UDK 519.681.2, 519.681.3

Partial Orders in Transition Systems from Resolvable Conflict Event Structures

Gribovskaya N.S. (Institute of Informatics Systems SB RAS)

Virbitskaite I.B. (Institute of Informatics Systems SB RAS)

Event structures are a well-established model in concurrency theory. Two structurally different methods of associating transition system semantics to event-oriented models are distinguished in the literature. One of them is based on configurations (event sets), the other — on residuals (model fragments). In this paper, we deal with a highly expressive model of event structures — event structures for resolvable conflict (*RC*-structures) — and provide isomorphism results on these two types of transition systems constructed from *RC*-structures, in step and partial order semantics.

Keywords: event structures, resolvable conflict, transition systems, partial order, true concurrency

1. Introduction

In [24], Nielsen, Plotkin and Winskel have introduced the concept of prime event structures which are abstract representations of the behaviour of safe Petri nets. This event-oriented model describes a concurrent system by means of a set of events, representing action occurrences, and for every two events it is specified whether one of them is a predecessor for the other (*causality* presented in the form a partial order), whether they exclude each other (*conflict*), or whether they may happen independently (*concurrency*). The behaviour of an event structure is formalized by associating to it a family of configurations, these being sets of events that occur during runs of the represented system. A configuration can also be understood as a state of the represented system, namely the state reached after executing all events in the configuration. Since then, many studies have been conducted on possible relations among events, giving rise to a number of different definitions of event structures. Flow event structures [9] drop the requirement that causality should be a partial order. Bundle event structures [21] are able to represent OR-causality by allowing each event to be caused by a member of a bundle of events. Asymmetric event structures [5] introduce the notion of weak causality that can model asymmetric conflicts. Inhibitor event structures [4] are able to faithfully capture the dependencies among events which arise in the presence of read and inhibitor arcs. In [6] event

structures, where the causality relation may be circular, are investigated, and in [1] the notion of dynamic causality is considered. Configuration structures [13, 15] represent the behaviours of event-oriented models as the collections of their configurations. The culminating point in these studies is a highly expressive concurrency model called event structures for resolvable conflict (*RC*-structures) [15] corresponding to arbitrary nets without self-loops, under the collective token interpretation.

The association of transition systems with event-oriented models has proved to contribute to studying and solving various problems in the analysis and verification of concurrent systems. It is distinguished two methods of providing transition system semantics with event structures: a *configuration-based* and a *residual-based*. In the first (more 'behavioral') method (see [1, 2, 12–15, 18, 19, 26] among others), states of transition systems are understood as sets of configurations, and state transitions are built by starting with the empty configuration and enlarging configurations by already executed events. In the second (more 'structural') method [3, 9, 10, 19–22, 25], states are understood as event structures, and transitions are built by starting with the given event structure as an initial state and removing already executed (or conflicting) parts thereof in the course of an execution. In the literature, the two semantics have occasionally been used alongside each other (see [19] as an example), but their general relationship has not been studied too deeply. In a seminal paper, viz. [23], *bisimulations* between configuration-based and residual-based transition systems have been proved to exist for prime event structures [27]. This result has been extended in [7] to more complex event structure models — prime/bundle/dual event structures with asymmetric conflict. A crucial technical subtlety pertains to the *removal operator* that lies at the heart of residual semantics. Counterexamples illustrate that an isomorphism cannot be achieved with the various removal operators defined in [7, 23]. The paper [8] demonstrates that, nevertheless, the removal operators can be tightened in such a way that *isomorphisms*, rather than just bisimulations, between the two types of transition systems constructed from a single event structure can be obtained within a wide range of event-oriented models (namely, extended prime event structures, bundle/dual event structures, flow event structures, stable/general event structures, configuration structures), in different true concurrent semantics. In the paper [17], the relationships between transition system semantics for *RC*-structures presented in standard form have been established in step semantics, because only this semantics has been developed in [15].

The aim of this paper is threefold: to define partial order semantics within the configurations

of RC-structures, to find a maximal subclass of the models where partial orders work properly, and also to understand how transition systems based on configurations and residuals of RC-structures, presented not necessary in standard form, are related in the context of partial order multiset and step semantics.

This paper is structured as follows. In the next section, we first recall the definitions of the structure and behavior (configurations) of RC-structures, and then consider and study their special properties. In addition, there are defined partial order multiset and step semantics for RC-structures. In Section 3, we determine and investigate a removal operator for RC-structures in order to obtain their residuals. Section 4 contains the definitions of configuration- and residual-based transition systems and provides isomorphism results between them in the semantics under study, within RC-structures, presented not necessary in standard form. Section 5 concludes.

2. Event Structures for Resolvable Conflict

2.1. Basic Definitions of RC-structures and their Properties

In this subsection, we deal with event structures for resolvable conflict that were put forward in [14] to give semantics to general Petri nets. A resolvable conflict structure consists of a set of events and an enabling relation $\vdash$ between sets of events. The enabling $X \vdash Y$ with sets X and Y imposes restrictions on the occurrences of events in Y by requiring that for all events in Y to occur, their *causes* – the events in X – have to occur before. This allows for modeling the case when a and b cannot occur together until c occurs, i.e., initially a and b are in conflict until the occurrence of c *resolves this conflict*. In resolvable conflict structures, the enabling relation can also model conflicts: events from a set Y are *in irresolvable conflict* iff there is no enabling of the form $X \vdash Y$ for any set X of events. Further, an event can be *impossible* (i.e. non-executable in any system's run) if it has no enabling or has infinite causes or has impossible causes/predecessors. In [15, 17], strict interrelations of resolvable conflict structures have been established with a variety of event-oriented models known from the literature that are unable to model the phenomena of resolvable conflict.

Definition 1. *An* event structure for resolvable conflict (RC-structure) over L *is a tuple* $\mathcal{E} = (E, \vdash, L, l)$*, where* E *is a set of events;* $\vdash \subseteq \mathcal{P}(E) \times \mathcal{P}(E)$ *is the* enabling relation*;* L *is a set of labels;* $l : E \to L$ *is a labeling function.*

Introduce auxiliary notions and notations. For an *RC*-structure $\mathcal{E}$ over L, a subset $X \subseteq E$, and events $e, d \in E$, define:

- $Con(X) \iff \forall \widehat{X} \subseteq X \exists Z \subseteq E\colon Z \vdash \widehat{X}$ (*consistency*);
- $e \sharp d \iff \neg Con(\{d, e\})$ (*conflict*);
- $e \prec d \iff e \in X$ for all $X \subseteq E$ such that $X \vdash \{d\}$ (*direct causality*);
- X is *left-closed* iff X is finite, and for all $\widetilde{X} \subseteq X$ there exists a set $\widehat{X} \subseteq X$ such that $\widehat{X} \vdash \widetilde{X}$. The set of the left-closed sets of $\mathcal{E}$ is denoted as $LC(\mathcal{E})$. Clearly, we have $Con(X)$ for all $X \in LC(\mathcal{E})$;
- X is a *configuration of* $\mathcal{E}$ iff X can be represented as an ordered set $\{e_1, \ldots, e_n\}$ $(n \geq 0)$ such that for all $i \leq n$ and for all $Y \subseteq \{e_1, \ldots, e_i\}$, there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \vdash Y$. Let $Conf(\mathcal{E})$ be the set of configurations of $\mathcal{E}$. Clearly, any configuration X is a left-closed set but not conversely. An event $e \in E$ is called *impossible* (non-executable) in $\mathcal{E}$ if it does not occur in any $X \in Conf(\mathcal{E})$;
- for $X' \subseteq X \in Conf(\mathcal{E})$ and $e, d \in X$, $e \prec_{X'} d \iff e \in Y$ for all $Y \subseteq X'$ such that $Y \vdash \{d\}$ (*direct causality within* X'). Let $\preceq_{X'}$ be the reflexive and transitive closure of $\prec_{X'}$. In case when specifying $\mathcal{E}$ is important for the context, we write $\preceq_X^{\mathcal{E}}$. To save space in the graphical representation, causality between events a and b within a configuration is indicated by the notation $a; b$, the absence of causality (independence) — through $a \| b$.
- for $X, X' \in Conf(\mathcal{E})$, $X \to X'$ iff $X = \{e_1, \ldots, e_m\}$ $(m \geq 0)$, $X' = \{e_1, \ldots, e_m, \ldots, e_n\}$ $(n \geq 0)$, and $m \leq n$.

Lemma 1. *Given an RC-structure* $\mathcal{E} = (E, \vdash, L, l)$ *and a configuration* $X \in Conf(\mathcal{E})$, $\preceq_X$ *is a partial order.*

Proof. Suppose that $\mathcal{E} = (E, \vdash, L, l)$ is an *RC*-structure and $X \in Conf(\mathcal{E})$. As $\preceq_X$ is the transitive and reflexive closure of $\prec_X$, it is sufficient to show that $\preceq_X$ is antisymmetric. Assume $a \preceq_X b$ and $b \preceq_X a$. This means that in X there exist events $e_1, \ldots, e_k$ $(k \geq 1)$ and events $d_1, \ldots, d_l$ $(l \geq 1)$ such that $a = e_1 \prec_X e_2 \ldots e_{k-1} \prec_X e_k = b$ and $b = d_1 \prec_X d_2 \ldots d_{l-1} \prec_X d_l = a$. Since $X \in Conf(\mathcal{E})$, we have an ordered set $X = \{x_1, \ldots, x_m\}$ such that for all $i \leq m$ and all $Y \subseteq \{x_1, \ldots, x_i\}$ there is $Z \subseteq \{x_1, \ldots, x_{i-1}\}$ such that $Z \vdash Y$. W.l.o.g. assume $a = x_p$ and $b = x_q$ $(1 \leq p, q \leq m)$. If $p = q$, the result is obtained. Check the case when $p \neq q$. Consider the sequence $a = e_1 \prec_X e_2 \ldots e_{k-1} \prec_X e_k = b$. As $b = x_q$, we can find $C^b \subseteq \{x_1, \ldots, x_{q-1}\}$ such that $C^b \vdash \{b\}$. By the definition of $\prec_X$, we obtain $e_{k-1} \in C^b \subseteq \{x_1, \ldots, x_{q-1}\}$. Repeating the above reasoning, we come to the conclusion $e_1 = a \in \{x_1, \ldots, x_{q-1}\}$. Hence, $p < q$. Next,

consider the sequence $b = d_1 \prec_X d_2 \ldots d_{l-1} \prec_X d_l = a$. Applying the reasoning analogous to the above, we obtain the contradiction $q < p$.

Thus, $p = q$, i.e. $a = b$. □

The next definition gives structural properties of RC-structures which, in suitable combinations, determine subclasses of the models.

Definition 2. *An RC-structure $\mathcal{E} = (E, \vdash, L, l)$ is:*

- rooted *iff* $(\emptyset, \emptyset) \in \vdash$;
- pure *iff* $X \vdash Y \Rightarrow X \cap Y = \emptyset$;
- singular *iff* $X \vdash Y \Rightarrow X = \emptyset \vee \mid Y \mid = 1$;
- locally conjunctive *iff* $X_i \vdash Y (i \in I \neq \emptyset) \wedge Con(\bigcup_{i \in I} X_i \cup Y) \Rightarrow \bigcap_{i \in I} X_i \vdash Y$;
- in standard form *iff* $\vdash = \{(A, B) \mid A \cap B = \emptyset,\ A \cup B \in LC(\mathcal{E})\}$.

Some of the definitions above lead to the following

Observation. Given an RC-structure $\mathcal{E} = (E, \vdash, L, l)$,

(i) $\emptyset \in LC(\mathcal{E})$ iff $\mathcal{E}$ is rooted;

(ii) $Conf(\mathcal{E}) = \emptyset$ iff $\mathcal{E}$ is not rooted;

(iii) $\mathcal{E}$ is pure, if $\mathcal{E}$ is in standard form.

Example 1. *First, consider the RC-structure $\mathcal{E}_1 = (E_1, \vdash_1, L, l_1)$ from [15], where $E_1 = \{a, b, c\}$; $\vdash_1$ consists of $\emptyset \vdash_1 X$ for all $X \neq \{a, b\}$ and $\{c\} \vdash_1 \{a, b\}$; $L = E_1$; and l_1 is the identity labeling function. It is easy to see that $LC(\mathcal{E}_1) = Conf(\mathcal{E}_1) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$. In addition, we get $\prec_X^{\mathcal{E}_1} = \emptyset$, for all $X \in Conf(\mathcal{E}_1)$, and, for example, $\{b\} \rightarrow \{b, c, a\}$. This RC-structure models the initial conflict between the events a and b that can be resolved by the occurrence of the event c. The RC-structure is:*

- *rooted (because $(\emptyset, \emptyset) \in \vdash_1$),*
- *pure (because $X \cap Y = \emptyset$, for all $(X, Y) \in \vdash_1$),*
- *not singular (because $(\{c\}, \{a, b\}) \in \vdash_1$),*
- *locally conjunctive (because X is unique, for all $(X, Y) \in \vdash_1$)).*
- *not in standard form (see the standard form of $\mathcal{E}_1$ in Example 3).*

Second, consider the RC-structure $\mathcal{E}_2 = (E_2, \vdash_2, L, l_2)$, where $E_2 = \{a, b, c\}$; $\vdash_2 = \{(\emptyset, \emptyset), (\emptyset, \{a\}), (\{a\}, \{b\}), (\{a\}, \{c\}), (\{b\}, \{c\}), (\emptyset, \{a, b\}), (\emptyset, \{a, c\}), (\emptyset, \{b, c\}), (\emptyset, \{a, b, c\})\}$; $L = E_2$; and l_2 is the identity labeling function. Using the relation $\vdash_2$, we get that $LC(\mathcal{E}_2) = Conf(\mathcal{E}_2) = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}\}$. In addition, $\prec_{\{a,b\}}^{\mathcal{E}_2} = \{(a, b)\}$, $\prec_{\{a,c\}}^{\mathcal{E}_2} = \{(a, c)\}$,

$\prec^{\mathcal{E}_2}_{\{b,c\}} = \{(b,c)\}$, $\prec^{\mathcal{E}_2}_{\{a,b,c\}} = \{(a,b)\}$, *and, for example,* $\{a\} \to \{a,b,c\}$. *The RC-structure is:*

- *rooted (because* $(\emptyset,\emptyset) \in \vdash_2$*),*
- *pure (because* $X \cap Y = \emptyset$*, for all* $(X,Y) \in \vdash_2$*),*
- *singular (because* $X = \emptyset \vee \mid Y \mid = 1$*, for all* $(X,Y) \in \vdash_2$*),*
- *not locally conjunctive (because* $(\{a\},\{c\}), (\{b\},\{c\}) \in \vdash_2$ *and* $(\emptyset,\{c\}) \notin \vdash_2$*),*
- *not in standard form (see the standard form of* $\mathcal{E}_2$ *in Example 2).*

Third, examine the RC-structure $\mathcal{E}_3 = (E_3, \vdash_3, L, l_3)$, *where* $E_3 = \{a,b\}$; $\vdash_3 = \{(\emptyset,\emptyset), (\emptyset,\{a\}), (\emptyset,\{b\}), (\{a\},\{a,b\})\}$; $L = E_3$; *and* l_3 *is the identity labeling function. It is easy to see that* $LC(\mathcal{E}_3) = Conf(\mathcal{E}_3) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$. *It is not difficult to observe that* $\prec^{\mathcal{E}_3}_{\{a,b\}} = \emptyset$. *Notice that* $\{a\} \to \{a,b\}$ *and* $\{b\} \not\to \{a,b\}$. *The RC-structure is:*

- *rooted (because* $(\emptyset,\emptyset) \in \vdash_3$*),*
- *not pure (because* $(\{a\},\{a,b\}) \in \vdash_3$*),*
- *not singular (because* $(\{a\},\{a,b\}) \in \vdash_3$*),*
- *locally conjunctive (because* X *is unique, for all* $(X,Y) \in \vdash_3$*)),*
- *not in standard form (see the standard form of* $\mathcal{E}_3$ *in Example 2).*

Fourth, check the RC-structure $\mathcal{E}_4 = (E_4, \vdash_4, L, l_4)$, *where* $E_4 = \{a\}$; $\vdash_4 = \{(\emptyset,\{a\}), (\{a\},\emptyset)\}$; $L = E_4$; *and* l_4 *is the identity labeling function. It is easy to see that* $LC(\mathcal{E}_4) = \{\{a\}\}$ *and* $Conf(\mathcal{E}_4) = \emptyset$. *The RC-structure is:*

- *not rooted (because* $(\emptyset,\emptyset) \notin \vdash_4$*),*
- *pure (because* $X \cap Y = \emptyset$*, for all* $(X,Y) \in \vdash_4$*),*
- *not singular (because* $(\{a\},\emptyset) \in \vdash_4$*),*
- *locally conjunctive (because* X *is unique, for all* $(X,Y) \in \vdash_4$*)),*
- *in standard form.*

Fifth, observe $\mathcal{E}_5 = (E_5, \vdash_5, L, l_5)$, *where* $E_5 = \{a,b,c\}$; $\vdash_5 = \{(\emptyset,\emptyset), (\emptyset,\{a\}), (\emptyset,\{b\}), (\{a\},\{c\}), (\{b\},\{c\}), (\emptyset,\{a,c\}), (\emptyset,\{b,c\})\}$; $L = E_5$; *and* l_5 *is the identity labeling function. It is not difficult to see that* $LC(\mathcal{E}_5) = Conf(\mathcal{E}_5) = \{\emptyset, \{a\}, \{b\}, \{a,c\}, \{b,c\}\}$. *Also,* $\prec^{\mathcal{E}_5}_{\{a,c\}} = \{(a,c)\}$, $\prec^{\mathcal{E}_5}_{\{b,c\}} = \{(b,c)\}$, *and* $\{a\} \to \{a,c\}$ *and* $\{b\} \to \{b,c\}$. *The RC-structure is:*

- *rooted (because* $(\emptyset,\emptyset) \in \vdash_5$*),*
- *pure (because* $X \cap Y = \emptyset$*, for all* $(X,Y) \in \vdash_5$*),*
- *singular (because* $X = \emptyset \vee \mid Y \mid = 1$*, for all* $(X,Y) \in \vdash_5$*),*
- *locally conjunctive (because* $(\{a\},\{c\}), (\{b\},\{c\})$*, and* $\neg(Con(\{a,b,c\}))$*),*
- *not in standard form (see the standard form of* $\mathcal{E}_5$ *in Example 2).*

Next, we establish essential facts concerning the standard form of *RC*-structures.

Proposition 1. *Given an RC-structure* $\mathcal{E} = (E, \vdash, L, l)$,

(i) $\mathcal{E}$ *can be transformed into the RC-structure* $SF(\mathcal{E}) = (E, \widetilde{\vdash}, L, l)$ *in standard form such that* $LC(\mathcal{E}) = LC(SF(\mathcal{E}))$. *Moreover,* $SF(\mathcal{E})$ *is rooted iff* $\mathcal{E}$ *is rooted;*

(ii) $Conf(\mathcal{E}) = Conf(SF(\mathcal{E}))$, *and, moreover,* $X \to X'$ *in* $\mathcal{E} \iff X \to X'$ *in* $SF(\mathcal{E})$, *if* $\mathcal{E}$ *is a pure RC-structure;*

(iii) for a configuration $X \in Conf(\mathcal{E})$, $\preceq_X^{\mathcal{E}} = \preceq_X^{SF(\mathcal{E})}$, *if* $\mathcal{E}$ *is a pure, singular and locally conjunctive RC-structure.*

Proof. (i) For the transformation $\mathcal{E} = (E, \vdash, L, l)$ into standard form $SF(\mathcal{E}) = (E, \widetilde{\vdash}, L, l)$, we can directly put $\widetilde{\vdash} = \{(A, B) \mid B \subseteq C \in LC(\mathcal{E}), A = C \setminus B\}$.

Clearly, $LC(\mathcal{E}) = \emptyset$ iff $LC(SF(\mathcal{E})) = \emptyset$. Suppose $X \in LC(\mathcal{E})$. For any $Y \subseteq X$, take $Z = X \setminus Y$. Then, $Z \widetilde{\vdash} Y$, by the construction of $\widetilde{\vdash}$. Hence, $Z \cup Y = X \in LC(SF(\mathcal{E}))$. Conversely, assume $X \in LC(SF(\mathcal{E}))$. Then, there is $Z \subseteq X$ such that $Z \widetilde{\vdash} X$. By the construction of $\widetilde{\vdash}$, $X = Z \cup X \in LC(\mathcal{E})$. Thus, $LC(\mathcal{E}) = LC(SF(\mathcal{E}))$. It is easy to see that $SF(\mathcal{E})$ is rooted iff $\mathcal{E}$ is rooted.

(ii) Assume $\mathcal{E}$ being a pure *RC*-structure. Using Observation (ii) and item (i), it is straightforward to show that $Conf(\mathcal{E}) = \emptyset$ iff $Conf(SF(\mathcal{E})) = \emptyset$. Take an arbitrary $X \in Conf(SF(\mathcal{E}))$. This means that $X = \{e_1, \ldots, e_n\}$ $(n \geq 0)$ such that for all $i \leq n$ and for all $Y \subseteq \{e_1, \ldots, e_i\}$, there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \widetilde{\vdash} Y$. Clearly, $X_i = \{e_1, \ldots, e_i\} \in LC(SF(\mathcal{E}))$, for all $i \leq n$. Thanks to item (i), $X_i \in LC(\mathcal{E})$, for all $i \leq n$. Take arbitrary $i \leq n$ and $Y \subseteq X_i$. Consider two possible cases.

