)$ and $fail_F(a)$ are definable, such that

(i) if $suc_F(a)$ then $Suc_F(a)$ and "not $Fail_F(a)$", and

(ii) if $fail_F(a)$ then $Fail_F(a)$.

Formally, the predicates are defined inductively in a similar manner.

(1) If there is a rule

$$b_1 \bigwedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that $suc_F(b_i)$ for any $1 \le i \le n$, and $fail_F(c_j)$ for any $1 \le j \le m$, then $suc_F(a)$.

(2)(a) If there is no rule, whose right hand of the implication $\Rightarrow$ is $a$, then $fail_F(a)$.

(b) If there is a rule

$$b_1 \bigwedge \ldots \bigwedge b_n \bigwedge not\ c_1 \bigwedge \ldots \bigwedge not\ c_m \Rightarrow a$$

such that $fail_F(b_i)$ for some $1 \le i \le m$, or $suc_F(c_j)$ for some $1 \le j \le m$, then $fail_F(a)$.

(c) If there is a rule

$$d_1 \bigwedge \ldots \bigwedge d_l \bigwedge not\ e_1 \bigwedge \ldots \bigwedge not\ e_k \Rightarrow not\ a$$

such that $suc_F(d_i)$ for any $1 \le i \le l$, and $fail_F(e_j)$ for any $1 \le j \le k$, then $fail_F(a)$.

The predicates are in accordance with reasoning of "negation as failure".

(a) If a query of $a$ succeeds, then a query of "*not a*" fails.

(b) If a query $a$ fails, then a query "*not a*" succeeds.

These predicates are sound with respect to a model $(I,J)$ constructed by the predicates $Suc_F(a)$ and $Fail_F(b)$, and related by the mapping $\Psi_F$.

**Proposition 3.** *Assume an expression $F$ over the set $A$ of algebraic elements. Let*

$$I = \{a \mid Suc_F(a)\} \text{ and } J = \{b \mid Fail_F(b)\}.$$

*We have that:*

(i) *if $suc_F(a)$ then $a \in I$.*

(ii) *if $fail_F(a)$ then $a \in J$.*

*Proof.* By the inductive definition and induction on proof,

(a) if $suc_F(a)$ then $Suc_F(a)$,

(b) if $fail_F(a)$ then $Fail_F(a)$, and

(c) $Suc_F(a)$ and $Fail_F(a)$ are exclusive.

Thus if $suc_F(a)$ then not $Fail_F(a)$. This may conclude the proposition. □

*Adjusting* (Procedure for Query):

A procedure may be constructed, in accordance with the definitions of the predicates $suc_F(a)$ and $fail_F(b)$.

By induction on the definitions of predicates $suc_F(a)$ and $fail_F(b)$, and on the following derivations $\langle a?\ suc\rangle$ and $\langle b?\ fail\rangle$, we can see that:

$$suc_F(a) \text{ iff } \langle a?\ suc\rangle, \text{ and}$$
$$fail_F(b) \text{ iff } \langle b?\ fail\rangle.$$

(1) With a given expression $F$, *query* of the *sequence* "$X$?" is assumed, where $X = y_1;\ldots;y_n$ $(n \geq 0)$ with $y_i$ being $a$ or *not* $a$ for $a \in A$ and with the concatenation operation ";" (which is treated as $\bigwedge$), where in case of "$n = 0$", $X$ is *null* (the empty query). A sequence query may be denoted as $y;X$? (with $y$ being $b$ or *not* $b$ for $b \in A$, and with $X$ a sequence query), or $Y;X$? with $Y$ and $X$ sequence queries.
(2) The notations $\langle X?\ suc\rangle$, and $\langle X?\ fail\rangle$ stand for the cases of the query $X$? to be a success, and a failure, respectively.

There are 2 routines of succeeding, and failing derivations for queries to be analyzed.

(i) $\langle null?\ suc\rangle$.

(ii) $\langle x;X?\ suc\rangle$, if $x = a$ (for $a \in A$) and there is $Y \Rightarrow x$ in $F$ such that $\langle Y;X?\ suc\rangle$.

(iii) (a) $\langle x;X?\ suc\rangle$, if $\langle a?\ fail\rangle$ for $x = not\ a$, and $\langle X?\ suc\rangle$.
(b) $\langle x;X?\ suc\rangle$, if $\langle a?\ fail\rangle$ for $x = not\ a$ where there is some $Y \Rightarrow x$ in $F$ with $\langle Y?\ suc\rangle$, and $\langle X?\ suc\rangle$.

(iv) $\langle x;X?\ fail\rangle$, if there is no part with $x = a \in A$ being the right hand of "$\Rightarrow$" in $F$, or $\langle X?\ fail\rangle$.

(v) $\langle x;X?\ fail\rangle$, if $x = a$ $(a \in A)$ such that $\langle Y?\ fail\rangle$ for some $Y$ (where $Y \Rightarrow x$ in $F$), or $\langle X?\ fail\rangle$.

(vi) (a) $\langle x;X?\ fail\rangle$, if $x = not\ a$ such that $\langle a?\ suc\rangle$, or $\langle X?\ fail\rangle$.
(b) $\langle x;X?\ fail\rangle$, if $x = a$ such that $\langle Y?\ suc\rangle$ for some $Y$ (where $Y \Rightarrow not\ a$ in $F$), or $\langle X?\ fail\rangle$.

## 4 SEMIRING STRUCTURE

As in the setting of a language system (S. Yamasaki, in *COMPLEXIS* 2020), when traversing the states (sites), pages from a state are concatenated to other pages of another state. This observation can be abstracted to some algebraic structure from expressions $F_1, F_2, \ldots$.

Given a logical or algebraic expression $F$ over the set $A$, we may have a pair

$$(I,J) \in 2^A \times 2^A,$$

which is assumed as a 3-valued model of $F$, and can be regarded as defining state changes (transitions).

With the set $A$, we can have denumerable expressions $F_1$, $F_2$, ..., causing state changes, which are in accordance with causal relations in a distributed system. (state transitions). Then the 3-valued models of expressions $F_1, F_2, \ldots$ may be assumed as the pairs

$$(I_1,J_1), (I_2,J_2), \ldots.$$

By means of the set concatenation "$\cdot$" (which gets the set of sequences obtained from taken elements of sets), we might have

$$\begin{aligned}(I_1,J_1) \bullet \ldots \bullet (I_n,J_n) \\ = (I_1 \cdot \ldots \cdot I_n \\ - \{w \mid \text{some element of } w \text{ of } I_1 \cdot \ldots \cdot I_n \\ \text{is in } J_1 \cup \ldots \cup J_n\}, \\ J_1 \cup \ldots \cup J_n),\end{aligned}$$

with multiplication "$\bullet$" to express the sequence formation.

In this paper, we newly have a semiring with respect to the view of human computer interaction. It is different from the star semiring constructed in the case (S. Yamasaki, in *COMPLXIS* 2017). Reflecting state transitions with models, alternations are denoted in terms of algebraic aspects. With alternation aspects, let $R_A$ ($R$, for short with the assumption of the set $A$) be the set of "direct sums" of the form $\Sigma_l(pSeq_l, nSet_l)$ with $l$ ranging indexes, where each pair $(pSeq_k, nSet_k)$ is supposedly *consistent*, with a model of an expression $F_k$ over $A$. From implementation views, the "sum" means *nondeterministic* selections as alternation so that it contains more complexity. Therefore human computer interaction may be of use, to control determinations. It is a compact representation that this section is to aim at, with respect to nondeterministic complexity. Keeping such complexity of what the sum contains, we newly have a formality of semiring structure as follows.

The operations $+$ (addition–alternation) and $\circ$ (multiplication–composition) on $R$ are defined:

(1)
$$\Sigma_i(pSeq_i, nSet_i) + \Sigma_j(pSeq_j, nSet_j) \\ = \Sigma_{k=i,j}(pSeq_k, nSet_k).$$

(2)
$$\begin{aligned}&\Sigma_i(pSeq_i, nSet_i) \circ \Sigma_j\ (pSeq_j, nSet_j)\\ &= \Sigma_{i,j}(pSeq_i \cdot pSeq_j\\ &\quad - \{uv \mid u \in pSeq_i, v \in pSeq_j,\\ &\qquad u \text{ is not consistent to } nSet_j\},\\ &\quad - \{uv \mid u \in pSeq_i, v \in pSeq_j,\\ &\qquad v \text{ is not consistent to } nSet_i\},\\ &\quad nSet_i \cup nSet_j),\end{aligned}$$

where

(i) the sequence $uv$ is constructed by concatenation of sequences $u$ and $v$,

(ii) by saying that $u$ and $v$ are not consistent to (the sets) $nSet_j$ and $nSet_i$, respectively, it means that $u$ and $v$ contain some element in $nSet_j$ and $nSet_i$, respectively, and

(iii) the operation $\cdot$ is the set concatenation consisting of concatenated sequences.