$e_i \in Y$. Since $X_i \in LC(\mathcal{E})$, there is $Z \subseteq X_i$ such that $Z \vdash Y$. Due to $\mathcal{E}$ being a pure *RC*-structure, $Z \cap Y = \emptyset$. Hence, $e_i \notin Z$. So, $Z \subseteq X_{i-1}$.

$e_i \notin Y$. This means that $Y \subseteq X_{i-1}$. Since $X_{i-1} \in LC(\mathcal{E})$, there is $Z \subseteq X_{i-1}$ such that $Z \vdash Y$.

Thus, $X \in Conf(\mathcal{E})$.

Suppose $X \in Conf(\mathcal{E})$. By Observation (iii), $SF(\mathcal{E})$ is a pure *RC*-structure. Following similar lines of the proof in the opposite direction, we obtain $X \in Conf(SF(\mathcal{E}))$.

Check that $X \to X'$ in $\mathcal{E} \iff X \to X'$ in $SF(\mathcal{E})$. Assume that $X \to X'$ in $\mathcal{E}$. This means that $X,\ X' \in Conf(\mathcal{E})$, $X = \{e_1, \ldots, e_k\}$ and $X' = \{e_1, \ldots, e_n\}$ $(0 \leq k \leq n)$. Due to $\mathcal{E}$ being a pure *RC*-structure, we may conclude that $X,\ X' \in Conf(SF(\mathcal{E}))$ with the same ordering of events as shown above. Hence, $X \to X'$ in $SF(\mathcal{E})$. Conversely, suppose that $X \to X'$ in $SF(\mathcal{E})$. The proof of the result is analogous to that in the opposite direction. Thus, $X \to X'$

in $\mathcal{E}$.

(iii) According to items (i) and (ii), we have that $LC(\mathcal{E}) = LC(SF(\mathcal{E}))$ and $Conf(\mathcal{E}) = Conf(SF(\mathcal{E}))$, respectively, since $\mathcal{E}$ is a pure RC-structure.

Consider arbitrary events $e, d \in X \in Conf(\mathcal{E})$ such that $e \prec_X^{\mathcal{E}} d$. Check that $e \prec_X^{SF(\mathcal{E})} d$. Take an arbitrary set $Z \subseteq X$ such that $Z \vdash_{SF(\mathcal{E})} \{d\}$. By the construction of $SF(\mathcal{E})$, we obtain $Z \cup \{d\} \in LC(\mathcal{E})$. This means that for all $Z' \subseteq Z \cup \{d\}$ there exists a set $Z'' \subseteq Z \cup \{d\}$ such that $Z'' \vdash_{\mathcal{E}} Z'$. Hence, we can find a set $W \subseteq Z \cup \{d\}$ such that $W \vdash_{\mathcal{E}} \{d\}$. Note that $W \cap \{d\} = \emptyset$, due to $\mathcal{E}$ being a pure RC-structure. So, $W \subseteq Z$. By the definition of $\prec_X^{\mathcal{E}}$, we have that $e \in W$. Hence, $e \prec_X^{SF(\mathcal{E})} d$.

Conversely, take arbitrary events $e, d \in X$ such that $e \prec_X^{SF(\mathcal{E})} d$. We need to show that $e \preceq_X^{\mathcal{E}} d$. Since $X \in Conf(\mathcal{E})$, we have an ordered set $X = \{e_1, \ldots, e_m\}$ such that for all $i \leq m$ and all $Y \subseteq \{e_1, \ldots, e_i\}$ there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \vdash_{\mathcal{E}} Y$. W.l.o.g. assume $d = e_q$ $(1 \leq q \leq m)$. Clearly, $X_q = \{e_1, \ldots, e_q\} \in LC(\mathcal{E})$ and $Con(X_q)$. Define the set $C^d = \{x \in X_q \mid x \preceq_X^{\mathcal{E}} d\}$. Check that $C^d \in LC(\mathcal{E})$. Take an arbitrary $W \subseteq C^d$. Treat three admissible cases.

$|W| = 0$. Since $X \in Conf(\mathcal{E})$, it holds that $Conf(\mathcal{E}) \neq \emptyset$. By Observation (ii), we get that $\mathcal{E}$ is a rooted RC-structure. Hence, we have $\emptyset \vdash_{\mathcal{E}} \emptyset = W$.

$|W| = 1$. W.l.o.g. assume $W = \{z\}$. As $W \subseteq X_q \in LC(\mathcal{E})$, there is at least one $A_i \subseteq X_q$ such that $A_i \vdash_{\mathcal{E}} \{z\}$. Due to $\mathcal{E}$ being a locally conjunctive RC-structure and $Con(X_q)$, we can find $A = \bigcap_{i \in I} A_i \subseteq X_q$ such that $A \vdash_{\mathcal{E}} \{z\}$. Then, we obtain that $a \prec_X^{\mathcal{E}} z$ for all $a \in A$, thanks to the definition of $\prec_X^{\mathcal{E}}$. Because of $z \in C^d$, we have $z \preceq_X^{\mathcal{E}} d$. Hence, for all $a \in A$ it holds $a \preceq_X^{\mathcal{E}} d$, by the transitivity of $\preceq_X^{\mathcal{E}}$. Thus, $A \subseteq C^d$.

$|W| \geq 2$. As $W \subseteq X_q \in LC(\mathcal{E})$, we can find $W' \subseteq X_q$ such that $W' \vdash_{\mathcal{E}} W$. Since $\mathcal{E}$ is a singular RC-structure and $|W| \geq 2$, it holds that $W' = \emptyset \subseteq C^d$.

Thus, $C^d \in LC(\mathcal{E})$ and $d \in C^d$. Due to the construction of $SF(\mathcal{E})$, we obtain $C^d \setminus \{d\} \vdash_{SF(\mathcal{E})} \{d\}$. As $e \prec_X^{SF(\mathcal{E})} d$, it holds $e \in C^d \setminus \{d\}$. This means that $e \preceq_X^{\mathcal{E}} d$, by the definition of C^d. □

We illustrate the validity of Proposition 1.

Example 2. *First, consider the rooted, pure, not locally conjunctive and singular RC-structure* $\mathcal{E}_2 = (E_2, \vdash_2, L, l_2)$ *from Example 1. Recall that* $E_2 = \{a, b, c\}$; $\vdash_2 = \{(\emptyset, \emptyset), (\emptyset, \{a\}), (\{a\}, \{b\}), (\{a\}, \{c\}), (\{b\}, \{c\}), (\emptyset, \{a, b\}), (\emptyset, \{a, c\}), (\emptyset, \{b, c\}), (\emptyset, \{a, b, c\})\}$; $L = E_2$; *and* l_2 *is the identity labeling function. We know from Example 1 that* $LC(\mathcal{E}_2) = Conf(\mathcal{E}_2) = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}\}$, *and* $\prec_{\{a,b,c\}}^{\mathcal{E}_2} = \{(a, b)\}$.

The structure $\mathcal{E}_2$ *can be presented in standard form* $\widetilde{\mathcal{E}}_2$, *with* $\widetilde{\vdash}_2 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset),\ (\{a\},\{b\}),\ (\{b\},\{a\}),\ (\emptyset,\{a,b\}),\ (\{a,b\},\emptyset),\ (\{a\},\{c\}),\ (\{c\},\ \{a\}),\ (\emptyset,\ \{a,c\}),\ (\{a,c\},\ \emptyset),\ (\{a\},\{b,c\}),\ (\{b,c\},\{a\}),\ (\{b\},\ \{a,c\}),\ (\{a,c\},\ \{b\}),\ (\{c\},\ \{a,b\}),\ (\{a,b\},\ \{c\}),\ (\emptyset,\ \{a,b,c\}),\ (\{a,b,c\},\ \emptyset)\}$. *It is easy to see that* $LC(\mathcal{E}_2) = LC(\widetilde{\mathcal{E}}_2)$, *and* $Conf(\mathcal{E}_2) = Conf(\widetilde{\mathcal{E}}_2)$, *as* $\mathcal{E}_2$ *is pure. Using the relation* $\widetilde{\vdash}_2$, *we get* $\prec^{\widetilde{\mathcal{E}}_2}_{\{a,b,c\}} = \{(a,b),(a,c)\}$. *Then,* $\prec^{\mathcal{E}_2}_{\{a,b,c\}} \neq \prec^{\widetilde{\mathcal{E}}_2}_{\{a,b,c\}}$, *because* $\mathcal{E}_2$ *is not locally conjunctive.*

Second, examine the rooted, not pure, locally conjunctive and not singular RC-structure $\mathcal{E}_3 = (E_3, \vdash_3, L, l_3)$ *from Example 1. Here,* $E_3 = \{a,b\}$; $\vdash_3 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\emptyset,\{b\}),\ (\{a\},\{a,b\})\}$; $L = E_3$; *and* l_3 *is the identity labeling function. We know from Example 1 that* $LC(\mathcal{E}_3) = Conf(\mathcal{E}_3) = \{\emptyset,\ \{a\},\ \{b\},\ \{a,b\}\}$, *and* $\prec^{\mathcal{E}_3}_{\{a,b\}} = \emptyset$. *In addition,* $\{a\} \to \{a,b\}$ *and* $\{b\} \not\to \{a,b\}$ *in* $\mathcal{E}_3$.

The structure $\mathcal{E}_3$ *can be presented in standard form* $\widetilde{\mathcal{E}}_3$, *with* $\widetilde{\vdash}_3 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset),\ (\emptyset,\{b\}),\ (\{b\},\emptyset),\ (\{a\},\{b\}),\ (\{b\},\{a\}),\ (\emptyset,\{a,b\}),\ (\{a,b\},\emptyset)\}$. *It is not difficult to see that* $Conf(\mathcal{E}_3) = \{\emptyset,\ \{a\},\ \{b\},\ \{a,b\}\}$, *and* $\prec^{\widetilde{\mathcal{E}}_3}_{\{a,b\}} = \emptyset$. *Also it is true that* $\{a\} \to \{a,b\}$ *and, however,* $\{b\} \to \{a,b\}$ *in* $\widetilde{\mathcal{E}}_3$, *because* $\mathcal{E}_3$ *is not pure RC-structure.*

Third, examine the rooted, pure, locally conjunctive and singular RC-structure $\mathcal{E}_5 = (E_5, \vdash_5, L, l_5)$ *from Example 1. Recall that* $E_5 = \{a,b,c\}$; $\vdash_5 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\emptyset,\{b\}),\ (\{a\},\{c\}),\ (\{b\},\{c\}),\ (\emptyset,\{a,c\}),\ (\emptyset,\{b,c\})\}$; $L = E_5$; *and* l_5 *is the identity labeling function. We know that* $LC(\mathcal{E}_5) = Conf(\mathcal{E}_5) = \{\emptyset,\ \{a\},\ \{b\},\ \{a,c\},\ \{b,c\}\}$, *and* $\prec^{\mathcal{E}_5}_{\{a,c\}} = \{(a,c)\}$, $\prec^{\mathcal{E}_5}_{\{b,c\}} = \{(b,c)\}$.

The structure $\mathcal{E}_5$ *can be presented in standard form* $\widetilde{\mathcal{E}}_5$, *with* $\widetilde{\vdash}_5 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset),\ (\emptyset,\{b\}),\ (\{b\},\emptyset),\ (\{a\},\{c\}),\ (\{c\},\ \{a\}),\ (\emptyset,\ \{a,c\}),\ (\{a,c\},\ \emptyset),\ (\{b\},\{c\}),\ (\{c\},\ \{b\}),\ (\emptyset,\ \{b,c\}),\ (\{b,c\},\ \emptyset)\}$. *It is easy to see that* $LC(\mathcal{E}_5) = LC(\widetilde{\mathcal{E}}_5)$, *and* $Conf(\mathcal{E}_5) = Conf(\widetilde{\mathcal{E}}_5)$, *as* $\mathcal{E}_5$ *is pure. Using the relation* $\widetilde{\vdash}_5$, *we get* $\prec^{\widetilde{\mathcal{E}}_5}_{\{a,c\}} = \{(a,c)\}$ *and* $\prec^{\widetilde{\mathcal{E}}_5}_{\{b,c\}} = \{(b,c)\}$. *Then,* $\prec^{\mathcal{E}_5}_{\{a,c\}} = \prec^{\widetilde{\mathcal{E}}_5}_{\{a,c\}}$ *and* $\prec^{\mathcal{E}_5}_{\{b,c\}} = \prec^{\widetilde{\mathcal{E}}_5}_{\{b,c\}}$, *because* $\mathcal{E}_5$ *is a rooted, pure, locally conjunctive and singular RC-structure.*

2.2. Different Semantics for *RC*-structures

In this subsection, we introduce partial order multiset (pom) and step (step) semantics for *RC*-structures.

We first define auxiliary notions and notations. For configurations $X, X' \in Conf(\mathcal{E})$, we write:

- $X \to_{pom} X'$ iff $X \to X'$;
- $X \to_{step} X'$ iff $X \to X'$ in $\mathcal{E}$, and $X'' \in Conf(\mathcal{E})$, for all $X \subseteq X'' \subseteq X'$.

For $\star \in \{pom, step\}$, a configuration $X \in Conf(\mathcal{E})$ is a *configuration in $\star$-semantics* of $\mathcal{E}$ iff $\emptyset \rightarrow_\star^* X$, where $\rightarrow_\star^*$ is the reflexive and transitive closure of $\rightarrow_\star$. Let $Conf_\star(\mathcal{E})$ denote the set of configurations in $\star$-semantics of $\mathcal{E}$.

Lemma 2. *Given an RC-structure $\mathcal{E}$ and $\star \in \{pom, step\}$,*

(i) $Conf_\star(\mathcal{E}) = Conf(\mathcal{E})$;

(ii) $X \rightarrow_\star X'$ in $\mathcal{E} \iff X \rightarrow_\star X'$ in $SF(\mathcal{E})$, if $\mathcal{E}$ is a pure RC-structure.

Proof. (i) Due to the definition of $\star$-semantics, we have $Conf_\star(\mathcal{E}) \subseteq Conf(\mathcal{E})$. We shall show $Conf(\mathcal{E}) \subseteq Conf_\star(\mathcal{E})$. Take an arbitrary $X \in Conf(\mathcal{E})$. This means that X can be represented as an ordered set $\{e_1, \ldots, e_n\}$ $(n \geq 0)$ such that for all $i \leq n$ and for all $Y \subseteq \{e_1, \ldots, e_i\}$, there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \vdash Y$. Clearly, $X_i = \{e_1, \ldots, e_i\} \in Conf(\mathcal{E})$ for all $i \leq n$. If $\mathcal{E}$ is a rooted *RC*-structure, it holds $\emptyset \in Conf(\mathcal{E})$. Then, we obtain $\emptyset \rightarrow_\star^* X$, by the definitions of a configuration and the relation $\rightarrow_\star$. Hence, $X \in Conf_\star(\mathcal{E})$. If $\mathcal{E}$ is not rooted, it is true that $Conf_\star(\mathcal{E}) = Conf(\mathcal{E}) = \emptyset$.

(ii) It follows from the definitions of the transition relations and Proposition 1(ii). □

For an *RC*-structure $\mathcal{E}$ and $X, X' \in Conf(\mathcal{E})$, we write $\mathcal{E}\lceil(X' \setminus X) = ((X' \setminus X), \preceq_{(X' \setminus X)} = \preceq_{X'} \cap (X' \setminus X \times X' \setminus X), L, l\,|_{(X' \setminus X)})$, if $X \rightarrow_{pom} X'$. We say that $\mathcal{E}\lceil(X' \setminus X)$ and $\mathcal{E}\lceil(Y' \setminus Y)$ are *isomorphic* iff there is a bijection $\iota : (X' \setminus X) \rightarrow (Y' \setminus Y)$ such that $(x, x') \in \preceq_{(X' \setminus X)}$ iff $(\iota(x), \iota(x')) \in \preceq_{(Y' \setminus Y)}$, for all $x, x' \in (X' \setminus X)$, and $l(x) = l(\iota(x))$, for all $x \in (X' \setminus X)$. Let $[\mathcal{E}\lceil(X' \setminus X)]$ denote the isomorphic class of $\mathcal{E}\lceil(X' \setminus X)$.

Given $\star \in \{pom, step\}$, an *RC*-structure $\mathcal{E}$ over L, and configurations $X, X' \in Conf_\star(\mathcal{E})$ such that $X \rightarrow_\star X'$, we write:

$$l_\star(X' \setminus X) = \begin{cases} [\mathcal{E}\lceil(X' \setminus X)], & \text{if } \star = pom, \\ M\ (\forall a \in L\colon M(a) = |\{e \in X' \setminus X \mid l(e) = a\}|), & \text{if } \star = step. \end{cases}$$

3. Removal Operator for RC-structures

The standard form of *RC*-structures and their ability to specify impossible events allow us to develop a relatively simple structural definition of the removal operator which is useful for constructing residuals of *RC*-structures.

Definition 3. *For an RC-structure $\mathcal{E} = (E, \vdash, L, l)$ in standard form and $X \in LC(\mathcal{E})$, the*

removal operator is defined as follows: $\mathcal{E} \setminus X = (E', \vdash', L, l')$, *where*

$$\begin{aligned} E' &= E \setminus X, \\ \vdash' &= \{(A', B') \mid \exists (A, B) \in \vdash \ s.t. \ A' = A \cap E', B' = B \cap E', (A' \cup B' \cup X) \in LC(\mathcal{E})\}, \\ l' &= l \mid_{E'} . \end{aligned}$$

According to the definition above, we remove all events in X and retain the events that, when combined with X, do not form left-closed sets and, therefore, conflict with some events in X; however, we make the conflicting retained events impossible by deleting their enabling relations.

Consider properties of the removal operator.

Lemma 3. *Given an RC-structure* $\mathcal{E}$ *in standard form and* $X \in Conf(\mathcal{E})$,

(i) $\mathcal{E} \setminus X$ *is an RC-structure;*

(ii) $(X \cup Y) \in LC(\mathcal{E}) \iff Y \in LC(\mathcal{E} \setminus X)$, *for any* $Y \subseteq E'$;

(iii) $\mathcal{E} \setminus X$ *is a rooted RC-structure* $\mathcal{E}$ *in standard form.*

Proof. (i) According to Definition 1, $\mathcal{E} \setminus X$ is an *RC*-structure.

(ii) Take an arbitrary $Y \subseteq E'$. Then, $X \cap Y = \emptyset$, due to Definition 3.

($\Rightarrow$) Suppose $(X \cup Y) \in LC(\mathcal{E})$. Then, for all $\widetilde{Y} \subseteq Y$, $(\widetilde{Y} \cup \widehat{Y}) \in LC(\mathcal{E})$, where $\widehat{Y} = X \cup Y \setminus \widetilde{Y}$. As $\mathcal{E}$ is in standard form, $\widehat{Y} \vdash \widetilde{Y}$, for all $\widetilde{Y} \subseteq Y$ and the corresponding $\widehat{Y}$. Obviously, $(\widetilde{Y} \cup (\widehat{Y}' = \widehat{Y} \setminus X) \cup X) \in LC(\mathcal{E})$ and $\widehat{Y}' \subseteq Y$. Due to the definition of $\vdash'$, for all $\widetilde{Y} \subseteq Y$, there exists $\widehat{Y}' \subseteq Y$ such that $\widehat{Y}' \vdash' \widetilde{Y}$. Thus, $Y \in LC(\mathcal{E} \setminus X)$.

($\Leftarrow$) Assume $Y \in LC(\mathcal{E} \setminus X)$. Then, for Y there is $\widehat{Y} \subseteq Y$ such that $\widehat{Y} \vdash' Y$. By the definition of $\vdash'$, this implies that $(X \cup \widehat{Y} \cup Y) = (X \cup Y) \in LC(\mathcal{E})$.

(iii) We first show that $\mathcal{E} \setminus X$ is in standard form.

($\Rightarrow$) Suppose $A' \vdash' B'$. Then, we can find $A \vdash B$ such that $A' = A \cap E'$, $B' = B \cap E'$ and $(A' \cup B' \cup X) \in LC(\mathcal{E})$, due to the definition of $\vdash'$. Since $\mathcal{E}$ is in standard form, it holds that $A \cap B = \emptyset$. This implies that $A' \cap B' = \emptyset$. Thanks to item (ii), we get that $(A' \cup B') \in LC(\mathcal{E} \setminus X)$.

($\Leftarrow$) Assume $C' \in LC(\mathcal{E} \setminus X)$. Take $B' \subseteq C'$ and $A' = C' \setminus B'$. According to item (ii), $(C' \cup X) = (A' \cup B' \cup X) \in LC(\mathcal{E})$. Moreover, since $(A' \cup X) \cap B' = \emptyset$, we get that $A' \cup X \vdash B'$, due to $\mathcal{E}$ being in standard form. Hence, $A' \vdash' B'$, by the definition of $\vdash'$.

As $X \in LC(\mathcal{E})$ and $\mathcal{E}$ is in standard form, $(\emptyset, \emptyset) \in \vdash'$, i.e. $\mathcal{E} \setminus X$ is rooted. □

Next we establish the relationships between configurations and partial orders in the original *RC*-structure and its residuals.

Proposition 2. *Given an RC-structure $\mathcal{E}$ in standard form,*

(i) for any $X, X' \in Conf(\mathcal{E})$ such that $X \to X'$,

(a) $X' \setminus X \in Conf(\mathcal{E} \setminus X)$;

(b) $\preceq^{\mathcal{E}\setminus X}_{X'\setminus X} = \preceq^{\mathcal{E}}_{X'} \cap (X'\setminus X \times X'\setminus X)$, if $\mathcal{E}$ is the standard form of a singular RC-structure;

(ii) for any $\mathcal{E}' = \mathcal{E} \setminus X$, with $X \in Conf(\mathcal{E})$, and for any $\mathcal{E}'' = \mathcal{E}' \setminus X'$, with $X' \in Conf(\mathcal{E}')$, $X \to X \cup X'$ in $\mathcal{E}$, and $\mathcal{E}'' = \mathcal{E} \setminus (X \cup X')$.

Proof. (i(a)) Take an arbitrary $X, X' \in Conf(\mathcal{E})$ such that $X \to X'$. As $X \to X'$, we have $X = \{e_1, \ldots, e_m\}$ $(m \geq 0)$, $X' = \{e_1, \ldots, e_m, e_{m+1}, \ldots, e_n\}$ $(n \geq 0)$, and $m \leq n$. Let $X' \setminus X = \{e_{m+1}, \ldots, e_n\}$. Check that $X' \setminus X \in Conf(\mathcal{E} \setminus X)$. Take an arbitrary $i \leq n - m$ and an arbitrary $A \subseteq \{e_{m+1}, \ldots, e_{m+i}\}$. Consider possible cases.

$A = \emptyset$. We have $\emptyset \vdash' \emptyset$, because $\mathcal{E} \setminus X$ is rooted, by Lemma 3(iii).

$A \neq \emptyset$. Notice that $\{e_1, \ldots, e_{m+i}\} \in Conf(\mathcal{E}) \subseteq LC(\mathcal{E})$, for all $i \leq n - m$. Define $B = \{e_1, \ldots, e_{m+i}\} \setminus A$, if $e_{m+i} \in A$, and $B = \{e_1, \ldots, e_{m+i-1}\} \setminus A$, otherwise. Clearly, $B \subseteq \{e_1, \ldots, e_{m+i-1}\}$. Since $\mathcal{E}$ is in standard form, we get $B \vdash A$. According to Lemma 3(ii), it holds $B \setminus X \cup A \in LC(\mathcal{E} \setminus X)$. We know that $\mathcal{E} \setminus X$ is in standard form, thanks to Lemma 3(iii). Hence, $B \setminus X \vdash' A$.

Thus, $X' \setminus X \in Conf(\mathcal{E} \setminus X)$.

(i(b)) Take arbitrary events $e, d \in X' \setminus X$. Assume $e \prec^{\mathcal{E}}_{X'} d$. Check that $e \prec^{\mathcal{E}\setminus X}_{X'\setminus X} d$. Take an arbitrary set $Z \subseteq X'\setminus X$ such that $Z \vdash_{\mathcal{E}\setminus X} \{d\}$. By Definition 3, we obtain $Z \cup \{d\} \cup X \in LC(\mathcal{E})$. This means that for all $Z' \subseteq Z \cup \{d\} \cup X$ there exists a set $Z'' \subseteq Z \cup \{d\} \cup X$ such that $Z'' \vdash_{\mathcal{E}} Z'$. Hence, we can find a set $W \subseteq Z \cup \{d\} \cup X$ such that $W \vdash_{\mathcal{E}} \{d\}$. By Observation (iii), it holds that $\mathcal{E}$ is a pure *RC*-structure. Thus, $W \cap \{d\} = \emptyset$. So, $W \subseteq Z \cup X \subset X'$. By the definition of $\prec^{\mathcal{E}}_{X'}$, we have $e \in W$. Since $e \in X' \setminus X$ and $W \subseteq Z \cup X$, we conclude $e \in Z$. Hence, $e \prec^{\mathcal{E}\setminus X}_{X'\setminus X} d$.

Conversely, suppose $e \prec^{\mathcal{E}\setminus X}_{X'\setminus X} d$. We need to show that $e \prec^{\mathcal{E}}_{X'} d$. Consider an arbitrary set $Z \subseteq X'$ such that $Z \vdash_{\mathcal{E}} \{d\}$. Due to $\mathcal{E}$ being in standard form, we obtain that $Z \cup \{d\} \in LC(\mathcal{E})$. Moreover, $d \notin Z$, by Observation (iii). As $\mathcal{E} = SF(\mathcal{F})$, we obtain $LC(\mathcal{E}) = LC(\mathcal{F})$, due to Proposition 1(i). It is sufficient to check that $X \cup Z \cup \{d\} \in LC(\mathcal{F})$. Take an arbitrary $W \subseteq X \cup Z \cup \{d\}$. Three cases are admissible.

$|W| = 0$. Since $X' \in Conf(\mathcal{E})$, we may conclude that $Conf(\mathcal{E}) \neq \emptyset$. This means that $\mathcal{E}$ is rooted, by Observation (ii). Due to Proposition 1 (i), we get that $\mathcal{F}$ is a rooted *RC*-structure. Hence, $\emptyset \vdash_{\mathcal{F}} \emptyset = W$.

$|W| = 1$. W.l.o.g. assume $W = \{z\}$. Consider two possible cases.

$z \in X$. Since $X \in LC(\mathcal{E}) = LC(\mathcal{F})$, there is $W' \subseteq X \subseteq X \cup Z \cup \{d\}$ such that $W' \vdash_{\mathcal{F}} \{z\}$.

$z \notin X$. Hence, $z \in (Z \cup \{d\}) \setminus X$. As $Z \cup \{d\} \in LC(\mathcal{E}) = LC(\mathcal{F})$, we can find $W' \subseteq Z \cup \{d\} \subseteq X \cup Z \cup \{d\}$ such that $W' \vdash_{\mathcal{F}} \{z\}$.

$|W| \geq 2$. Since $W \subseteq X \cup Z \cup \{d\} \subseteq X' \in LC(\mathcal{E}) = LC(\mathcal{F})$, there is $W' \subseteq X'$ such that $W' \vdash_{\mathcal{F}} W$.

Due to $\mathcal{F}$ being a singular *RC*-structure and $|W| \geq 2$, it is true that $W' = \emptyset \subseteq X \cup Z \cup \{d\}$. Thus, $X \cup Z \cup \{d\} \in LC(\mathcal{E}) = LC(\mathcal{F})$. Due to Definition 3, we obtain $Z \setminus X \vdash_{\mathcal{E}\setminus X} \{d\}$. As $e \prec^{\mathcal{E}\setminus X}_{X'\setminus X} d$, it holds $e \in Z \setminus X$. Thus, $e \prec^{\mathcal{E}}_{X'} d$.

(ii) Assume that $X \in \mathit{Conf}(\mathcal{E})$ and $X' \in \mathit{Conf}(\mathcal{E}')$, where $\mathcal{E}' = \mathcal{E} \setminus X$. Then, we can arrange $X = \{e_1, \ldots, e_n\}$ so that for all $i \leq n$ and for all $Y \subseteq \{e_1, \ldots, e_i\}$, there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \vdash Y$, and arrange $X' = \{e'_1, \ldots, e'_m\}$ so that for all $j \leq m$ and for all $Y' \subseteq \{e'_1, \ldots, e'_j\}$, there is $Z' \subseteq \{e'_1, \ldots, e'_{i-1}\}$ such that $Z' \vdash' Y'$. Define $e_{n+1} = e'_1$, …, $e_{n+m} = e'_m$. Take an arbitrary $i \leq n + m$ and an arbitrary $Y \subseteq \{e_1, \ldots, e_i\}$. If $i \leq n$, the result follows from the fact that $X = \{e_1, \ldots, e_n\} \in \mathit{Conf}(\mathcal{E})$. Consider the case when $n < i$. Define $Y' = Y \cap E'$. Due to $X' \in \mathit{Conf}(\mathcal{E}')$, there is $Z' \subseteq \{e'_1, \ldots, e'_{i-1}\}$ such that $Z' \vdash' Y'$. By virtue of Lemma 3(iii), $\mathcal{E}' = \mathcal{E} \setminus X$ is in standard form. Thus, $Z' \cup Y' \in LC(\mathcal{E}')$. According to Lemma 3(ii), we may conclude $Z' \cup Y' \cup X \in LC(\mathcal{E})$. Due to $\mathcal{E}$ being in standard form, this implies $Z \vdash Y$, where $Z = (Z' \cup Y' \cup X) \setminus Y$. Clearly, $Z \subseteq Z' \cup X \subseteq \{e_1, \ldots, e_{i-1}\}$. Hence, $X \cup X' \in \mathit{Conf}(\mathcal{E})$. Obviously, $X = \{e_1, \ldots, e_n\} \rightarrow X \cup X' = \{e_1, \ldots, e_n, e'_1, \ldots, e'_m\}$.

Check that $(\mathcal{E} \setminus X) \setminus X' = \mathcal{E} \setminus (X \cup X')$. Define $\mathcal{E}'' = \mathcal{E}' \setminus X'$ and $\widetilde{\mathcal{E}} = \mathcal{E} \setminus (X \cup X')$. We need to show $\mathcal{E}'' = \widetilde{\mathcal{E}}$. By Definition 3, $E'' = E' \setminus X' = E \setminus X \setminus X' = E \setminus (X \cup X') = \widetilde{E}$, and $l'' = l \mid_{E''} = l \mid_{\widetilde{E}} = \widetilde{l}$. Verify that $LC(\mathcal{E}'') = LC(\widetilde{\mathcal{E}})$. Thanks to Lemma 3(iii), $\widetilde{\mathcal{E}}$ and $\mathcal{E}''$ are rooted *RC*-structures in standard form. Then, $LC(\mathcal{E}'') \neq \emptyset$ and $LC(\widetilde{\mathcal{E}}) \neq \emptyset$. W.l.o.g. take an arbitrary $Y \in LC(\mathcal{E}'')$. According to Lemma 3(ii), $Y \in LC(\mathcal{E}'') \iff Y \cup X' \in LC(\mathcal{E}') \iff Y \cup X' \cup X \in LC(\mathcal{E}) \iff Y \in LC(\mathcal{E} \setminus (X \cup X')) = LC(\widetilde{\mathcal{E}})$. Hence, $\vdash'' = \widetilde{\vdash}$. □

We demonstrate the validity of Proposition 2.

Example 3. *First, consider the rooted, pure, locally conjunctive and not singular RC-structure* $\mathcal{E}_1 = (E_1, \vdash_1, L, l_1)$ *from Example 1. Recall that* $E_1 = \{a, b, c\}$; $\vdash_1$ *consists of* $\emptyset \vdash_1 X$ *for all* $X \neq \{a, b\}$ *and* $\{c\} \vdash_1 \{a, b\}$; $L = E_1$; *and* l_1 *is the identity labeling function. We know from Example 1 that* $LC(\mathcal{E}_1) = \mathit{Conf}(\mathcal{E}_1) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{c, a\}, \{b, c\}, \{b, c, a\}\}$.

By building the standard form of $\mathcal{E}_1$, *we get the enabling relation of* $\widetilde{\mathcal{E}}_1$: $\widetilde{\vdash}_1 = \{(\emptyset, \emptyset), (\emptyset, \{a\}), (\{a\}, \emptyset), (\emptyset, \{b\}), (\{b\}, \emptyset), (\emptyset, \{c\}), (\{c\}, \emptyset), (\emptyset, \{a, c\}), (\{a, c\}, \emptyset), (\{a\}, \{c\}), (\{c\}, \{a\}), (\emptyset, \{b, c\}), (\{b, c\}, \emptyset), (\{b\}, \{c\}), (\{c\}, \{b\}), (\{a\}, \{b, c\}), (\{b, c\}, \{a\}), (\{b\}, \{a, c\}),$

$(\{a,c\},\ \{b\})$, $(\{c\},\ \{a,b\})$, $(\{a,b\},\ \{c\})$, $(\emptyset,\ \{a,b,c\})$, $(\{a,b,c\},\ \emptyset)\}$. *It is easy to see that* $LC(\mathcal{E}_1) = LC(\widetilde{\mathcal{E}}_1)$ *and* $Conf(\mathcal{E}_1) = Conf(\widetilde{\mathcal{E}}_1)$. *In addition, we have that* $\{b\} \rightarrow \{b,c\}$ *and* $\{b\} \rightarrow \{b,c,a\}$ *in* $\widetilde{\mathcal{E}}_1$. *Moreover, it holds* $\prec^{\widetilde{\mathcal{E}}_1}_{\{b,c,a\}} = \emptyset$.

Construct the RC-structure $\widetilde{\mathcal{E}}_1 \setminus \{b\} = (\{a,c\}, \vdash^*_1, L, l^*_1 = l_1|_{\{a,c\}})$, *where* $\vdash^*_1 = \{(\emptyset,\emptyset),\ (\emptyset,\{c\}),\ (\{c\},\emptyset),\ (\emptyset,\{a,c\}),\ (\{a,c\},\emptyset),\ (\{a\},\{c\}),\ (\{c\},\ \{a\})\}$. *Then, we obtain* $Conf(\widetilde{\mathcal{E}}_1 \setminus \{b\}) = \{\emptyset,\ \{c\},\ \{c,a\}\}$. *So,* $\{b,c,a\} \setminus \{b\} = \{c,a\} \in Conf(\widetilde{\mathcal{E}}_1 \setminus \{b\})$. *On the other hand,* $\prec^{\widetilde{\mathcal{E}}_1}_{\{b,c,a\}} \cap (\{c,a\} \times \{c,a\}) = \emptyset \neq \prec^{\widetilde{\mathcal{E}}_1 \setminus \{b\}}_{\{c,a\}} = \{(c,a)\}$, *although* $\{b\} \rightarrow \{b,c,a\}$ *in* $\widetilde{\mathcal{E}}_1$. *This is because* $\mathcal{E}_1$ *is not singular.*

Next, construct the RC-structure $(\widetilde{\mathcal{E}}_1 \setminus \{b\}) \setminus \{c\} = (\{a\}, \vdash^{**}_1, L, l^{**}_1 = l^*_1|_{\{a\}})$, *where* $\vdash^{**}{}_1 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset)\}$.

At last, construct the RC-structure $\widetilde{\mathcal{E}}_1 \setminus (\{b\} \cup \{c\}) = (\{a\}, \vdash^{***}_1, L, l^{***}_1 = l^*_1|_{\{a\}})$, *where* $\vdash^{***}{}_1 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset)\}$. *So, we see that* $\{b\} \rightarrow \{b,c\}$ *in* $\widetilde{\mathcal{E}}_1$, *and* $(\widetilde{\mathcal{E}}_1 \setminus \{b\}) \setminus \{c\} = \widetilde{\mathcal{E}}_1 \setminus (\{b\} \cup \{c\})$, *confirming the validity of Proposition 2(ii).*

Second, consider the rooted, pure, locally conjunctive and singular RC-structure $\mathcal{E}_5 = (E_5,\ \vdash_5,\ L,\ l_5)$ *from Example 1. Recall that* $E_5 = \{a,b,c\}$; $\vdash_5 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\emptyset,\{b\}),\ (\{a\},\{c\}),\ (\{b\},\{c\}),\ (\emptyset,\{a,c\}),\ (\emptyset,\{b,c\})\}$; $L = E_5$; *and* l_5 *is the identity labeling function. Moreover,* $LC(\mathcal{E}_5) = Conf(\mathcal{E}_5) = \{\emptyset,\{a\},\{b\},\{a,c\},\{b,c\}\}$, *and* $\prec^{\mathcal{E}_5}_{\{a,c\}} = \{(a,c)\}$ *and* $\prec^{\mathcal{E}_5}_{\{b,c\}} = \{(b,c)\}$.

From Example 2 we know that the enabling relation in the standard form $\widetilde{\mathcal{E}}_5$ *of* $\mathcal{E}_5$ *looks like this:* $\widetilde{\vdash}_5 = \{(\emptyset,\emptyset),\ (\emptyset,\{a\}),\ (\{a\},\emptyset),\ (\emptyset,\{b\}),\ (\{b\},\emptyset),\ (\{a\},\{c\}),\ (\{c\},\ \{a\}),\ (\emptyset,\ \{a,c\}),\ (\{a,c\},\ \emptyset),\ (\{b\},\{c\}),\ (\{c\},\ \{b\}),\ (\emptyset,\ \{b,c\}),\ (\{b,c\},\ \emptyset)\}$. *In addition,* $LC(\mathcal{E}_5) = LC(\widetilde{\mathcal{E}}_5)$, $Conf(\mathcal{E}_5) = Conf(\widetilde{\mathcal{E}}_5)$, *and* $\prec^{\widetilde{\mathcal{E}}_5}_{\{a,c\}} = \{(a,c)\}$, $\prec^{\widetilde{\mathcal{E}}_5}_{\{b,c\}} = \{(b,c)\}$. *Notice that* $\{b\} \rightarrow \{b,c\}$ *in* $\widetilde{\mathcal{E}}_5$.

Consider the RC-structure $\widetilde{\mathcal{E}}_5 \setminus \{b\} = (\{a,c\},\ \vdash^*_5,\ L,\ l^*_5)$, *where* $\vdash^*_5 = \{(\emptyset,\ \emptyset),\ (\emptyset,\ \{c\}),\ (\{c\},\ \emptyset)\}$. *Then, we obtain* $Conf(\widetilde{\mathcal{E}}_5 \setminus \{b\}) = \{\emptyset,\ \{c\}\}$, *and* $\prec^{\widetilde{\mathcal{E}}_5 \setminus \{b\}}_{\{c\}} = \emptyset$. *Hence, it holds that* $\prec^{\widetilde{\mathcal{E}}_5}_{\{b,c\}} \cap (\{c\} \times \{c\}) = \prec^{\widetilde{\mathcal{E}}_5 \setminus \{b\}}_{\{c\}}$. *So, the example with* $\mathcal{E}_5$ *confirms the validity of Proposition 2(i(b)).*

4. Transition Systems $TC(\cdot)$ and $TR(\cdot)$ from RC-structures

In this section, we first give some basic definitions concerning labeled transition systems. Then, we define the mappings $TC(\mathcal{E})$ and $TR(\mathcal{E})$, which associate two distinct kinds of transition systems – one whose states are configurations and one whose states are residual RC-structures – with the RC-structure $\mathcal{E}$ labeled over the set L of labels.

A transition system $T = (S, \rightarrow, i)$ labeled over a set $\mathcal{L}$ of labels consists of a set of states S, a transition relation $\rightarrow \subseteq S \times \mathcal{L} \times S$, and an initial state $i \in S$. Two transition systems

labeled over $\mathcal{L}$ are *isomorphic* if their states can be mapped one-to-one to each other, preserving transitions and initial states.

Let L be a fixed set of labels in RC-structures). Let $\mathbb{L}_{step} := \mathbb{N}_0^L$ (the set of multisets over L, or functions from L to the non-negative integers), and $\mathbb{L}_{pom} := Pom_L$ (the set of isomorphic classes of partial orders labeled over L) be another sets of labels. (The sets will be used as the set of labels in the transition systems.)

We are ready to define labeled transition systems with configurations as states.

Definition 4. *For a rooted RC-structure $\mathcal{E}$ over L and $\star \in \{pom, step\}$,*

$$TC_\star(\mathcal{E}) \text{ is a transition system } (Conf_\star(\mathcal{E}), \multimap_\star, \emptyset) \text{ over } \mathbb{L}_\star,$$

where $X \stackrel{p}{\multimap_\star} X'$ iff $X \to_\star X'$ and $p = l_\star(X' \setminus X)$ in $\mathcal{E}$.

We exemplify a feature of the above definition.

Example 4. *Consider the not rooted RC-structure $\mathcal{E}_4 = (E_4, \vdash_4, L, l_4)$ from Example 1. Recall that $E_4 = \{a\}$; $\vdash_4 = \{(\emptyset, \{a\}), (\{a\}, \emptyset)\}$; $L = E_4$; and l_4 is the identity labeling function. We know that $LC(\mathcal{E}_4) = \{\{a\}\}$, and $Conf(\mathcal{E}_4) = \emptyset$, as $\mathcal{E}_4$ is not rooted, i.e. $(\emptyset, \emptyset) \notin \vdash_4$. We cannot construct the transition system $TC_\star(\mathcal{E}_4) = (Conf_\star(\mathcal{E}_4), \multimap_\star, \emptyset)$ over $\mathbb{L}_\star$ ($\star \in \{pom, step\}$) because the initial state $\emptyset$ must belong to $Conf(\mathcal{E}_4)$.*

Lemma 4. *Given a rooted RC-structure $\mathcal{E}$,*

(i) $TC_{step}(\mathcal{E}) = TC_{step}(SF(\mathcal{E}))$, if $\mathcal{E}$ is a pure RC-structure;

(ii) $TC_{pom}(\mathcal{E}) = TC_{pom}(SF(\mathcal{E}))$, if $\mathcal{E}$ is a pure, singular and locally conjunctive RC-structure.

Proof. Let $\star \in \{step, pom\}$. Due to Lemma 2(i), we obtain that $Conf_\star(\mathcal{E}) = Conf(\mathcal{E})$ and $Conf_\star(SF(\mathcal{E})) = Conf(SF(\mathcal{E}))$. Since $\mathcal{E}$ is a pure RC-structure, we have that $Conf(\mathcal{E}) = Conf(SF(\mathcal{E}))$, by Proposition 1(ii). Thus, we obtain $Conf_\star(\mathcal{E}) = Conf_\star(SF(\mathcal{E}))$. Moreover, according to Lemma 2(ii), the relation $\to_\star$ in $\mathcal{E}$ is equal to the relation $\to_\star$ in $SF(\mathcal{E})$.

Thanks to the construction of $SF(\mathcal{E})$ and the definition of $l_{step}(\cdot)$, it easy to see that $l^{\mathcal{E}}_{step} = l^{SF(\mathcal{E})}_{step}$.

If $\mathcal{E}$ is a pure, singular and locally conjunctive RC-structure, it holds that $\preceq^{\mathcal{E}}_X = \preceq^{SF(\mathcal{E})}_X$, for all $X \in Conf(\mathcal{E}) = Conf(SF(\mathcal{E}))$, due to Proposition 1(iii). Hence, $l^{\mathcal{E}}_{pom} = l^{SF(\mathcal{E})}_{pom}$.