Note: The operation $\circ$ preserves "consistency" of each pair in a resultant direct sum.

Identities with respect to $+$ and $\circ$:

We can have identities with respect to the addition and multiplication in terms of alternation and composition, respectively, if we care the direct sum of the "form" $\Sigma_i(pSeq_i, nSet_i)$.

(i) $\Sigma_i(pSeq_i, nSet_i)$ is denoted $\emptyset_\Sigma$, if the direct sum is the empty. It is the identity with respect to $+$.

(ii) The empty sequence in $A^*$ is represented by $\varepsilon$. $(\{\varepsilon\}, \emptyset)$ is the identity with respect to $\circ$.

We finally have a semiring $R$ regarding consistent sequences caused by models of expressions, in terms of the following propositions:

**Proposition 4.** *The structure* $\langle R_A, +, \circ, \emptyset_\Sigma, (\{\varepsilon\}, \emptyset)\rangle$ *is a semiring.*

*Proof.* We can see the conditions of a semiring as follows.

(i) The operation $+$ is defined so that commutative and associative laws may obviously hold. With the identity $\emptyset_\Sigma$, $\langle R, +, \emptyset_\Sigma\rangle$ is a commutative monoid (a commutative semigroup with the identity).

(ii) The operation $\circ$ is associative, so that $\langle R, \circ, (\{\varepsilon\}, \emptyset)\rangle$ is a semigroup with the identity $(\{\varepsilon\}, \emptyset)$, that is, a monoid.

(iii) Left and right multiplications over addition are both distributive:

$$\begin{aligned}&\Sigma_i(pSeq_i, nSet_i)\\ &\circ(\Sigma_j(pSeq_j, nSet_j) + \Sigma_k(pSeq_k, nSet_k))\\ &= (\Sigma_i(pSeq_i, nSet_i) \circ \Sigma_j(pSeq_j, nSet_j))\\ &+(\Sigma_i(pSeq_i, nSet_i) \circ \Sigma_k(pSeq_k, nSet_k)).\end{aligned}$$

$$\begin{aligned}&(\Sigma_j(pSeq_j, nSet_j) + \Sigma_k(pSeq_k, nSet_k))\\ &\circ\Sigma_i(pSeq_i, nSet_i)\\ &= (\Sigma_j(pSeq_j, nSet_j) \circ \Sigma_i(pSeq_i, nSet_i))\\ &+(\Sigma_k(pSeq_k, nSet_k) \circ \Sigma_i(pSeq_i, nSet_i)).\end{aligned}$$

(iv) $\emptyset_\Sigma \circ \Sigma_i(pSeq_i, nSet_i) = \Sigma_i(pSeq_i, nSet_i) \circ \emptyset_\Sigma = \emptyset_\Sigma$. That is, annihilation holds for $\circ$, with the identity $\emptyset_\Sigma$ regarding $+$.

□

## 5 CONCLUSION

The primary contribution of this paper is to take Heyting algebra expressions as representations of causal relations with model theories to be newly established.

(a) The form of the expressions is restricted to a representation of reference data as static link, which is specific in the class of expressions as terms possibly occurring in postfix modal operator of modal mu-calculus, or as programs in a language system (S. Yamasaki, in *COMPLEXIS* 2020). In this sense, this paper is viewed as an application of the algebraic expressions as terms or programs in the previous papers.
(b) The causal relations are abstractions of the static links between data with references to each other in distributed systems, where practices are not concretized but abstracted with model theories. The model theories are based on 3-valued domain, where a nonmonotonic mapping is virtually associated with

expressions with Heyting implication and negatives. For a prefixpoint of the mapping, some inductive construction is presented. We then have models for a given expression conditioned to some representation forms. This model theory is relevant to those in logic programming (Yamasaki, 2006), but more general than, with respect to strict negation. As a software technology to analyze algebraic expression queries, negation as failure rule is applied as sound procedure.

As another result, a semiring structure is formally constructed with respect to state transitions virtually caused by dynamic traverses through reference links, which is related to automata theory (Droste et al., 2009) rather than context-free language aspects (Winter et al., 2013). The semiring involves nondeterminism by direct sum of objects derived from models, which require human interaction to selection of suitable objects. The abstract representation involves nondeterministic alternation of transitions from a state, to which human interaction may be implemented which transition to select.