So, by Definition 4, we get $\multimap^{\mathcal{E}}_\star = \multimap^{SF(\mathcal{E})}_\star$. Thus, $TC_\star(\mathcal{E}) = TC_\star(SF(\mathcal{E}))$. □

We illustrate the validity of Lemma 4.

Example 5. *From Example 1 we know that $\mathcal{E}_1$ is a rooted and pure RC-structure, $\mathcal{E}_3$ is a rooted and not pure RC-structure, and $\mathcal{E}_5$ is a rooted, pure, singular and locally conjunctive*

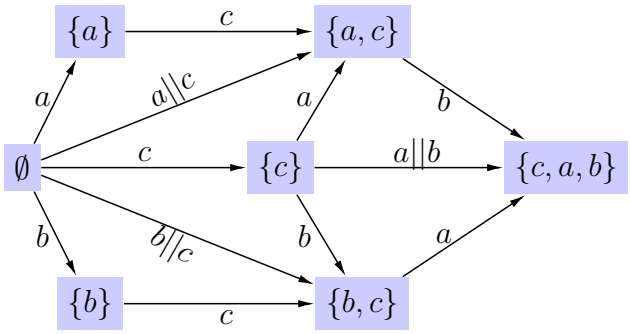

Fig. 1. The configuration transition system $TC_{step}(\mathcal{E}_1)$

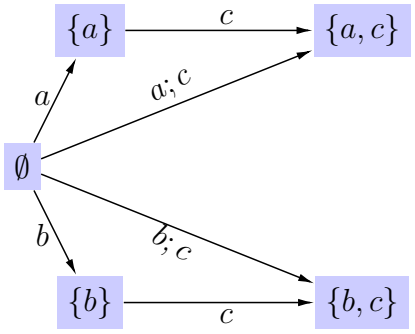

Fig. 2. The configuration transition system $TC_{pom}(\mathcal{E}_5)$

RC-structure. The transition system $TC_{step}(\mathcal{E}_1)$ is shown in Fig. 1, and the transition system $TC_{pom}(\mathcal{E}_5)$ is depicted in Fig. 2. Using Examples 3 and 2, respectively, it is easy to make sure that $TC_{step}(\mathcal{E}_1) = TC_{step}(SF(\mathcal{E}_1))$ and $TC_{pom}(\mathcal{E}_5) = TC_{pom}(SF(\mathcal{E}_5))$. However, this does not apply to $\mathcal{E}_3$ for both semantics. It is easy to see this by looking at Fig. 3 and 4.

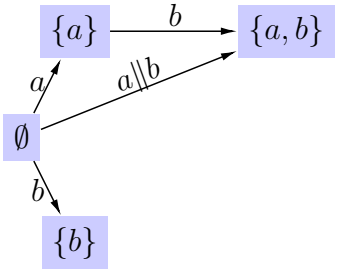

Fig. 3. The configuration transition system $TC_{step/pom}(\mathcal{E}_3)$

Next, introduce the definition of labeled transition systems with residuals as states.

Definition 5. *For an RC-structure $\mathcal{E}$ over L in standard form and $\star \in \{pom, step\}$,*

$$TR_\star(\mathcal{E}) \text{ is the transition system } (Reach_\star(\mathcal{E}), \rightharpoonup_\star, \mathcal{E}) \text{ over } \mathbb{L}_\star,$$

where $\mathcal{F} \xrightarrow{p}_\star \mathcal{F}'$ for some $p \in \mathbb{L}_\star$ iff $\mathcal{F}' = \mathcal{F} \setminus X$ and $\emptyset \rightarrow_\star X$ in $\mathcal{F}$, for some $X \in Conf_\star(\mathcal{F})$ with $p = l_\star(X \setminus \emptyset)$, and $Reach_\star(\mathcal{E}) = \{\mathcal{F} \mid \exists \mathcal{E}_0, \ldots, \mathcal{E}_k \ (k \geq 0)$ s.t. $\mathcal{E}_0 = \mathcal{E}$, $\mathcal{E}_k = \mathcal{F}$, and

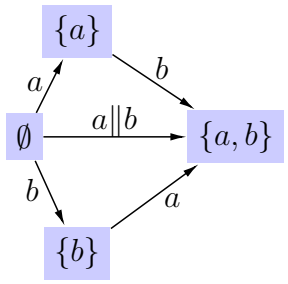

Fig. 4. The configuration transition system $TC_{step/pom}(SF(\mathcal{E}_3))$

$\mathcal{E}_i \xrightarrow{p}_\star \mathcal{E}_{i+1}\ (i < k)\}$.

Example 6. *Consider* $\widetilde{\mathcal{E}}_1 = SF(\mathcal{E}_1)$ *in Example 3 and* $\widetilde{\mathcal{E}}_5 = SF(\mathcal{E}_5)$ *in Example 2. The transition system* $TR_{step}(\widetilde{\mathcal{E}}_1)$ *is shown in Fig. 5, and the transition system* $TR_{pom}(\widetilde{\mathcal{E}}_5)$ *is depicted in Fig. 6.*

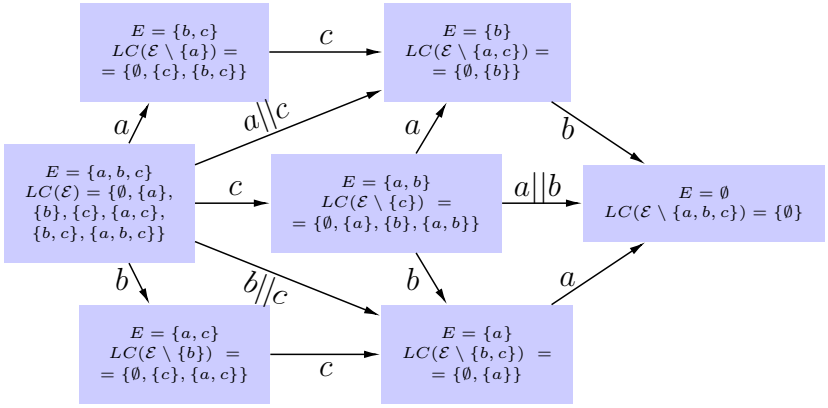

Fig. 5. The residual transition system $TR_{step}(SF(\mathcal{E}_1))$

We establish the relationships between states and transitions of $TC(\cdot)$ and $TR(\cdot)$ in both semantics.

Proposition 3. *Given a rooted RC-structure* $\mathcal{E}$ *in standard form and* $\star \in \{pom, step\}$,

(i) for any $X \in Conf_\star(\mathcal{E})$, $\mathcal{E} \setminus X \in Reach_\star(\mathcal{E})$;

(ii) for any $\mathcal{E}' \in Reach_\star(\mathcal{E})$, *there exists* $X \in Conf_\star(\mathcal{E})$ *such that* $\mathcal{E}' = \mathcal{E} \setminus X$;

(iii) for any $X', X'' \in Conf_\star(\mathcal{E})$, *if* $X' \xrightarrow{p}_\star X''$ *in* $TC_\star(\mathcal{E})$, *then* $\mathcal{E} \setminus X' \xrightarrow{p'}_\star \mathcal{E} \setminus X''$ *in* $TR_\star(\mathcal{E})$, *moreover,* $p = p'$ *if either* $\star = step$ *or* $\star = pom$ *and* $\mathcal{E}$ *is the standard form of a singular RC-structure;*

(iv) for any $\mathcal{E}', \mathcal{E}'' \in Reach_\star(\mathcal{E})$, *if* $\mathcal{E}' \xrightarrow{p}_\star \mathcal{E}''$ *in* $TR_\star(\mathcal{E})$, *then there are* $X', X'' \in Conf_\star(\mathcal{E})$ *such that* $\mathcal{E}' = \mathcal{E} \setminus X'$, $\mathcal{E}'' = \mathcal{E} \setminus X''$, *and* $X' \xrightarrow{p'}_\star X''$ *in* $TC_\star(\mathcal{E})$, *moreover,* $p = p'$ *if either* $\star = step$ *or* $\star = pom$ *and* $\mathcal{E}$ *is the standard form of a singular RC-structure.*

Proof.

(i) Take an arbitrary $X \in Conf_\star(\mathcal{E})$. Due to Lemma 2(i), we have that $Conf_\star(\mathcal{E}) = Conf(\mathcal{E})$. Hence, $X \in Conf(\mathcal{E})$. This implies that $X = \{e_1, \ldots, e_n\}$ $(n \geq 0)$ such that for all $i \leq n$ and for all $Y \subseteq \{e_1, \ldots, e_i\}$, there is $Z \subseteq \{e_1, \ldots, e_{i-1}\}$ such that $Z \vdash Y$. Clearly, $X_i = \{e_1, \ldots, e_i\} \in Conf(\mathcal{E})$ and $X_i \to X_{i+1}$, for all $i \leq n-1$. Let $Y_i = \{e_i\}$ for all $i \leq n$. Check that $\mathcal{E} \setminus X_i = \mathcal{E} \setminus Y_1 \setminus \ldots \setminus Y_i$, for all $i \leq n$. We shall proceed by induction on n.

$n = 0$. Obvious.

$n = 1$. Then, $Y_1 = X_1 \in Conf(\mathcal{E})$ and$\mathcal{E} \setminus X_1 = \mathcal{E} \setminus Y_1$.

$n > 1$. By the induction hypothesis, we have $\mathcal{E} \setminus X_{n-1} = \mathcal{E} \setminus Y_1 \setminus \ldots \setminus Y_{n-1}$. Since $X_{n-1}, X_n \in Conf(\mathcal{E})$ and $X_{n-1} \to X_n$ in $\mathcal{E}$, we obtain $X_n \setminus X_{n-1} = Y_n \in Conf(\mathcal{E} \setminus X_{n-1})$, due to Proposition 2(i(a)). According to Proposition 2(ii), it holds that $\mathcal{E} \setminus X_{n-1} \setminus Y_n = \mathcal{E} \setminus (X_{n-1} \cup Y_n) = \mathcal{E} \setminus X_n$. Hence, $\mathcal{E} \setminus Y_1 \setminus \ldots \setminus Y_{n-1} \setminus Y_n = \mathcal{E} \setminus X_n$.

It is easy to see that $\mathcal{E} \rightharpoonup_\star \mathcal{E} \setminus X_1 \rightharpoonup_\star \ldots \rightharpoonup_\star \mathcal{E} \setminus X_n$. Thus, $\mathcal{E} \setminus X \in Reach_\star(\mathcal{E})$.

(ii) Take an arbitrary $\mathcal{E}' \in Reach_\star(\mathcal{E})$. This means that $\mathcal{E} = \mathcal{E}_0 \stackrel{p_1}{\rightharpoonup}_\star \mathcal{E}_1 \ldots \mathcal{E}_{n-1} \stackrel{p_n}{\rightharpoonup}_\star \mathcal{E}_n = \mathcal{E}'$ $(n \geq 0)$. By the definition of $\stackrel{p_{i+1}}{\rightharpoonup}_\star$, it holds that $\mathcal{E}_{i+1} = \mathcal{E}_i \setminus X_{i+1}$, for some $X_{i+1} \in Conf_\star(\mathcal{E}_i)$ and $p_{i+1} = l_\star^{\mathcal{E}_i}(X_{i+1})$. Due to Lemma 2(i), we have $Conf_\star(\mathcal{E}_i) = Conf(\mathcal{E}_i)$. Hence, $X_{i+1} \in Conf(\mathcal{E}_i)$. Verify that $Y_{i+1} = \bigcup_{j=1}^{i+1} X_j \in Conf(\mathcal{E})$ and $\mathcal{E}_{i+1} = \mathcal{E} \setminus Y_{i+1}$, for all $i < n$. We shall proceed by induction on n.

$n = 0$. Obvious.

$n = 1$. Then, $Y_1 = X_1 \in Conf(\mathcal{E})$ and $\mathcal{E}_1 = \mathcal{E}_0 \setminus X_1 = \mathcal{E} \setminus Y_1$.

$n > 1$. By the induction hypothesis, $Y_{n-1} = \bigcup_{j=1}^{n-1} X_j \in Conf(\mathcal{E})$ and $\mathcal{E}_{n-1} = \mathcal{E} \setminus Y_{n-1}$. Check that $Y_n = \bigcup_{j=1}^{n} X_j \in Conf(\mathcal{E})$ and $\mathcal{E}_n = \mathcal{E} \setminus Y_n$. As $\mathcal{E}_n = \mathcal{E}_{n-1} \setminus X_n$, it holds that $\mathcal{E}_n = (\mathcal{E} \setminus Y_{n-1}) \setminus X_n$. According to Proposition 2(ii), we have that $Y_{n-1} \cup X_n = Y_n \in Conf(\mathcal{E})$ and $\mathcal{E}_n = \mathcal{E} \setminus (Y_{n-1} \cup X_n) = \mathcal{E} \setminus Y_n$. Thus, $\mathcal{E}' = \mathcal{E} \setminus Y_n$. Moreover, it is true that $Y_n \in Conf_\star(\mathcal{E})$, due to Lemma 2(i).

(iii) Take arbitrary $X', X'' \in Conf_\star(\mathcal{E})$ such that $X' \stackrel{p}{\rightharpoonup}_\star X''$ in $TC_\star(\mathcal{E})$. Then, we have that $X' \to_\star X''$ in $\mathcal{E}$ and $p = l_\star^{\mathcal{E}}(X'' \setminus X')$. This means that $X' \to X''$. As $\mathcal{E}$ is an *RC*-structure in standard form, we obtain that $X'' \setminus X' \in Conf(\mathcal{E} \setminus X')$, by Propositions 2(i(a)). Moreover, since $X' \to_\star X''$ in $\mathcal{E}$, we may conclude that $\emptyset \to_\star X'' \setminus X'$ in $\mathcal{E} \setminus X'$, by the construction of $\mathcal{E} \setminus X'$. Next, due to Propositions 2(ii), it holds that $\mathcal{E} \setminus X'' = (\mathcal{E} \setminus X') \setminus (X'' \setminus X')$. Then, $\mathcal{E} \setminus X' \stackrel{p'}{\rightharpoonup}_\star \mathcal{E} \setminus X''$ in $TR_\star(\mathcal{E})$, where $p' = l_\star^{\mathcal{E} \setminus X'}(X'' \setminus X')$. Moreover, if $\star = step$, we get that $p = l_{step}^{\mathcal{E}}(X'' \setminus X') = l_{step}^{\mathcal{E} \setminus X'}(X'' \setminus X') = p'$. If $\star = pom$ and $\mathcal{E}$ is the standard form of a

singular RC-structure, we have $\prec^{\mathcal{E}\setminus X'}_{X''\setminus X'} = \prec^{\mathcal{E}}_{X''} \cap (X''\setminus X' \times X''\setminus X')$, by Proposition 2(i(b)). Hence, $p = l^{\mathcal{E}}_{pom}(X''\setminus X') = l^{\mathcal{E}\setminus X'}_{pom}(X''\setminus X') = p'$.

(iv) Take arbitrary $\mathcal{E}', \mathcal{E}'' \in Reach_\star(\mathcal{E})$ such that $\mathcal{E}' \xrightarrow{p}_\star \mathcal{E}''$ in $TR_\star(\mathcal{E})$. Due to item (ii), there is $X' \in Conf_\star(\mathcal{E})$ such that $\mathcal{E}' = \mathcal{E}\setminus X'$. According to the definition of $\xrightarrow{p}_\star$, there is $\widetilde{X}' \in Conf_\star(\mathcal{E}')$ such that $\mathcal{E}'' = \mathcal{E}'\setminus \widetilde{X}'$, $\emptyset \to_\star \widetilde{X}'$ in $\mathcal{E}'$ and $p = l^{\mathcal{E}'}_\star(\widetilde{X}')$. Then, $X'' = X' \cup \widetilde{X}' \in Conf_\star(\mathcal{E})$, $X' \to X''$, and $\mathcal{E}'' = \mathcal{E}\setminus X''$, due to Proposition 2(ii). As $\emptyset \to_\star \widetilde{X}'$ in $\mathcal{E}'$ and $X' \to X''$, we have that $X' \xrightarrow{p'}_\star X''$ in $TC_\star(\mathcal{E})$, where $p' = l^{\mathcal{E}}_\star(X''\setminus X')$. Moreover, if $\star = step$, we get that $p' = l^{\mathcal{E}}_{step}(X''\setminus X') = l^{\mathcal{E}\setminus X'}_{step}(X''\setminus X') = p$. If $\star = pom$ and $\mathcal{E}$ is the standard form of a singular RC-structure, it holds that $\prec^{\mathcal{E}\setminus X'}_{X''\setminus X'} = \prec^{\mathcal{E}}_{X''} \cap (X''\setminus X' \times X''\setminus X')$, by Proposition 2(i(b)). Hence, $p' = l^{\mathcal{E}}_{pom}(X''\setminus X') = l^{\mathcal{E}\setminus X'}_{pom}(X''\setminus X') = l^{\mathcal{E}'}_{pom}(\widetilde{X}') = p$. □

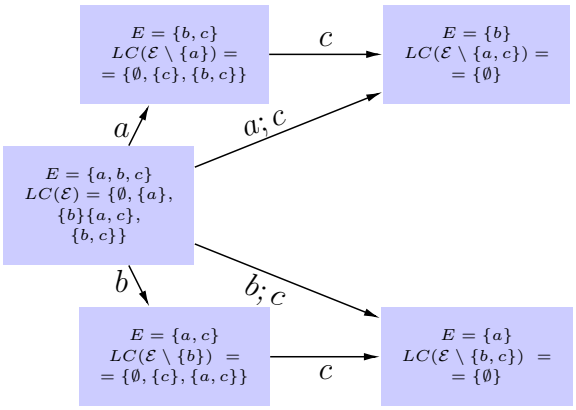

Fig. 6. The residual transition system $TR_{pom}(SF(\mathcal{E}_5))$

Finally, we formulate the main results of the paper.

Theorem 1. *Given a rooted RC-structure $\mathcal{E}$ and $\star \in \{step, pom\}$,*

(i) $TC_{step}(\mathcal{E})$ and $TR_{step}(SF(\mathcal{E}))$ are isomorphic, if $\mathcal{E}$ is a pure RC-structure;

(ii) $TC_{pom}(\mathcal{E})$ and $TR_{pom}(SF(\mathcal{E}))$ are isomorphic, if $\mathcal{E}$ is a pure, singular and locally conjunctive RC-structure.

Proof. Let $\star \in \{step, pom\}$. First, define a mapping $g : Conf_\star(SF(\mathcal{E})) \to Reach_\star(SF(\mathcal{E}))$ as follows: $g(X) = SF(\mathcal{E})\setminus X$ for all $X \in Conf_\star(SF(\mathcal{E}))$. Since $\mathcal{E}$ is a rooted RC-structure, it holds that $SF(\mathcal{E})$ is a rooted RC-structure, by Proposition 1(i). Then, $g(X)$ is well-defined, due to Proposition 3(i).

Since $SF(\mathcal{E})$ is a rooted RC-structure, it is clear that $\emptyset \in Conf(SF(\mathcal{E}))$. Thanks to Lemma 2(i), we get that $Conf_\star(SF(\mathcal{E})) = Conf(SF(\mathcal{E}))$. Hence, $g(\emptyset) = SF(\mathcal{E})\setminus\emptyset = SF(\mathcal{E})$.

Check that g is a bijective mapping. Assume $g(X) = g(X')$, for some $X, X' \in Conf_\star(SF(\mathcal{E}))$.

This means that $SF(\mathcal{E}) \setminus X = SF(\mathcal{E}) \setminus X'$. By Definition 3 and the construction of $SF(\mathcal{E})$, we get $E \setminus X = E \setminus X'$. Since $X, X' \subseteq E$, we have $X = X'$. So, g is an injective mapping.

Take an arbitrary $\mathcal{E}' \in Reach_\star(SF(\mathcal{E}))$. Due to Proposition 3(ii), we get $\mathcal{E}' = SF(\mathcal{E}) \setminus X$, for some $X \in Conf_\star(SF(\mathcal{E}))$. Hence, g is a surjective mapping.

(i) According to Propositions 3(iii) and (iv), and the fact that g is a bijective mapping, we have that $X \xrightarrow{p}_{step} X'$ in $TC_{step}(SF(\mathcal{E}))$ iff $g(X) \xrightarrow{p}_{step} g(X')$ in $TR_{step}(SF(\mathcal{E}))$. So, g is indeed an isomorphism between $TC_{step}(SF(\mathcal{E}))$ and $TR_{step}(SF(\mathcal{E}))$. Due to Lemma 4 (i), we may conclude that $TC_{step}(\mathcal{E}) = TC_{step}(SF(\mathcal{E}))$, because $\mathcal{E}$ is a pure RC-structure. Thus, $TC_{step}(\mathcal{E})$ and $TR_{step}(SF(\mathcal{E}))$ are isomorphic.