As related works on logical frameworks possibly for AI, we should learn concepts and ideas as follows. They may be hints on advancements to be considered, as regards practical aspect of this paper:

(a) The paper (Beddor and Goldstein, 2018) presents the belief predicate with the credence function of agents, concerning epistemic contradictions. The contradictions of complexity may be avoided by grades of such a function.
(b) There is a paper (P. Kremer, 2018) presenting second-order propositional frameworks, with epistemic and intuitionistic logic. It may be relevant to the extension of this paper with HA expressions to more facility of complex expressiveness.
(c) With the second-order (quantified) propositions, the paper (Goranko and Kuusisto, 2018) involves dependence and independence concepts, which may control implementations of programs or queries if data base is designed with such concepts of representation complexity.
(d) "Distributed knowledge" is discussed (Naumov and Tao, 2019), with quantified variables of quantifies ranging over the set of agents. Concerning applications of the second-order predicates to knowledge, the paper (Kooi, 2016) contains the concept of knowing. Distributive knowledge processing is of more complexity even for the state constrained programs.
(e) For an extension of propositional modal logic without quantification, the paper (Fitting, 2002) introduces relations and terms with scoping mechanism by lambda abstraction. It is considered as presenting functional programming included in modal logic.

## REFERENCES

Beddor, B. and Goldstein, S. (2018). Believing epistemic contradictions. *Rev.Symb.Log.*, 11(1):87–114.

Bertolissi, C., Cirstea, H., and Kirchner, C. (2006). Expressing combinatory reduction systems derivations in the rewriting calculus. *Higher-Order.Symbolic.Comput.*, 19(4):345–376.

Cardelli, L. and Gordon, A. (2000). Mobile ambients. *Theoret.Comput.Sci.*, 240(1):177–213.

Dam, M. and Gurov, D. (2002). Mu-calculus with explicit points and approximations. *J.Log.Comput.*, 12(1):119–136.

Dragoni, A., Giorgini, P., and Serafini, L. (1985). Mental states recognition from communication. *J.Log.Program.*, 2(4):295–312.

Droste, M., Kuich, W., and Vogler, H. (2009). *Handbook of Weighted Automata*. Springer.

Fitting, M. (2002). Modal logics between propositional and first-order. *J.Log.Comput.*, 12(6):1017–1026.

Giordano, L., Martelli, A., and Schwind, C. (2000). Ramification and causality in a modal action logic. *J.Log.Comput.*, 10(5):625–662.

Goranko, V. and Kuusisto, A. (2018). Logics for propositional determinacy and independence. *Rev.Symb.Log.*, 11(3):470–506.

Governatori, G., Maher, M., Autoniou, G., and Billington, D. (2004). Argumentation semantics for defeasible logic. *J.Log.Comput.*, 14(5):675–702.

Hanks, S. and McDermott, D. (1987). Nonmonotonic logic and temporal projection. *Artifi.Intelli.*, 33(3):379–412.

Kooi, B. (2016). The ambiguity of knowability. *Rev.Symb.Log.*, 9(3):421–428.

Kozen, D. (1983). Results on the propositional mu-calculus. *Theoret.Comput.Sci.*, 27(3):333–354.

Merro, M. and Nardelli, F. (2005). Behavioral theory for mobile ambients. *J.ACM.*, 52(6):961–1023.

Mosses, P. (1992). *Action Semantics*. Cambridge University Press.

Naumov, P. and Tao, J. (2019). Everyone knows that some knows: Quantifiers over epistemic agents. *Rev.Symb.Log.*, 12(2):255–270.

Reiter, R. (2001). *Knowledge in Action*. MIT Press.

Reps, T., Schwoon, S., and Somesh, J. (2005). Weighted pushdown systems and their application to interprocedural data flow analysis. *Sci.Comput.Program.*, 58(1-2):206–263.

Rutten, J. (2001). *On Streams and Coinduction*. CWI.

Spalazzi, L. and Traverso, P. (2000). A dynamic logic for acting, sensing and planning. *J.Log.Comput.*, 10(6):787–821.

Winter, J., Marcello, B., Bonsangue, M., and Rutten, J. (2013). Coalgebraic characterizations of context-free languages. *Formal Methods in Computer Science*, 9(3):1–39.

Yamasaki, S. (2006). Logic programming with default, weak and strict negations. *Theory Prac.Log.Program.*, 6(6):737–749.