(ii) Since $SF(\mathcal{E})$ is the standard form of a singular RC-structure and g is a bijective mapping, we obtain that $X \xrightarrow{p}_{pom} X'$ in $TC_{pom}(SF(\mathcal{E}))$ iff $g(X) \xrightarrow{p}_{pom} g(X')$ in $TR_{pom}(SF(\mathcal{E}))$, by Propositions 3(iii) and (iv). So, g is indeed an isomorphism. Since $\mathcal{E}$ is a pure, singular and locally conjunctive RC-structure, we get that $TC_{pom}(\mathcal{E}) = TC_{pom}(SF(\mathcal{E}))$, by Lemma 4 (ii). Thus, $TC_{pom}(\mathcal{E})$ and $TR_{pom}(SF(\mathcal{E}))$ are isomorphic. □

Example 7. *From example 1 we know that $\mathcal{E}_1$ is a rooted, pure, not singular, and locally conjunctive RC-structure, and $\mathcal{E}_5$ is a rooted, pure, singular and locally conjunctive RC-structure. It is easy to see that $TC_{step}(\mathcal{E}_1)$ shown in Fig. 1 and $TR_{step}(SF(\mathcal{E}_1))$ shown in Fig. 5 are isomorphic, and the transition system $TC_{pom}(\mathcal{E}_5)$ depicted in Fig. 2 and $TR_{step}(SF(\mathcal{E}_5))$ shown in Fig. 6 are isomorphic. However, this does not apply to $\mathcal{E}_1$ in partial order semantics because the structure is not singular. In Fig. 7 and 8, we see that $\emptyset \xrightarrow{a\|c} \{a, c\}$ and $\emptyset \xrightarrow{b\|c} \{b, c\}$ in $TC_{pom}(\mathcal{E}_1)$, and $\mathcal{E}_1 \xrightarrow{c;a} \mathcal{E}_1 \setminus \{a, c\}$ and $\emptyset \xrightarrow{c;b} \mathcal{E}_1 \setminus \{b, c\}$ in $TR_{pom}(SF(\mathcal{E}_1))$.*

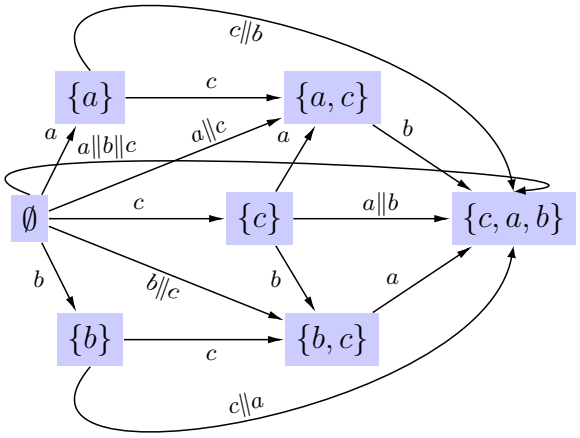

Fig. 7. The configuration transition system $TC_{pom}(\mathcal{E}_1)$

5. Concluding Remarks

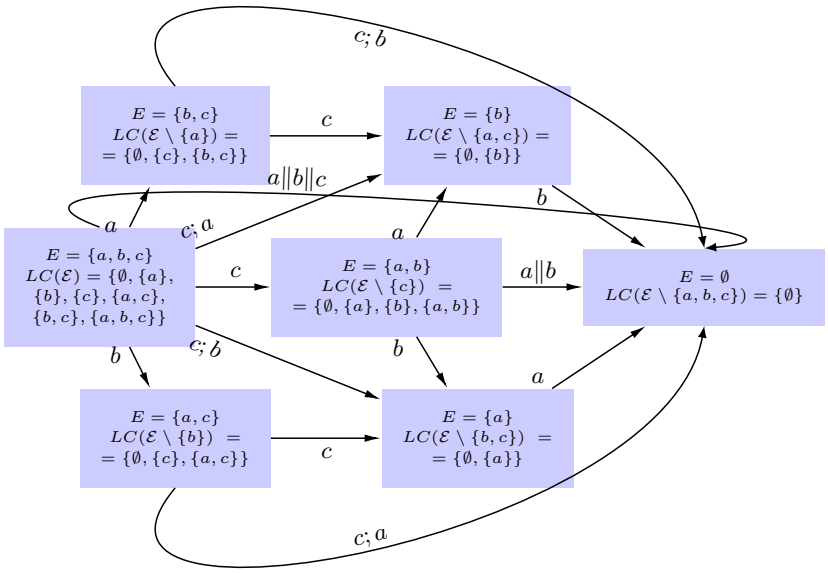

Fig. 8. The residual transition system $TR_{pom}(SF(\mathcal{E}_1))$

In this paper, we investigated two different ways of providing various transition system semantics for *RC*-structures which are very expressive among the event-oriented models known in the literature. First, we have developed the notion of partial orders within the configurations of *RC*-structures and discovered their subclasses in which partial orders are preserved in the standard form of the models and under the removal operator. Second, we have revealed closed relationships between configurations in the original *RC*-structure, in its standard form and in its residuals, under some conditions. Third, we have formulated properties of the model under consideration, which guarantee the coincidence of configuration transition systems obtained from the *RC*-structure and its standard form. As our main result, we have demonstrated how transition systems based on configurations and residuals of *RC*-structures, presented not necessary in standard form, are related in the context of partial order multiset and step semantics.

Work is currently underway to extend our approach to other event-oriented models (e.g., to precursor [11], probabilistic [28], and local [18] event structures, and also to event structures with dynamic causality [1]), and it gave preliminary results. Another future direction of our research is to extend our results in comparing of the two types of transition systems studied to the multiset case of transition relation and to the non-pure case of *RC*-structures.

References

1. Arbach Y., Karcher D., Peters K., Nestmann U. Dynamic causality in event structures // Logical Methods Comput. Sci. 2018. Vol. 14(1).

2. Armas-Cervantes A., Baldan B., Garcia-Banuelos L. Reduction of event structures under history preserving bisimulation // J. Log. Algebr. Meth. Program. 2016. Vol. 85(6). P. 1110-1130.
3. Baier C., Majster-Cederbaum M. The connection between event structure semantics and operational semantics for TCSP // Acta Informatica. 1994. Vol. 31.
4. Baldan P., Busi N., Corradini A., Pinna G.M. Domain and event structure semantics for Petri nets with read and inhibitor arcs // Theoret. Comput. Sci. 2004. Vol. 323(1-3). P. 129–189.
5. Baldan P., Corradini A., Montanari U. Contextual Petri Nets, Asymmetric Event Structures, and Processes // Information and Computation. 2001. Vol. 171(1). P. 1–49.
6. Bartoletti M., Cimoli T., Pinna G.M. Lending petri nets // Sci. Comput. Program. 2015. Vol. 112. P. 75–101.
7. Best E., Gribovskaya N., Virbitskaite I. Configuration- and residual-based transition systems for event structures with asymmetric conflict // Proc. SofSem 2017. Lecture Notes in Computer Science. 2018. Vol. 10877. P. 117-139.
8. Best E., Gribovskaya N., Virbitskaite I. From Event-Oriented Models to Transition Systems // Proc. Petri Nets 2018. Lecture Notes in Computer Science. 2017. Vol. 10139. P. 132–146.
9. Boudol G. Flow event structures and flow nets // Lecture Notes in Computer Science. 1990. Vol. 469. P. 62–95.
10. Crafa S., Varacca D., Yoshid N. Event structure semantics of parallel extrusion in the pi-calculus // Lecture Notes in Computer Science. 2012. Vol. 7213. P. 225–239.
11. Fecher, H., Majster-Cederbaum, M.: Event structures for arbitrary disruption. Fundamenta Informaticae. 2005. Vol. 68(1-2). P. 103–130.
12. van Glabbeek R.J. On the expressiveness of higher dimensional automata // Theoretical Computer Science. 2006. Vol. 356(3). P. 265–290.
13. van Glabbeek R.J., Goltz U. Refinement of actions and equivalence notions for concurrent systems // Acta Informatica. 2001. Vol. 37. P. 229–327.
14. van Glabbeek R.J., Plotkin G.D. Event structures for resolvable conflict // Lecture Notes in Computer Science. 2004. Vol. 3153. P. 550–561.
15. van Glabbeek R.J., Plotkin G.D. Configuration structures, event structures and Petri nets // Theoretical Computer Science. 2009. Vol. 410(41). P. 4111-4159.
16. van Glabbeek R.J., Vaandrager F.W. Bundle event structures and CCSP // Lecture Notes in Computer Science. 2003. Vol. 2761. P. 57–71.
17. Gribovskaya N., Virbitskaite I. Preserving Behavior in Transition Systems from Event Structure Models // CS&P 2018: Proceedings of the 27th International Workshop on Concurrency, Specification and Programming, Berlin, Germany, September 24–26, 2018. CEUR Workshop Proceedings 2240. CEUR-WS.org 2018.
18. Hoogers P.W., Kleijn H.C.M., Thiagarajan P.S. An event structure semantics for general Petri nets // Theoretical Computer Science. 1996. Vol. 153. P. 129–170.
19. Katoen J.-P. Quantitative and qualitative extensions of event structures: PhD Thesis, Twente University, 1996.

20. Katoen J.-P., Langerak R., Latella D. Modeling systems by probabilistic process algebra: an event structures approach // IFIP Transactions. 1993. Vol. C-2. P. 253–268.
21. Langerak R. Bundle event structures: a non-interleaving semantics for LOTOS: Formal Description Techniques V. IFIP Transactions. 1993. Vol. C-10. P. 331–346.
22. Loogen R., Goltz U. Modelling nondeterministic concurrent processes with event structures // Fundamenta Informatica. 1991. Vol. 14(1). P. 39–74.
23. Majster-Cederbaum M., Roggenbach M. Transition systems from event structures revisited // Information Processing Letters. 1998. Vol. 67(3). P. 119-124.
24. Nielsen M., Plotkin G., Winskel G. Petri nets, event structures and domains // Theoretical Computer Science. 1981. Vol. 13(1). P. 85–08.
25. Nielsen M., Thiagarajan P.S. Regular event structures and finite Petri nets: the conflict-free case // Lecture Notes in Computer Science. 2002. Vol. 2360. P. 335–351.
26. Winskel G. Event structures // Lecture Notes in Computer Science. 1986. Vol. 255. P. 325–392.
27. Winskel G. Introduction to event structures // Lecture Notes in Computer Science. 1989. Vol. 354. P. 364–397.
28. Winskel G. Distributed probabilistic and quantum strategies // Electronic Notes in Theoretical Computer Science. 2013. Vol. 298. P. 403–425.

УДК 004.822:004.89

Методика разработки лексико-семантических паттернов для извлечения терминологии научной предметной области

Кононенко И.С. (Институт систем информатики СО РАН),
Сидорова Е.А. (Институт систем информатики СО РАН)

В статье описывается подход к автоматизации извлечения терминологии для пополнения онтологии научной предметной области из текстов на русском языке. Применимость методов автоматического пополнения онтологии из текстов на естественном языке зависит от характеристик корпуса текстов и используемого языка. Специфика входного языка, характеризующегося сильной флективностью и свободным порядком слов, и отсутствие большого корпуса текстов приводят к выбору лингвистического подхода, базирующегося на использовании лексико-семантических паттернов. К особенностям предлагаемой методики извлечения информации относятся а) автоматическое пополнение предметного словаря на основе онтологии и корпуса текстов и разметка его с помощью системы семантических признаков; б) определение небольшого набора исходных структурных мета-паттернов, устанавливающих концептуальные контексты извлечения онтологической информации; в) автоматическое порождение по набору структурных мета-паттернов множества лексико-семантических паттернов, определяющих лексические, семантические и синтаксические свойства контекстов извлечения.

Ключевые слова: *построение онтологии, лексико-семантический паттерн, извлечение терминов, генерация паттернов.*

1. Введение

В настоящее время для формализации и систематизации знаний и данных в научных предметных областях (НПО) активно используются онтологии, которые позволяют описать научную дисциплину или область научных знаний во всех ее аспектах, включая характерные объекты и предметы исследования, применяемые научные методы, выполняемые проекты и полученные результаты [3]. Процесс разработки такой онтологии состоит из нескольких этапов, главными из которых являются построение терминологической части онтологии,

предполагающее создание таксономий понятий и отношений и описание их свойств, и пополнение онтологии, т.е. добавление в нее экземпляров понятий и отношений. Если первый этап задает скелет онтологии, то второй этап наполняет ее содержанием.

Чтобы получить онтологию, которая бы достаточно полно описывала НПО, требуется обработать огромное количество научных публикаций и информационных ресурсов, содержащих сведения из моделируемой области. Для облегчения и ускорения этого процесса разрабатываются методы автоматического пополнения онтологии на основе текстов на естественном языке [9, 13] и web-документов [6, 7]. Для автоматической обработки текстов используются подходы на основе кластеризации, в которых применяются широко известные кластерные и статистические методы, и подходы на основе шаблонов, в которых используются лингвистические шаблоны. Однако первый подход для хорошей работы требует наличия больших корпусов текстов, поэтому более распространены методы на основе лингвистических шаблонов.

У истоков лингвистического подхода стоит предложенная в работе [10] идея о возможности автоматизации построения семантических связей на основе диагностических контекстов, представленных в виде лексико-синтаксических шаблонов. Данный метод, известный как шаблоны Херст (Hearst patterns), предназначен для обработки неструктурированных англоязычных текстов. Он широко применялся для извлечения родовидовых отношений и предполагал извлечение из коллекции документов упорядоченных пар слов, соответствующих множеству заранее составленных шаблонов. Подход M. Hearst использовался и совершенствовался многими другими исследователями, а также применялся для других языков.

В ряде работ предлагается формальный аппарат для записи лексических и лексико-синтаксических шаблонов. Так, в работе [5] формулируется XML-схема языка для формализации лексико-синтаксических шаблонов, используемых для пополнения онтологий. Система Alex [2] и являющийся ее развитием инструмент DigLex [4] предоставляют довольно гибкие средства описания слов и словосочетаний в виде шаблонов, которые используются затем для автоматического распознавания этих единиц в тексте. Они расширяют возможности традиционных лексикографических систем: язык описания шаблонов поддерживает использование альтернатив, ссылки на шаблоны, повторители, условия на контекст, дистантный контекст и т.п. Язык позволяет записывать правила не только для распознавания текстовых объектов, но и для определения их лексических и семантических атрибутов. Однако в языках Alex и DigLex нет встроенных средств для указания грамматических признаков распознаваемых лексических единиц и грамматического

согласования нескольких единиц, необходимых для однозначного выделения языковых конструкций (например, именных групп). Этого последнего недостатка лишен предложенный в работе [1] язык LSPL, позволяющий задавать грамматические свойства входящих в него элементов.

Исследования, решающие задачу автоматического или полуавтоматического пополнения онтологии, опираются на паттерны (шаблоны), которые отображают языковые структуры, встречающиеся в текстах, в соответствующие элементы онтологии (понятия, отношения, экземпляры понятий и отношений). Это лексико-синтаксические паттерны, которые используют лексические представления и синтаксическую информацию [12, 14] или лексико-семантические паттерны, которые в процессе извлечения объединяют лексические представления с синтаксической и семантической информацией [11, 15].

В статье описывается подход к автоматизации пополнения онтологий НПО, базирующийся на использовании лексико-семантических паттернов (ЛСП) как расширения лексико-синтаксических паттернов онтологического проектирования. Особенностью данного подхода является то, что применяемые в нем ЛСП автоматически строятся на основе паттернов онтологического проектирования (паттернов ОП) других типов [8], входящих в систему автоматизации разработки онтологий на основе разнородных паттернов онтологического проектирования [16]. Еще одной отличительной чертой описываемого подхода является ориентация на русский язык, который, как и многие другие славянские языки, является сильно флективным языком со свободным порядком слов, что предполагает акцент на вариативности способов представления элементов онтологии в тексте и поиск методик генерации альтернативных средств выражения для максимизации полноты извлечения.

2. Особенности предметного словаря для пополнения онтологий

Для автоматического пополнения онтологии с помощью ЛСП требуется обеспечить извлечение из текста специфических терминов данной НПО. При этом для извлечения заранее известных терминов используются словари универсальной и предметной лексики, а для извлечения новых терминов (в частности, наименований объектов НПО или специфичных предикатных слов) – специализированные терминологические паттерны (Т-ЛСП) .

Предметный словарь – это объем лексики, организованной по семантическому принципу с отражением определенного набора базовых формальных отношений (см. Рис. 1). В словарной статье хранится вся необходимая информация как для извлечения термина из

текста, так и для поддержки последующих этапов анализа текста. Каждый термин, найденный в тексте с помощью предметного словаря, снабжается морфологической и семантической информацией, которая в дальнейшем используется в процессе применения ЛСП.

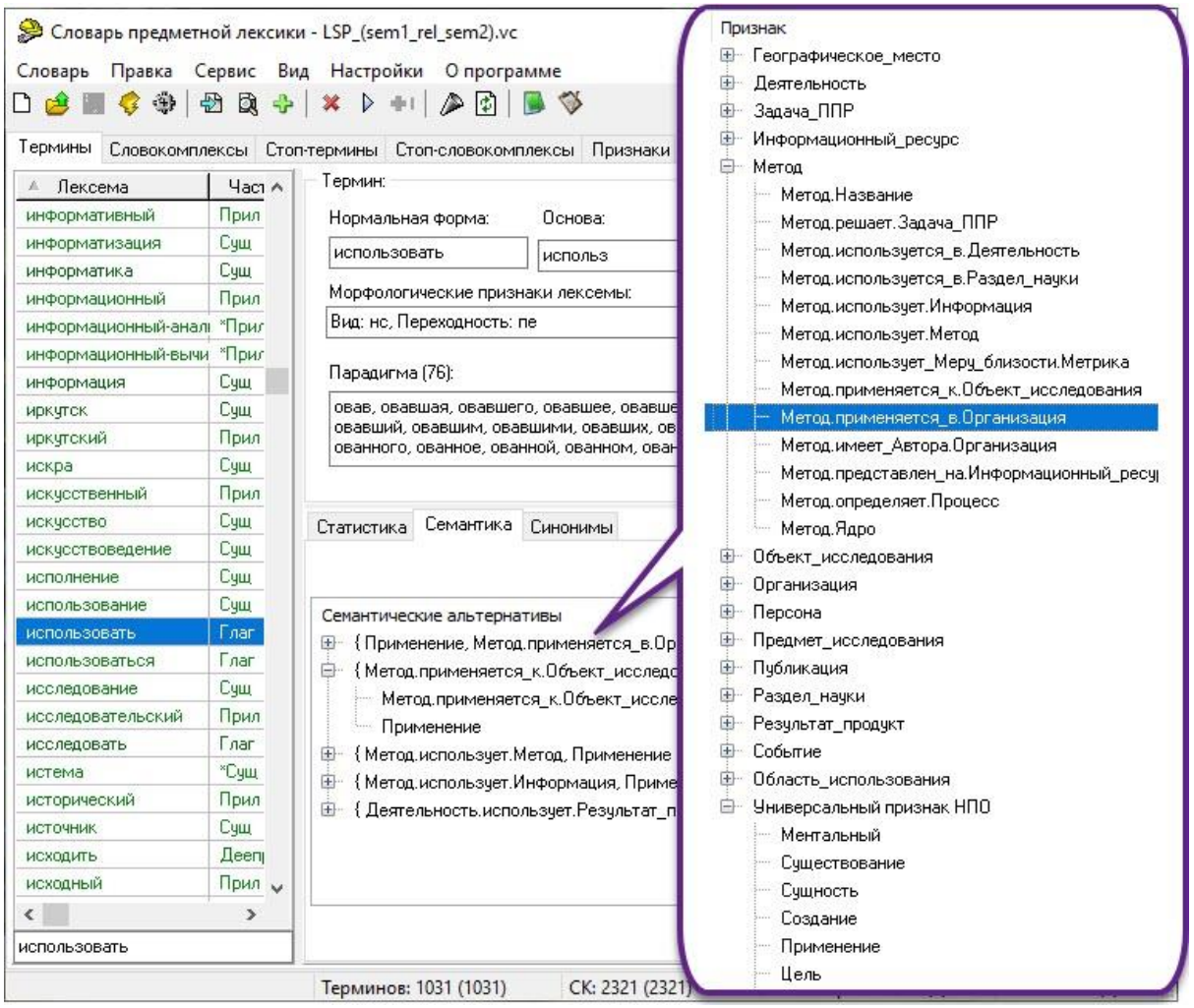

Рис. 1. Предметный словарь.

Снабжение терминов ПО лексико-семантическими характеристиками опирается на универсальную иерархию классов терминов, построенную вручную для научной области в целом, и иерархию предметных лексико-семантических классов, генерируемую автоматически на основе структуры заданной онтологии.

Для автоматической генерации словаря создана методика автоматического формирования системы лексико-семантических характеристик на основе имен онтологии. Предметный словарь создается как расширение словаря общенаучной лексики и включает две независимые иерархии лексико-семантических классов: универсальная иерархия

(*Универсальный признак НПО*) и предметно-ориентированная иерархия, создаваемая на основе онтологии НПО.

2.1. Универсальная иерархия признаков словаря

Универсальная иерархия включает признаки для разметки следующих групп терминов:

a) названия основных сущностей научных текстов, такие как *ученый, проект, организация, научный результат*;

b) названия типовых объектов (например, разделов науки – *информатика, физика*);

c) предикаты, используемые для описания (*Существование*) или связывания (*Применение*) сущностей; предикаты представлены, как правило, глаголами, включая личные, причастные и деепричастные формы, глагольными группами с зависимым инфинитивом (*использовать - решить использовать*), глаголами с зависимым предлогом (*применять к*), реализуемым как N-грамма, субстантивированными отглагольными именами (*использование*), существительными (*название*), ролевой лексикой (*разработчик, автор*), устойчивыми лексическими оборотами, реализуемыми как N-грамма (*под названием)*;

d) вспомогательная лексика (*Вспом*), включая связочные (*быть*, *являться*), фазовые (*начать*), модальные (*решить*, *позволять*) глаголы, семантически пустые глаголы типа лексических функций (*получать*, *давать*), а также слова и лексические конструкции, необходимые для представления универсальных категорий, таких как фамилии и имена людей, дат, типовых сокращений (*км, д.н., NLP, ПО, ОАО*) и т.п.

2.2. Генерация лексико-семантической иерархии словаря

Предметно-ориентированная иерархия лексико-семантических классов включает онтологически обусловленные признаки, названия которых формируются на основе следующих наименований.

- Имена классов паттерна(ов) содержания и соответствующие имена классов наследников, представленных в рассматриваемой онтологии НПО:

  имя_класса

  для терминов, являющихся именами классов, например: *Математический метод, Нормирование, Метод визуализации*;

  имя_класса.Ядро

  для слов, являющихся вершинами имен классов, например, слово *метод* снабжается признаком *Метод исследования.Ядро*

- Имена признаков-атрибутов класса, формируемых по шаблонам:

имя_класса.имя_признака

для значений атрибутов, например: *Метод.Название, Персона.Фамилия, Организация.Дата_основания*;

имя_класса.имя_признака.type

для названий самих атрибутов, например: *Метод.Название.type, Метод.Описание.type, Организация.Дата_основания.type*;

- Имена признаков-отношений между классами, формируемых по шаблону:

  имя_класса.имя_отношения.имя_класса

  например: *Метод.применяется_к.Объект_исследования,*

  Организация.расположена_в.Географическое_место

Все термины словаря универсальной научной лексики размечены признаками из универсальной иерархии. Новые термины из конкретной научной ПО размечаются с помощью признаков из предметно-ориентированной иерархии.

Термины конкретной НПО, присутствующие в словаре универсальной научной лексики, получают синкретический признак с одновременно выраженным значением универсального и предметного лексико-семантического класса. Так, для глагола ‘*использовать*’ выделено пять синкретических признаков (см. Рис. 1), каждый из которых включает универсальный класс *Применение* в сочетании с онтологически обусловленным признаком:

Метод.применяется_в.Организация,

Метод.применяется_к.Объект_исследования,

Метод.использует.Метод,

Метод.использует.Информация,

Деятельность.использует.Результат_продукт

Лексико-семантические признаки используются при описании ЛСП (в аргументах и результате) как способ обращения к терминам ПО с определенной семантикой.

3. Терминологические лексико-семантические паттерны

Для пополнения онтологии для каждого ее класса, представляющего понятие моделируемой предметной области, строится набор лексико-семантических паттернов, описывающих различные способы представления соответствующей ему информации в научных текстах. В данной работе рассматриваются терминологические паттерны (Т-ЛСП), которые используются для извлечения из текстов новых терминов ПО, не заданных в словаре.

Каждый ЛСП реализует модель вида: <Arguments, Constraints, Results>, где Arguments – множество семантических аргументов факта, которым сопоставляются термины ПО, Constraints – семантические, синтаксические и/или позиционные условия на аргументы, а Results описывает результат применения ЛСП, которым может быть новый термин и его семантических класс. Таким образом, Т-ЛСП представляют собой лексико-семантические шаблоны, формируемые на основе опорных терминов/маркеров с указанием семантических и грамматических ограничений.

Для извлечения новых терминов предложены два типа паттернов.

1) Паттерны первого типа моделируют именную группу (ИГ), в вершине которой представлено существительное или именная группа – представитель класса онтологии, и позволяют извлекать имена индивидов, относящихся к данному классу.
2) Паттерны второго типа позволяют извлекать новые термины на основе контекста, в котором присутствуют индикаторы отношений (атрибутов), в качестве которых, как правило, выступают предикатные термины, сопоставленные названиям отношений или атрибутов класса онтологии. Т-ЛСП первой подгруппы данного типа представляют контексты, структурными составляющими которых являются смысловые компоненты: Субъект, Объект и Предикат. Т-ЛСП второй подгруппы данного типа представляют контексты, структурными составляющими которых являются смысловые компоненты: Объект, Атрибут, Значение.

Для того, чтобы обеспечить автоматическую генерацию паттернов разработан язык и предложена методика создания типовых паттернов (или мета-паттернов), в состав которых включаются переменные. Создание исходных структурных мета-паттернов для извлечения информации для конкретной онтологии осуществляется путем означивания переменных именами онтологических классов, атрибутов и отношений. Итоговые мета-паттерны, помимо семантической информации, содержат лексические маркеры, синтаксические и позиционно-структурные ограничения.

3.1. Извлечение наименований объектов на основе именных групп

Паттерны первого типа позволяют извлекать имена индивидов на основе центрального слова или термина с привлечением синтаксических правил сборки именных групп. Во множестве терминологических наименований наиболее частотны паттерны следующего вида (типовой пример):

[<Прил>*, X, [<Прил, рд>*, <Сущ, рд>]*] ⇒ X.Название (1)

Данный Т-ЛСП включает три аргумента: 1) цепочку прилагательных <Прил>* (значок * в рамках языка паттернов означает цепочку компонент произвольной длины включая нулевую), 2) термин, имеющий лексико-семантический класс X (здесь и далее под X понимается не только имя класса, но и центральное слово, отмечаемое лексико-семантическим признаком *имя_класса.Ядро*) и 3) именную группу в родительном падеже, собранную по вложенному подшаблону вида [<Прил, рд>*, <Сущ, рд>]*; ограничениями здесь являются: a) указание семантического класса для 2-го аргумента, b) указание падежа для вложенного Т-ЛСП, используемого в качестве 3-го аргумента; результат Т-ЛСП определяет лексико-семантический признак *X.Название* для всех терминов, извлекаемых с помощью данного Т-ЛСП.

Подшаблон [<Прил, рд>*, <Сущ, рд>]* дает возможность учесть согласование *Прил* и *Сущ* по падежу для зависимой группы существительного в родительном падеже в рамках именных групп. Это позволяет избежать извлечения некорректных индивидов: **соответствующие этим проектам агрегированные оценки, *задачи принятия решений аксиоматические теории рационального поведения, *модели различные варианты.* К сожалению, язык шаблонов в его текущей версии не позволяет в общем случае учесть согласование существительного и зависимого прилагательного.

Данный типовой паттерн позволяет генерировать конкретные Т-ЛСП путем подстановки в качестве X имён классов онтологии. Например, для извлечения названия метода может быть автоматически сгенерирован следующий паттерн:

[<Прил>*, Метод, [<Прил, рд>*, <Сущ, рд>]*] ⇒ Метод.Название

Этот паттерн позволяет извлечь такие термины: *метод опорных векторов, метод анкетного опроса, метод медиан рангов, метод нейронных сетей, метод анализа иерархий Саати, метод интервью, метод самооценки*.

3.2. Извлечение наименований объектов на основе индикаторов отношений

Вторая группа паттернов формируется на основе индикаторов отношений, в качестве которых, выступают термины из словаря, сопоставленные названиям отношений онтологии.

Исходные мета-шаблоны данного типа описываются в соответствии со следующим принципом. В структуре шаблона выделяются два известных компонента: упоминание отношения и одного из аргументов данного отношения (указываются семантические классы

терминов и их грамматические признаки), а также третий – неизвестный компонент, который и требуется извлечь (помечается переменной вида ***$t***).

[X.REF, X.Rel.Y, $t<ИГ>] ⇒ Y.Название (2)

и симметричное

[$t<ИГ>, X.Rel.Y, Y.REF] ⇒ X.Название

Отношение Rel связывает объекты классов X и Y (в онтологии X.Rel - Object Property).

При генерации учитываются следующие особенности конструкций.

a) Третий компонент представлен именной группой, все ИГ извлекаются по шаблонам, аналогичным шаблонам из п.1, в которых в качестве X указано *Сущ*.

b) Для известного объекта отношения заводится служебный шаблон, объединяющий три варианта: имя класса, название и ядро имени класса:

[X] [X.Название] [X.Ядро] => **X.REF**

c) В качестве имени отношения выступает глагольная группа (ГГ), которая может включать вспомогательный глагол в спрягаемой форме и значимый предикат в форме инфинитива:

ГГ = [<Глаг>] [Вспом<Глаг>, <Инф>]

d) В большинстве шаблонов данного типа будет использоваться разрыв (обозначаем в мета-паттернах ***gap***), ограничения для которого задаются следующим паттерном:

gap = [s/"."] [s/"!"] [s/"?"] [s/":"] [s/";"] [s/"("] [s/")"] [s/"/”"] [s/"/n"] [s/","]

Разработана типология шаблонов описываемой группы, которые строятся на базе приведенных исходных схем паттернов для представления всего разнообразия конструкций. ЛСП данной группы представлены в текстах преимущественно глагольными и – существенно реже – субстантивными конструкциями. Подробно рассмотрены паттерны с глагольными формами в рамках простого или сложного предложения.

Варианты возможных контекстов, описываемых паттернами, определяют следующие факторы: (1) синтаксические ограничения, т.е. грамматические классы и согласование грамматической информации, (2) позиционно-структурные ограничения, т.е. порядок следования компонент и принадлежность компонент одной/нескольким клаузам в рамках предложения.

В результирующих шаблонах учитываются различные варианты порядка слов и возможные разрывы между компонентами исходной типовой схемы. В качестве ограничений на элементы в разрывах используются отсутствие отрицаний, разделителей предложений и других знаков пунктуации.

3.2.1. Активные конструкции с переходным глаголом

Структурными составляющими трехчленной активной конструкции являются смысловые компоненты Субъект, Объект и Предикат. В активной конструкции:

- субъект личной формы глагола выражается именительным падежом
- объект выражается винительным падежом,
- предикат представлен глагольной формой, соответствующей действительному залогу.

Учитывая бинарность отношения, извлекаемая сущность может быть указана а) как переменная в позиции субъекта, если объект представлен явно (известным термином) или б) как переменная в позиции объекта, если известен термин в позиции субъекта. Такие симметричные конструкции имеются для каждого типа глагольных форм. При этом разрыв допускается только между известными (явно заданными) компонентами контекста. Ниже представлены варианты метапатернов для простого предложения (a-d, g-i) и сложного предложения с придаточным определительным (d-e). Предикат простого предложения или придаточного определительного выражен переходным личным глаголом или зависимым от вспомогательного слова инфинитивом в составе группы сказуемого:

ГГ1 = [<Глаг, пе>] [<Глаг, Вспом>, <Инф, пе>]

Кроме того, шаблоны покрывают причастные и деепричастные обороты в рамках простого предложения.

1) Личная форма глагола и инфинитив

a. [X.REF<им>, gap, X.Rel.Y<ГГ1>, $t<ИГ, вн>] ⇒ Y.Название

// методы моделирования выполняют сбор необходимых научных данных и их систематизацию

Симметричная конструкция:

[$t<ИГ,им>, X.Rel.Y<ГГ1>, gap, X.REF<вн>] ⇒ X.Название

b. [X.Rel.Y <ГГ1>, $t<ИГ, вн>, X.REF<им>] ⇒ Y.Название

// решает задачу определения объектов метод прямого исключения рекурсии;

c. [X.Rel.Y<ГГ1>, gap ,X.REF<им>, $t<ИГ, вн>] ⇒ Y.Название

// наилучшим образом обеспечивает такой метод доступ к информации о потребителе;

d. [$t<ИГ, вн>, X.Rel.Y<ГГ1>, gap, X.REF<им>] ⇒ Y.Название

// количественную оценку уникальности совпадающих признаков позволяет получить вероятностный метод;

e. [X.REF, gap , “,”, ‘который’<им>, gap, X.Rel.Y<ГГ1>, $t<ИГ, вн>] ⇒ Y.Название

// прием цепных подстановок, который выполняет расчет величины влияния факторов в общем комплексе их воздействия на уровень совокупного финансового показателя;

f. [$t<ИГ>, “,”, ‘который’<вн>, gap, X.Rel.Y<ГГ1>, gap, X.REF<им>] ⇒ Y.Название

// проблеме снижения энергозатрат, которую позволяет решить цитратный метод.

2) Действительное причастие

g. [X.REF, “,”, gap, X.Rel.Y<Прич, пе, дст>, $t<ИГ, вн>] ⇒ Y.Название

// метод недоопределенных вычислений, эффективно решающий задачу удовлетворения ограничений в самой общей постановке;

h. [X.Rel.Y<Прич, пе, дст>, $t<ИГ, вн>, X.REF] ⇒ Y.Название

// устанавливающий взаимосвязи физических величин дидактический метод размерностей;

3) Деепричастие

i. [X.Rel.Y<Деепр, пе, дст>, $t<ИГ, вн>, “,”, X.REF<им>] ⇒ Y.Название

// выполняя поиск в подмножестве набора данных, алгоритм clarans.

3.2.2. Пассивные конструкции

Трехчленная пассивная конструкция зеркально отражает активную конструкцию. В пассивной конструкции:

- субъект выражается творительным падежом,
- объект личной формы глагола выражается именительным падежом,
- предикат выражен формой глагола, соответствующей страдательному залогу.

1) Возвратный глагол

Описываются конструкции с возвратными глаголами, имеющими невозвратные корреляты, то есть образованными от невозвратных с помощью постфикса *-ся* (помечаются в словаре признаком *Рефл*). Предикат простого предложения или придаточного

определительного выражен личным глаголом или зависимым от вспомогательного слова инфинитивом в составе группы сказуемого:

ГГ2 = [<Глаг, Рефл>] [<Глаг, Вспом>,<Инф, Рефл>]

При этом онтологическое отношение представлено в словаре невозвратным коррелятом и/или возвратным дериватом. Ниже приведены варианты метапатернов для простого предложения (a-b, d-e) и сложного предложения с придаточным определительным (c).

a. [$t<ИГ, им>, X.Rel.Y<ГГ2,>, gap, X.REF<тв>] ⇒ Y.Название

// *статическая задача решается известным методом нечетких когнитивных карт*;

b. [X.Rel.Y<ГГ2>, $t<ИГ, им>, X.REF<тв>] ⇒ Y.Название

// *выполняется поиск логических закономерностей методами интеллектуального анализа данных*;

c. [$t<ИГ>,“,”, ‘который’<им>,gap, X.Rel.Y<ГГ2>, gap, X.REF<тв>] ⇒ Y.Название

// *динамическая задача, которая также решается методом нкк*;

d. [$t <ИГ>, “,” , X.Rel.Y< Прич, Рефл, дст>, gap, X.REF<тв>] ⇒ Y.Название

// *ряд практически важных задач по определению равновесных и кинетических параметров, решавшихся этим методом*;

e. [X.Rel.Y < Прич, Рефл, дст>, gap , X.REF<тв>, $t<ИГ>] ⇒ Y.Название

// *решающаяся таким методом задача удовлетворения ограничений в самой общей постановке*;

2) Страдательное причастие

f. [$t<ИГ>, “,”, X.Rel.Y<Прич, пе, стр>, gap, X.REF<тв>] ⇒ Y.Название

// *проблемы классификации образов, решаемые применением нейронных сетей*;

g. [X.Rel.Y<Прич, пе, стр>, gap, X.REF, $t<ИГ>] ⇒ Y.Название

// *решенная предложенным методом задача аппроксимации функций*;

f. [$t<ИГ>, X.Rel.Y<Кратк_Прич, пе, стр>, gap, X.REF<тв>] ⇒ Y.Название

// *задача персонализации решена методами коллаборативной фильтрации.*

3.2.3. Глагольные конструкции с управляемым предлогом

В ряде случаев в наименованиях онтологических отношений представлено сильное или слабое предложное управление глагола. В онтологии эксперимента большинство таких

наименований составляют возвратные глаголы с предлогом *решается_на, решается_в, сводится_к, используется_в, применяется_в, применяется_к.* Возможно и краткое страдательное причастие с предлогом *представлен_на.* Ниже представлены варианты метапатернов для простого предложения (a, d-g) и сложного предложения с придаточным определительным (b-c). В них предикат выражен личным глаголом, причастием или кратким причастием, либо зависимым от вспомогательного слова инфинитивом или кратким причастием в составе группы сказуемого:

ГГЗ = [<Глаг/Кратк_Прич>] [<Глаг, Вспом>,<Инф>][<Глаг, Вспом>,<Кратк_Прич>]

Основной особенностью соответствующих конструкций является подъем прямого объекта в позицию субъекта (пассивизация) при отсутствии реального субъекта-агенса и ввод в рассмотрение косвенного объекта. В этом случае семантический класс косвенного объекта (например, *Организация, Информационный_Ресурс*) определяет возможность обратной конструкции, в которой косвенный объект поднимается в позицию подлежащего, т.е. связь с предикатом реализуется беспредложно (a).

В данной версии языка шаблонов возможно лишь обобщенно представить глагольное управление. Следует отметить возможность отрыва предлога от управляющей лексемы. При этом используемый в шаблонах разрыв по-прежнему характеризуется строгим отсутствием в нем пунктуации, что позволяет уменьшить шум при извлечении.

a. [X.REF<им>, gap, X.Rel.Y<ГГЗ>, gap, <Предл>, $t<ИГ>] ⇒ Y.Название

// *аналитические материалы представлены на сайте минэкономразвития России, подход к данной проблеме описан в работе л. флорианя*;

Обратная конструкция:

[$t<ИГ,им>, X.Rel.Y<ГГЗ>, gap, X.REF<вн>] ⇒ Y.Название

// *сайт минэкономразвития представляет аналитические материалы*;

b. [X.REF, “,”, ‘который’<им>, gap, X.Rel.Y<ГГЗ>, gap, <Предл>, $t<ИГ>] ⇒ Y.Название

// *метод, который применяется только к крахмалосодержащим растениям*;

c. [$t<ИГ>, “,”, <Предл>, ‘который’, gap, X.Rel.Y<ГГЗ>, gap, X.REF<им>] ⇒ Y.Название

// *этап изучения объекта системного исследования, на котором возникает задача формализации объекта*;

d. [Предл, $t<ИГ>, X.Rel.Y<ГГ3>, gap, X.REF<им>] ⇒ Y.Название

// *на этапе анализа данных применяется не только метод размерностей*;

e. [X.Rel.Y <ГГ3>, gap, X.REF<им>, gap, <Предл>, $t<ИГ>] ⇒ Y.Название

// *применяется описанный диахронический подход в ономасиологии*;

f. [X.Rel.Y<Прич>, gap , <Предл>, $t<ИГ, вн>, X.REF] ⇒ Y.Название

// *примененный в данном исследовании метод экспертных оценок;*

g. [X.REF, “,”, gap, X.Rel.Y<Прич>, gap , <Предл>, $t<ИГ> ⇒ Y.Название

// *Лоренцева ионизация, часто применяющаяся в физике ускорителей; методы квазиклассического приближения, применяемые в квантовой физике.*

3.3. Извлечение значений атрибутов

На основе типовых паттернов можно генерировать Т-ЛСП, предназначенные для извлечения значений атрибутов объектов класса X, которые в онтологии представлены свойством Datatype Property.

[X, X.A.type, $t<ИГ>] ⇒ X.A (3)

Во втором компоненте шаблона указано явно выраженное имя атрибута, а в третьем компоненте – извлекаемое свойство объекта ***$t,*** которое должно быть представлено именной группой. В случае ключевого атрибута *Название* это эквивалентно извлечению объекта. Результатом применения таких паттернов будет создание новых терминов и приписывание им семантического признака X.A.

В разделе 3.1 предложен подход к извлечению неизвестных базе знаний терминов на основе имен классов, представленных ИГ. Полезным дополнением этому является извлечение значений атрибута *X.Название* с учетом контекста, содержащего предикат класса *Именование* (*называть, именовать, название, под названием*) и явным образом говорящего о вводе в текст нового термина.

Соответствующие паттерны используют предварительно заданные шаблоны именной группы и глагольной лексемы, включающей личные формы глагола и причастия:

Г = [<Глаг>] [<Прич>] [<Кратк_Прич>]

gap1 обозначает возможный разрыв, не исключающий наличия запятой:

gap1 = [s/"."][s/"!"][s/"?"][s/":"][s/";"][s/"("][s/")"][s/"/””][s/"/n"]

Ниже представлены паттерны, выделенные при рассмотрении конструкций с терминами, представляющими наименования объектов различных классов (приведены примеры для классов X = *Метод*, *Научное_направление*, *Задача*, *Информационный_Ресурс*). Конструкции классифицируются в соответствии с грамматическим классом предиката наименования.

1) Глагольные конструкции

a. [X<им/вин>, gap1, Именование<Г> , gap, $t<ИГ, тв>] ⇒ X.Название

*// Применяемый в работе метод **называется** методом магнетрона; Описывается метод, **называемый** методом согласования для обращения соответствующих интегральных уравнений; Используются различные математические методы, **именуемые** в данном контексте экономико-математическими*

b. [‘предлагать’<Г>, gap, X<им/вин>, gap1, Именование<Г>, gap, $t<ИГ, тв>] ⇒ X.Название

*// По аналогии с известным расчетным статическим методом предлагаемый метод **назовем** расчетным динамическим; В данной работе предлагается метод, **называемый** «элитным отбором» или «элитной стратегией»; Предлагаемый метод **назовем** методом двух групп; В работе предлагается метод, **называемый** Voxel Cone Tracing (VCT); В качестве альтернативного способа предлагается метод, **называемый** криотерапией;*

Снятие ограничения на падеж (тв) расширяет класс распознаваемых конструкций*:*

*// В работе предлагается метод, **называемый** иерархическая редукция (hierarchical reduction).*

Приведенные паттерны позволяют учесть и достаточно частотные в научных статьях конструкции с местоименным анафорическим элементом:

c. [‘предлагать’<Г>, gap, X<им/вин>, gap1, Именование<Глаг>, gap, <Мест, вн>, $t<ИГ, тв>] ⇒ X.Название

*// Предлагаемый метод, **назовем** его «методом перекоса»; Предлагаемый метод, **назовем** его «оценочное взвешенное пересечение».*

2) Конструкции с существительным

d. [X, gap1, <Глаг, Вспом>, Именование<Сущ> , gap, $t<ИГ>] ⇒ X.Название

*// Имеется развитое направление исследований, **получившее название** математической экономики; Классическая задача исследования операций **получила название** обобщенной транспортной задачи; Один из подходов к многоэтапному оптимальному выбору в условиях*

*вероятностной неопределенности, который **носит название** дерева решений (decision tree); В рассказе и. одоевцевой, **носящем название** цитированного здесь стихотворения г. иванова «эпилог»; Явление, **получившее** в современных исследованиях **название** гипертекста; Проклятия у кадарцев **носят название** дерга; Вычислительное направление исследований в дальнейшем трансформировалось в новую методологию и технологию проведения научных исследований, которое **получило название** вычислительного эксперимента.*

3) Конструкции с лексическим оборотом *под названием*

e. [X, gap1, ‘под названием’, $t<ИГ>] ⇒ X.Название

*// kaufman и rousseeuw (1990) также предложили алгоритм, известный сейчас **под названием** clara (clustering large applications); после второй мировой войны стало развиваться научное направление **под названием** "исследование операций"; «принятие решений» — один из основополагающих терминов в научном направлении, известном **под названием** «исследование операций»; в основной статье в этом сборнике **под названием** "основы теории измерений", изложение шло на абстрактно-математическом уровне; Тянь чжан и др. предложили метод агломерационной иерархической кластеризации **под названием** birch; Ветви экономической теории, известной в официальных кругах **под названием** «статистика».*

Таким образом, из текста конкретного жанра с помощью характерных для данного жанра шаблонов извлекаются целевые понятия, зафиксированные в паттерне содержания.

5. Автоматическая генерация терминологических паттернов

Как было сказано выше, терминологические лексико-семантические паттерны автоматически строятся на основе онтологии паттернов онтологического проектирования, словаря общенаучной лексики и текущей версии онтологии НПО. На Рис.2 представлена схема взаимосвязей компонентов системы, на основе которых осуществляется генерация Т-ЛСП и пополнение словаря.

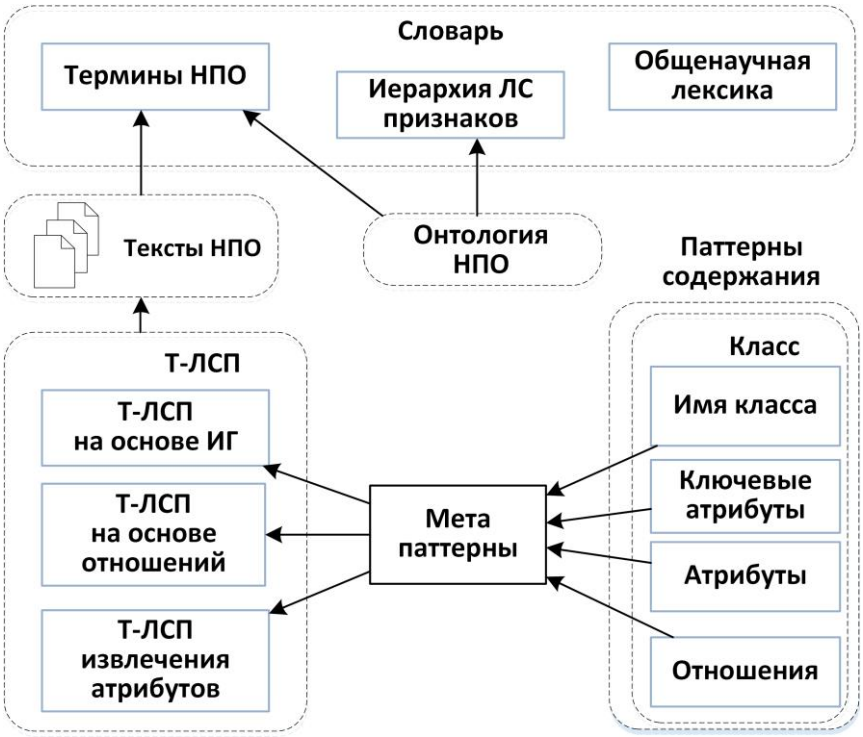

Рис.2. Схема взаимосвязей компонентов системы знаний для генерации Т-ЛСП.

Процесс генерации ЛСП начинается с создания и наполнения предметного словаря. Из онтологии и описания паттерна содержания извлекаются термины (лексемы и терминоподобные N-граммы) и формируются лексико-семантические классы по приведенной в п.1.2. методике. Все термины размечаются соответствующими семантическими признаками.

Отношение синонимии в явном виде отсутствует в словаре, однако, мы считаем квазисинонимами все термины, размеченные одним лексико-семантическим признаком. Поэтому для терминов с соотнесенными предметными и универсальными признаками можно выявить квазисинонимы по универсальному признаку (из словаря общенаучной лексики) и разметить их соответствующим предметным признаком.

На этапе генерации словаря используется корпус научных текстов, релевантных для рассматриваемой НПО, поскольку он является источником предметной лексики, употребляемой специалистами в данной НПО. Из корпуса можно извлечь не только типовые названия сущностей (классов сущностей), но и специфические индикаторы отношений или атрибутов.

На следующем этапе на основе анализа структуры паттерна содержания осуществляется означивание переменных в мета-паттернах и формируются Т-ЛСП. С помощью созданных Т-ЛСП анализируются тексты научного корпуса и словарь пополняется найденными новыми терминами.

При извлечении объектов необходима информация о ключевых атрибутах классов онтологии. В приведенных выше примерах в качестве такого атрибута выступает *Название*, однако у класса могут быть и другие ключевые атрибуты (например, *Фамилия* и *Имя* у *Персоны*) которые потребуют создания других типовых паттернов.

Таким образом, на основе онтологических компонентов знаний можно выделить необходимые для извлечения информации из текста и пополнения онтологии знания о языке предметной области и возможные способы языкового описания онтологических сущностей в текстах. Формализация этих знаний в виде системы ЛСП позволяет применить существующие технологии автоматической обработки текстов для решения поставленной задачи.

5. Анализ экспериментов

На основе предложенной методики проведен эксперимент по созданию Т-ЛСП и извлечению новых терминов для онтологии НПО «Поддержка принятия решений в слабоформализованных областях». Экспериментальная проверка была проведена для паттерна содержания *Метод*. На основе текущего состояния онтологии данной НПО и описания паттерна содержания *Метод* были автоматически созданы словарь предметной области, включающий 214 терминов, размеченных с помощью 21 лексико-семантического признака, и 34 Т-ЛСП для извлечения новых терминов (названий экземпляров классов и предикатных слов). Для генерации иерархии лексико-семантических признаков использовались метки (rdfs:label) атрибутов и отношений класса *Метод*, а также значения атрибутов для экземпляров этого и связанных с ним классов.

Для оценки качества полученных ЛСП использовался корпус, состоящий из научных публикаций на русском языке, общим объемом 53,9 тыс. токенов. С помощью Т-ЛСП, предназначенных для извлечения наименований индивидов класса *Метод*, найдено 111 вхождений и выделено 73 уникальных термина, которым был приписан лексико-семантический класс *Метод.Название*. Из них 70 терминов оказались новыми (отсутствовавшими в словаре, построенном по онтологии): *метод анкетного опроса, метод медиан рангов, метод нейронных сетей, метод анализа иерархий Саати, метод интервью, метод самооценки* и т.д. Аналогичные Т-ЛСП применялись для извлечения имен индивидов других классов, связанных с классом *Метод*. Так, для класса *Задача* было найдено 42 вхождения шаблона и выявлено 33 термина, из которых 30 являлись новыми: *задача когнитивного моделирования*, *задача порядковой классификации*, *задача целочисленного программирования* и т.д. Среди ошибочных результатов можно отметить термины с

указанием источника, строящихся по тому же синтаксическому правилу, например, *задача дипломной работы*.

Для извлечения новых предикатных терминов использовались Т-ЛСП, построенные по мета-паттернам, моделирующим ситуацию онтологической связи двух объектов. Так, для Т-ЛСП, служащего для извлечения предикатных терминов, относящихся к классу *Метод.решает.Задача*, в корпусе было найдено 22 вхождения и выявлены такие термины как: *поставили, позволяют построить, позволил выделить, ищут решения, позволяет решать* и т.п. Всего было найдено 38 новых предикатных терминов.

Общая точность работы Т-ЛСП была оценена тремя экспертами и составила в среднем 73,97%, при этом согласие между экспертами оценено в 81,74%.

Анализ материалов и результатов эксперимента выявил недочеты в описаниях контекстов извлечения, приводящих к ложно-негативным или ложно-позитивным результатам.

1) Использование наименований классов в качестве индикаторов имен соответствующих индивидов имеет ряд ограничений а) формирование имени индивида на основе имени класса покрывает лишь некоторое подмножество упоминаемых в текстах индивидов; б) далеко не всегда для конкретного класса в качестве индикатора, т.е. вершины именной группы, представляющей наименование индивида, используется модель с полным именем класса – чаще используется голая вершина или альтернативная модель, в которой имя класса подчиняет зависимые, не учтенные в модели именования индивидов 1 типа:

деятельность + *по* + Сущ vs. **деятельность* + Сущ,рд

2) Морфолого-синтаксическая информация используется неполно в силу специфики языка шаблонов: а) невозможно задать общие условия согласования морфологических характеристик существительного и зависимого прилагательного (можно лишь задать характеристики препозитивного прилагательного, если определены соответствующие характеристики существительного); б) невозможно задать общие условия соответствия морфологических характеристик подлежащего и сказуемого; в) невозможно учесть падежное управление индивидуальных глаголов с помощью общего ограничения (последние два ограничения можно учесть лишь частично для определенных конструкций, таких как активная конструкция для переходных глаголов и соответствующая пассивная конструкция на базе страдательных форм и возвратных глаголов).

3) Методика поверхностного (неполного) синтаксического анализа, реализуемого с помощью языка шаблонов, не всегда позволяет решить важную проблему границ покрытого текстового фрагмента, прежде всего, границ именной группы (что характерно для всей области извлечения информации).

4) Разрывный контекст, покрываемый ЛСП, представляет существенную проблему: так, необходимо исключить из контекстов не только лексическое отрицание, но и другие лексические и грамматические способы выражения ирреальности и проблематичной достоверности (сомнения): *не может, не позволяет, едва ли, сомнительно* и т.п.

5) Особенности исходной онтологии определяют формулировки генерируемых ЛСП: а) имеющиеся в онтологии орфографические или содержательные ошибки, такие как неверное задание значения атрибута (описание вместо названия) или отнесение индивида к классу (название организации в классе объектов исследования), определяют генерацию некорректных ЛСП; б) генерация ЛСП затруднена отсутствием стандартного языка представления онтологических сущностей и отношений. В качестве примеров можно привести именные группы сложной структуры (в том числе сочинительные конструкции) в названиях классов, использование нумерации в названии объектов.

Заключение

В статье представлен основанный на правилах лингвистический подход к извлечению из текстов информации для поддержки процесса автоматического пополнения онтологии на основе исходной онтологии, начального (универсального) словаря и корпуса текстовых документов. Принятый подход базируется на следующих методических принципах:

- автоматическое пополнение предметного словаря на основе онтологии и корпуса текстов и разметка его с помощью системы семантических признаков, также основанных на онтологии,
- определение небольшого набора исходных структурных мета-паттернов, устанавливающих концептуальные контексты извлечения онтологической информации,
- автоматическое порождение по набору структурных мета-паттернов множества лексико-семантических паттернов, определяющих лексические, семантические и синтаксические свойства контекстов извлечения.

В работе приведены основные паттерны для извлечения объектов (экземпляров классов онтологии) и их свойств с учетом вариативности и множественности языковых выражений, представляющих элементы онтологии в тексте на русском языке, отличающемся сильной флективностью и свободным порядком слов.

Данный подход апробируется при разработке и пополнении онтологий различных научных предметных областей («Поддержка принятия решений», «Поддержка решения вычислительно сложных задач на суперкомпьютерах»).

Список литературы

1. Большакова Е.И., Баева Н.В., Бордаченкова Е.А., Васильева Н.Э., Морозов С.С. Лексико-синтаксические шаблоны в задачах автоматической обработки текстов // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог’2007. М.: Издательский центр РГГУ. 2007. С. 70-75.
2. Жигалов В.А., Жигалов Д.В., Жуков А.А., Кононенко И.С., Соколова Е.Г., Толдова С.Ю. Система Alex как средство для многоцелевой автоматизированной обработки текстов // Труды международного семинара Диалог’2002 «Компьютерная лингвистика и интеллектуальные технологии». М.: Наука. 2002. Т.2. С. 192-208.
3. Загорулько Ю.А., Сидорова Е.А., Загорулько Г.Б., Ахмадеева И.Р., Серый А.С. Автоматизация разработки онтологий научных предметных областей на основе паттернов онтологического проектирования // Онтология проектирования. 2021. Т.11, №4(42). С. 500-520.
4. Ковалев А.И., Сидорова Е.А. Инструмент разработки предметных словарей на основе лексических шаблонов DigLex // Материалы Всероссийской конференции с международным участием «Знания – Онтологии – Теории» (ЗОНТ–2015), 6 - 8 октября 2015 г., Новосибирск. Новосибирск: Институт математики им. С.Л. Соболева СО РАН. 2015. Т. 1. С. 123–130.
5. Рабчевский Е.А. Автоматическое построение онтологий на основе лексико-синтаксических шаблонов для информационного поиска // Труды 11й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL’2009. Петрозаводск. 2009. С. 69–77.
6. Alani H., Kim S., Millard D.E., Weal M.J., Hall W., Lewis P.H., Shadbolt N.R. Automatic Ontology-Based Knowledge Extraction from Web Documents // IEEE Intelligent Systems 18(1), 2003. P.14–21.
7. Akhmadeeva I. R., Zagorulko Y. A., Mouromtsev D. I. Ontology-Based Information Extraction for Populating the Intelligent Scientific Internet Resources // Knowledge Engineering and Semantic Web: 7th International Conference, KESW 2016, Proceedings. / Ngomo A. C. N., Křemen P. (Eds.). Communications in Computer and Information Science. Springer International Publishing, 2016. Vol. 649. P. 119-128.
8. Blomqvist E., Hammar K., Presutti V. Engineering Ontologies with Patterns: The eXtreme Design Methodology // In: Hitzler P., Gangemi A., Janowicz K., Krisnadhi A., Presutti V. (eds.): Ontology Engineering with Ontology Design Patterns. Studies on the Semantic Web. Amsterdam, IOS Press, 2016. Vol. 25. P. 23–50.
9. Ganino G., Lembo D., Mecella M., Scafoglieri F. Ontology population for open-source intelligence: a GATE-based solution // Software: Practice and Experience. 2018. 48(12). P. 2302-2330.
10. Hearst M.A. Automatic Acquisition of Hyponyms from Large Text Corpora // Proceedings of the 14th International Conference on Computational Linguistics. 1992. P. 539–545.

11. Ijntema, W., Sangers, J., Hogenboom, F., Frasincar, F. A lexico-semantic pattern language for learning ontology instances from text // Journal of Web Semantics. 2012. Vol. 15. P. 37–50.
12. Maynard D, Funk A, Peters W. Using Lexico-Syntactic Ontology Design Patterns for Ontology Creation and Population // In: Proc. Workshop on Ontology Patterns (WOP 2009), collocated with the 8th Int. Semantic Web Conf. (ISWC-2009). CEUR Workshop Proceedings. 2009. Vol. 516. P. 39–52.
13. Petasis G, Karkaletsis V, Paliouras G, Krithara A, Zavitsanos E. Ontology Population and Enrichment: State of the Art. In: Paliouras, G., Spyropoulos, C.D., Tsatsaronis, G. (eds): Knowledge-Driven Multimedia Information Extraction and Ontology Evolution). Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011. Vol. 6050. P. 134–166.
14. Ruiz-Martínez M.J., Valencia-García R., Martínez-Béjar R., Hoffmann A. 2012 BioOntoVerb: A top level ontology based framework to populate biomedical ontologies from texts // Knowledge-Based Systems. 2012. Vol. 36. P. 68-80.
15. Saeeda L., Med M., Ledvinka M., Blaško M., Křemen P. Entity Linking and Lexico-Semantic Patterns for Ontology Learning // The Semantic Web (ESWC 2020). Lecture Notes in Computer Science. Springer, Cham. 2020. Vol. 12123. P. 138–153.
16. Zagorulko Yu. A., Zagorulko G.B., Borovikova O.I. Pattern-Based Methodology for Building the Ontologies of Scientific Subject Domains // In: H. Fujita and E. Herrera-Viedma (Eds.): New Trends in Intelligent Software Methodologies, Tools and Techniques. Proceedings of the 17th International Conference SoMeT_18. Series: Frontiers in Artificial Intelligence and Applications. Amsterdam: IOS Press, 2018. Vol. 303. P. 529-542.

УДК 004.827, 004.832.22, 004.832.25

Система программирования в ограничениях Nemo

Сидоров В.А. (Институт систем информатики СО РАН)

Метод недоопределённых вычислений является одним из подходов для решения задачи удовлетворения ограничений. На его основе разработан ряд программных систем, в том числе система программирования в ограничениях Nemo. В статье рассматриваются принципы, используемые для разработки системы Nemo, их реализация на практике и возможности, предоставляемые пользователю для решения задач.

Ключевые слова: *задача удовлетворения ограничений, метод недоопределённых вычислений, язык описания модели.*

1. Введение

Задача удовлетворения ограничений (**ЗУО**) является одним из способов формулировки и решения нестандартных и сложных вычислительных задач. Неформально ЗУО можно описать следующим образом:

- Решаемая задача состоит из набора переменных и набора ограничений.
- С каждой переменной связана собственная область определения (множество допустимых значений), которая может изменяться в процессе решения ЗУО.
- Ограничения связывают переменные и ограничивают множества их значений.
- Решением ЗУО является такой набор значений переменных, для которого удовлетворяются все ограничения.
- Целью решения ЗУО может быть как поиск одного решения (возможно, с заданными свойствами), так и поиск всех решений.

Впервые задача удовлетворения ограничений была сформулирована в начале 70х годов [13, 15].

В начале 80х годов А.С. Нариньяни был предложен *метод недоопределённых вычислений* [5, 8], который является одним из методов решения задачи удовлетворения ограничений. Метод обладает следующими особенностями:

- Значения переменных являются *недоопределёнными*, представленными в виде множества допустимых значений.
- Алгоритм вычислений фиксированный. На каждом шаге удовлетворяется (вычисляется) одно активное ограничение; выделяются переменные, значение

которых изменилось на данном шаге; активируются ограничения, связанные с данными переменными.

Отметим важную особенность метода недоопределённых вычислений — универсальность. Алгоритм не зависит от типа обрабатываемых данных; в частности он позволяет решать задачи, содержащие как непрерывные, так и дискретные численные значения переменных.

На основе метода недоопределённых вычислений было создано нескольких линеек программных систем, таких как математические вычислители UniCalc [1], экспертные системы Semp [3], мультиагентные системы ТАО [2], электронные таблицы FinPlan [17]. В статье рассматривается система программирования в ограничениях Nemo, относящаяся к семейству систем NeMo [4, 11], характерными особенностями которых является возможность работать с различными типами данных и специфицировать задачу на высоком уровне.

Система Nemo изначально проектировалась как промышленная система с сохранением широких возможностей по расширяемости и универсальности, характерных для семейства NeMo. Рассмотрим её свойства в следующем порядке:

- Вначале сформулируем общие принципы (и вытекающие из них свойства) используемые при разработке системы;
- Далее опишем реализацию указанных свойств в архитектуре системы;
- В конце опишем некоторые возможности предоставляемые пользователям системы для формулировки и решения различных типов задач.

2. Принципы создания системы Nemo

Предназначение системы Nemo определяется следующим образом:

1. Система предназначена для использования в роли универсального встраиваемого вычислителя. Главными характеристиками в этом случае являются **производительность** и **технологичность**.
2. Система предназначена для исследования различных задач удовлетворения ограничений и алгоритмов их решения. Ключевой характеристикой при этом является **универсальность** - возможность настраивать систему для решения различных классов задач.

На первый взгляд, эти характеристики противоречат друг другу, но это не так. Например, для достижения максимальной производительности требуется использовать различные алгоритмы для различных классов задач, что, в свою очередь, определяется степенью универсальности системы.

Для реализации этих характеристик был определен ряд свойств, которые использовались как при разработке архитектуры системы, так и были доступны далее на уровне пользователя:

1. **Универсальность** означает способность формулировать и решать максимально широкий список задач. На степень универсальности влияют следующие свойства системы Nemo:
   - Декларативное представление модели: описание модели содержит только описание задачи, а не алгоритм её решения. Это имеет ряд преимуществ и недостатков:
     - Упрощается формулировка задачи пользователем.
     - Порядок задания ограничений (описания задачи) не влияет на результат вычислений.
     - Декларативность ограничивает круг решаемых задач – система Nemo позволяет решать только статические (не изменяющиеся в процессе решения) задачи.
   - Расширяемость системы позволяет использовать дополнительные специализированные типы данных и ограничений, и, соответственно, настраивать систему для решения конкретных классов задач.
   - Поддержка объектно-ориентированного строения модели повышает уровень описания задачи, что упрощает формулировку сложных задач.
   - Использование концепции «недоопределённость» позволяет работать с неточными и не полностью определёнными данными.
   - Единый универсальный алгоритм решения задачи позволяет решать любые задачи удовлетворения ограничений.
2. Под **производительностью** мы понимаем не только собственно скорость решения задачи, но и возможность описывать и решать большие и сложные задачи.
   - Скорость вычислений. Для решения ЗУО используется универсальный встроенный в систему алгоритм достижения совместности и различные методы перебора с откатами [12, 16]. В худшем случае, скорость перебора экспоненциально зависит от числа переменных, но в среднем её можно улучшить за счёт использования различных эвристик, специализированных ограничений и более эффективного представления данных.

- Возможность формулировать и решать большие задачи закладывалась на этапе проектирования системы.
- Высокоуровневый язык описания модели позволяет быстро описывать решаемую задачу.

3. **Технологичность**. Простота интеграции в системы сторонних производителей достигается за счёт:
   - Модульного строения системы.
   - Мощного внешнего программного интерфейса (API) системы.
   - Использования парадигмы «интерфейс-реализация» (COM-технология) при создании API.

Далее рассмотрим реализацию этих принципов с точки зрения внутренней архитектуры системы (инструментальный уровень) и с точки зрения средств, предоставляемых пользователю для решения конкретных задач (пользовательский уровень).

3. Архитектура системы Nemo. Инструментальный уровень

Система Nemo состоит из:

- Загружаемых модулей (библиотек), содержащий типы данных и ограничения, используемые при создании и обработке вычислительной модели.
- Ядро системы, включающего в себя вычислительную модель, реализацию метода недоопределённых вычислений и менеджер загружаемых модулей.
- API системы.
- Набор утилит для тестирования и отладки системы, трансляторы с языка описания модели Nemo.

Архитектура проектировалась с учётом поддержки различных вычислителей и различных API, хотя на практике был реализован всего один вычислитель и один API для него.

3.1. Библиотеки типов данных и ограничений

Ядро системы реализует базовый алгоритм метода недоопределённых вычислений и не содержит никакой информации о существующих типах данных и ограничениях – вся информация о них хранится во внешних библиотеках. Соответственно, тип данных (или ограничение) становится доступными только после загрузки соответствующей библиотеки с помощью менеджера модулей. Это является результатом последовательного следования принципу расширяемости.

Разработан и реализован достаточно большой список дополнительных модулей — в стандартной конфигурации система включает 12 библиотек, содержащих большой набор (порядка 700) типов данных и отношений.

3.2. Ядро системы

Внутренняя реализация описания задачи (*вычислительная модель*) строится на основе 2 понятий: объект и дескриптор объекта.

- Все переменные задачи являются объектами. Сам объект содержит только своё текущее значение, которое может изменяться в процессе вычислений.
- Дескриптор объекта содержит всю информацию об объекте (полное описание типа данных, доступные интерфейсы и свойства). Также он используется для создания, удаления и копирования объекта.
- Ограничение также является объектом. Связь между ограничением и его аргументами вынесена на уровень выше, в составной объект (см. далее).
- Дескриптор ограничения хранит имя и типы аргументов ограничения.
- Составной объект представляет собой контейнер для объектов.
- Дескриптор составного объекта хранит описание структуры составного объекта (список дескрипторов составляющих его объектов и ограничений, их имена) и таблицу связей между внутренними ограничениями и объектами.
- Отдельная сущность - обобщённый (generic) дескриптор, который создаёт дескрипторы, а не объекты.
- Вычислительная модель является обычным составным объектом.

Ядро также содержит менеджер модулей, список загруженных дескрипторов и реализацию алгоритма недоопределённых вычислений, работающего с вычислительной сетью.

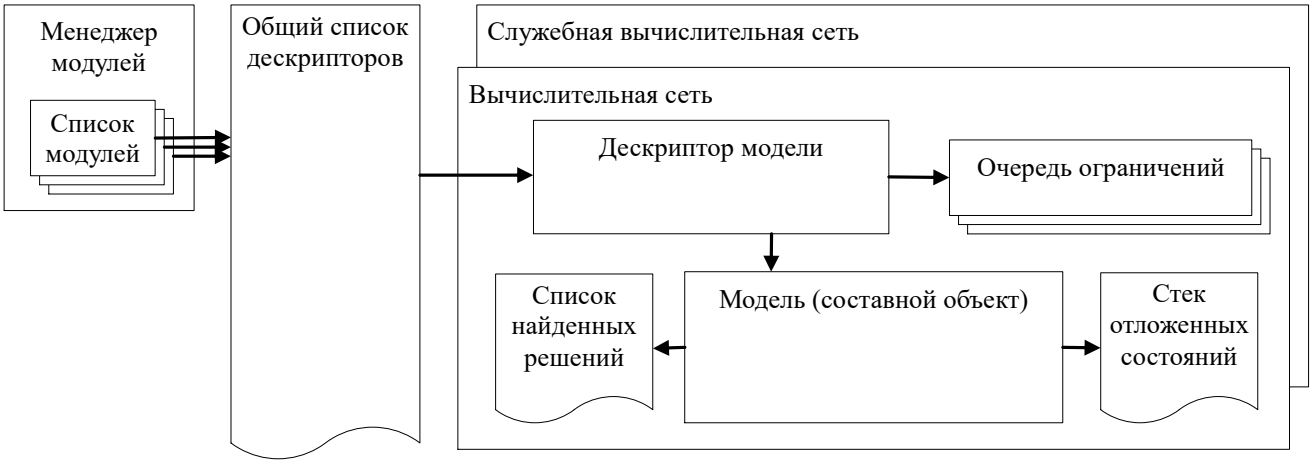

Рис. 1. Строение ядра системы Nemo.

Компоненты вычислителя взаимодействуют следующим образом:

- Менеджер модулей позволяет загружать библиотеки (внешние модули), извлекать из них дескрипторы и добавлять их в общий список дескрипторов.
- Общий список дескрипторов содержит все доступные в конкретный момент времени дескрипторы.
- Внутреннее представлением решаемой задачи (вычислительная сеть) создаётся на основе дескриптора модели, который является обычным дескриптором составного объекта.
- Дескриптор модели используется для создания самой модели (которая является составным объектом), очереди ограничений, и для хранения связей между ограничениями и их аргументами.
- Во время вычислений текущий набор значений объектов модели может сохраняться:
  - в списке найденных решений – если это решение;
  - в стеке отложенных состояний – для обработки этого набора значения в дальнейшем, на следующих шагах алгоритма.
- В некоторых специальных случаях (таких как решение задачи иерархического удовлетворения ограничений, вычисление невязки и т.д.) создаётся служебная вычислительная сеть.

3.3. API

Программный интерфейс системы (API) позволят пользователю формулировать и решать задачи с помощью более высоко-уровневых понятий. Интерфейс API создан с помощью языка описания интерфейсов (idl), реализация написана на C++. API создан в соответствии с моделью «интерфейс-реализациия» объектно-ориентированного парадигмы программирования. API системы состоит из следующих интерфейсов:

Фабрика. Объект этого класса всегда присутствует в единственном экземпляре и служит для создания объектов всех остальных классов.

Модуль. Модуль представляет собой контейнер, содержащий списки типов данных и отношений. Модуль может быть записан в файл и загружен из файла.

Тип данных. Служит для доступа к типам данных модели. Пользователь может создать новый типы данных.

Отношение. Служит для доступа к функциям, операциям и предикатам модели. Пользователь может создать новое отношение (новую функцию или операцию).

Переменная. Переменная модели. При создании переменной указываются её имя и тип данных. Переменная также содержит методы для установки и получения её значения, пометки переменной как константы, для доступа к внутренним слотам переменной (если тип данных переменной - составной).

Терм. Терм является внутренней вершиной дерева выражения. При создании терма указываются отношение и список аргументов (которыми могут быть либо переменные, либо термы). Также как и переменная, терм имеет собственное имя и тип данных.

Ограничение. Элементарный элемент, с помощью которого строится модель. Все факты (уравнения, неравенства, предикаты), которые добавляются в модель, должны быть ограничениями. Ограничение создаётся по терму, являющемуся корнем дерева выражения.

Решатель. Служит для создания модели и управления процессом вычислений.

Таблица. Таблица служит для хранения данных в собственном формате и обеспечивает создание и доступ к строкам и столбцам данных. Таблица используется при создании табличного отношения.

Строка таблицы. Одна строка данных таблицы. Используется для заполнения таблицы данными.

3.4. Сервисный уровень

Сервисный уровень содержит программы и модули для работы с системой Nemo. Из них можно выделить:

- **Компилятор языка** описания модели Nemo является основным инструментом для формулировки задач и тестирования системы.
- **Интерпретатор языка** описания модели Nemo позволяет в интерактивном режиме работать с моделью: загружать модули, создавать переменные, модифицировать модель и выполнять вычисления. Интерпретатор используется для тестирования методов динамического управления моделью через API.
- **Отладчик** является основным средством отладки алгоритма вычислений. Он позволяет выполнять вычисления по-шагово, отображая текущие значения переменных и состояние очереди ограничений.
- Утилита для просмотра содержимого библиотек.
- **Компилятор таблиц** позволяет создавать табличные ограничения на основе текстового представления данных.

- **Декомпиллятор** восстанавливает описание модели на языке Nemo.

Все указанные утилиты и модули взаимодействуют с системой исключительно через её API.

4. Архитектура системы Nemo. Пользовательский уровень

Для пользователя, система, в первую очередь, является инструментом для решения задач. С этой точки зрения её описание представляет собой набор методов и приёмов, используемых для формулировки и решения различных задач. Система Nemo предоставляет 2 способа, с помощью которых возможно сформулировать решаемую задачу:

- Низкоуровневый – с помощью API системы, на языке программирования (C++, Basic).
- Высокоуровневый – с помощью языка описания модели Nemo, в текстовом виде.

Далее мы будем использовать второй способ (описание на языке Nemo), так как язык поддерживает все возможности, предоставляемые API и при этом оперирует более высокоуровневыми понятиями. Отметим, что язык Nemo является развитием языка описания модели системы HeMo+ [7] при сохранении его базовых принципов и свойств.

Напомним базовые свойства системы, которые можно использовать при описании задачи:

- Декларативное представление модели.
- Модульное строение системы.
- Расширяемость (возможность создания новых типов данных и новых отношений).
- Объектно-ориентированное строение модели.
- Поддержка концепции недоопределённости.
- Поддержка различных классов решаемых задач.

4.1. Декларативное представление модели

Так как алгоритм для решения задач встроен в систему, то пользователю достаточно специфицировать задачу (задать объекты, их значения и связывающие их отношения) в терминах языка системы и запустить механизм вычислений.

```
uses "NemNumbers";
main
    x, y, d, z : interval real;
    x + y = 12.0;
```

```
    2 * x = y;
    z^3 + y^5 = d;
    x^2 + 4 * z - 6.33 = 0.0;
end;
```

Пример 1. Простая система уравнений.

В примере:

- загружается библиотека «NemNumbers», которая содержит реализацию основных числовых типов данных и операций над ними (в том числе типа данных «interval real»);
- структурные скобки «main … end» ограничивают описание задачи;
- декларируются 4 переменных: x, y, d, z;
- задаются ограничения задачи в виде системы уравнений.

Как уже было отмечено, декларативность с одной стороны упрощает формулировку задачи, с другой – ограничивает класс решаемых задач. В частности, система не позволяет решать динамические задачи удовлетворения ограничений. Что не мешает использовать систему Nemo как компоненту для решения ЗУО в системах с динамически изменяемой моделью (например, система Semp-ТАО [6]).

4.2. Модульность

Модульное строение системы позволяет наращивать систему новыми типами данных (классами) и отношениями путём подключения дополнительных внешних модулей (библиотек).

Вычислитель системы Nemo реализует базовый алгоритм вычислений и содержит системные классы и методы. При этом он не имеет никакой информации ни о каких конкретных типах данных (классах) и отношениях – все они загружаются из внешних модулей динамически. Перед языком, и частично перед API, это ставит несколько проблем:

- Представление констант. Действительно, так как не существует предопределённых типов данных, и так как число типов данных не ограничено, то транслятор не должен иметь встроенных средств разбора констант. Для решения этой проблемы были приняты следующие соглашения:
  - Сделано исключения для распространённых типов данных: заданы форматы представления значений для типов с именами ***bool***, ***real***, ***int*** и ***string***.
  - Заданы форматы представления значений для видов недоопределённости ***Interval***, ***Enum*** и ***Multiinterval***.

- Для каждого типа данных реализован (непосредственно в дескрипторе объекта) метод получения значение объекта из текстовой строки. При этом формат текстовой строки определяется внутри реализации этого метода. Соответственно, в язык введена конструкция «**const**(***TypeName***, "Value")», которая создаёт константный объект типа TypeName со значением, загруженным из текстовой строки "Value".

- Отношения в языке задаются в виде операций и функций. Наличие большого (потенциально - неограниченного) количества отношений требует от языка и API поддержки возможности свободного задания имён и сигнатур операций и функций. А также возможность их переопределения.
- В связи с большим количеством операций возникает проблема указания их ассоциативности и приоритета, используемых при разборе выражений. Для её решения введены два соглашения:
  - Приоритеты операций с одним именем совпадают. Ассоциативности операций с одним именем совпадают.
  - Приоритеты и ассоциативности операций загружаются из внешнего текстового файла.

Заметим, что для API этой проблемы не существует, так как выражения собираются из термов по-одному, вне зависимости от приоритетов, ассоциативности и префиксной/инфиксной формы вызова отношения.

```
class Point;
    slots (x, y, z: real);
end;

function Norm( P : Point ) : positive;
    Result^2 = (P.x^2 + P.y^2 + P.z^2);
end;

operator - ( P1, P2 : Point ) : Point;
    Result.x = P1.x - P2.x;
    Result.y = P1.y - P2.y;
    Result.z = P1.z - P2.z;
end;
```

```
operator + ( P1, P2 : Point ) : Point;
    Result.x = P1.x + P2.x;
    Result.y = P1.y + P2.y;
    Result.z = P1.z + P2.z;
end;

function Distance( P1, P2: Point ): positive;
    Result = Norm(P1 - P2);
end;
```

Пример 2. Фрагмент модуля геометрической библиотеки.

В примере:

- создаётся новый тип данных «Point», содержащий 3 вещественных поля: x, y, z;
- создаются функции «Norm» и «Distance», а также операции «-» и «+» для работы с этим типом данных.

4.3. Виды недоопределённости

Вид недоопределённости задаёт способ представления множества допустимых значений переменной [10]. В язык введена поддержка видов недоопределённости: ***Single***, ***Interval***, ***Enum***, ***MultiInterval***. При описании переменной модели, можно явно указывать вид недоопределённости; по умолчанию используется ***Interval***.

```
uses "NemNumbers";
uses "NemEnumIntegers";
uses "NemAbstractTypes";

main
 // qi is the queen in the i-th column
    q1, q2, q3, q4, q5, q6, q7, q8 : enum int;
    q1 = [1, 8]; q2 = [1, 8]; q3 = [1, 8]; q4 = [1, 8];
    q5 = [1, 8]; q6 = [1, 8]; q7 = [1, 8]; q8 = [1, 8];

    alldiff( q1, q2, q3, q4, q5, q6, q7, q8 );
    alldiff( q1-3, q2-2, q3-1, q4, q5+1, q6+2, q7+3, q8+4 );
    alldiff( q1+3, q2+2, q3+1, q4, q5-1, q6-2, q7-3, q8-4 );
end;
```

%exact = **On**;

Пример 3. Задача о расстановке ферзей.

В примере:

- Для решения ЗУО в дискретных областях удобно использовать представление значений переменных в виде перечисления. Таким типом данных является «enum int», реализация которого загружается из библиотеки «NemEnumIntegers».
- В данной задаче удобно устанавливать начальные значения переменных с помощью интервалов: «q1 = [1,8]».
- Ограничение «alldiff» задаёт попарное неравенство всех своих аргументов. Ограничение имеет произвольное количество аргументов и реализовано в библиотеке «NemAbstractTypes».
- Параметр алгоритма «%exact» указывает, что выполняется поиск точных решений (см. далее подраздел 4.9. Классы решаемых задач).

4.4. Типы данных

Тип данных представляет собой описание, содержащее информацию, общую для некоторой группы переменных. Типы данных системы Nemo (и языка) можно разделять по различным критериям.

По роли в модели:

- **Абстрактные типы данных** служат для создания иерархии типов данных.
- **Конкретные типы данных** используются для создания объектов модели.

По строению:

- **Простые типы данных.** Объекты таких типов являются неделимым сущностями**.**
- **Структурные типы данных**. Переменные таких типов имеют внутреннее строение, которое можно использовать при описании модели.

По способу создания:

- **Инструментальные типы данных** создаются разработчиками системы. Такие типы реализуются на языке программирования (C++) и могут быть абсолютно произвольными. При описании модели такие типы данных становятся доступными только после загрузки соответствующего модуля.
- **Составные типы данных** могут создаваться пользователем различными способами:

1. Создать тип данных, явно описав его структуру (для записей – это список слотов; для массивов – тип элементов и размер).
2. Наследовать тип данных от уже существующего типа.
3. Добавить к типу данных собственную подмодель (дополнительный набор ограничений на внутренние поля объектов).

- Типы данных, созданные с помощью **обобщённых** (родовых) **классов**.

Отметим что, так как система Nemo работает с недоопределёнными данными, то для каждого (не абстрактного) типа данных должен указываться вид недоопределённости. Иерархия типов данных, реализованных в стандартной конфигурации, представлена на следующей схеме:

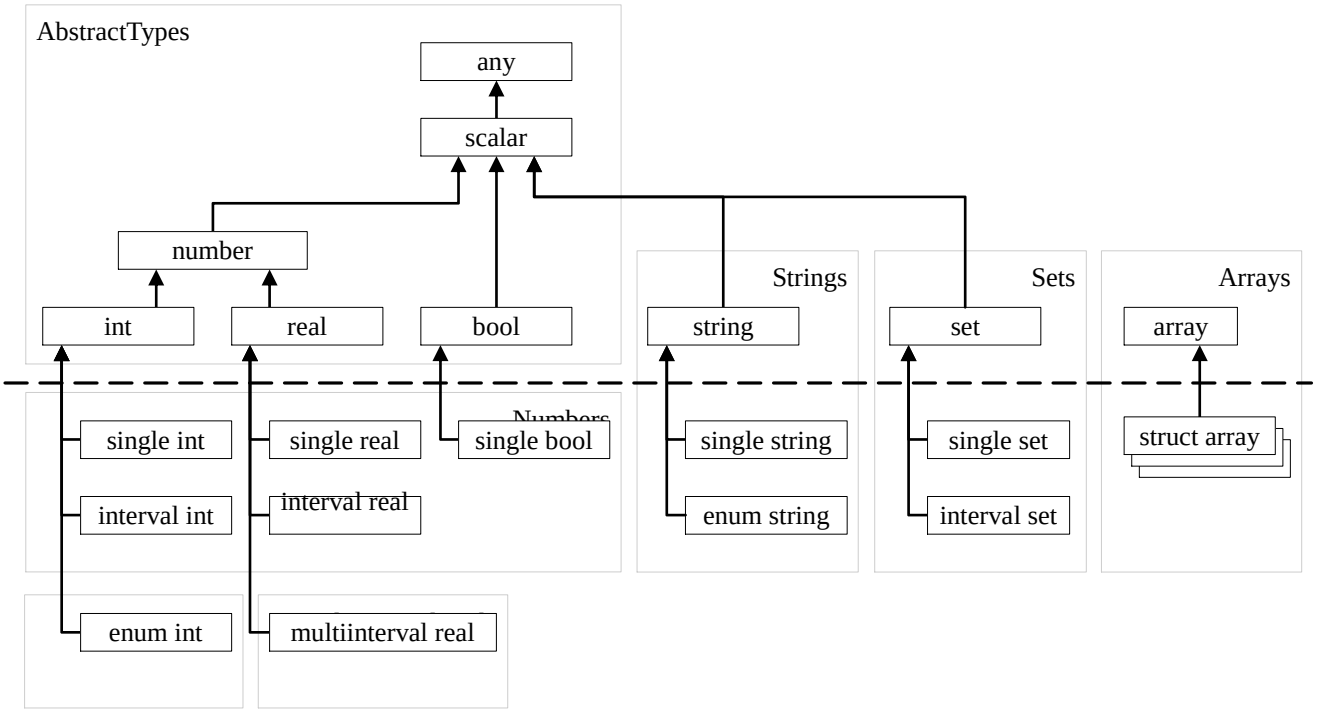

Рис. 2. Иерархия типов данных

Все указанные типы данных могут использоваться сразу же после подключения соответствующих библиотек.

4.5. Отношения

Также как и типы данных (по отношению к переменным), отношения обобщают информацию, общую для некоторой группы ограничений. Отношения можно разделять по различным критериям:

С точки зрения синтаксиса языка:

- **Функции**. Имя функции является идентификатором. В ограничениях функция используется только в префиксной форме.

- **Операции**. Имя операции состоит из специальных символов. В ограничениях операция может использоваться как в инфиксной, так и в префиксной формах.
- **Методы** типов данных. Метод связан с конкретным типом данных и задаёт ограничения, связывающие объект данного типа с внешними объектами.

```
function Norm( P : Point ) : positive;
    Result^2 = (P.x^2 + P.y^2 + P.z^2);
end;

operator + ( P1, P2 : Point ) : Point;
    Result.x = P1.x + P2.x;
    Result.y = P1.y + P2.y;
    Result.z = P1.z + P2.z;
end;

method angle::deg() : real;
    Pi*Result = 180*This;
end;
```

Пример 4. Фрагмент модуля геометрической библиотеки.

В примере создаются отношения:

- функция «Norm»;
- операция сложения для объектов типа данных «Point»;
- метод «deg» для инструментального типа данных «angle».

Заметим, что на уровне API такого разделения не существует – ограничения создаются с явным указанием имени отношения и списка аргументов.

По способу создания:

- **Инструментальные отношения** создаются разработчиками системы. Такие отношения реализуются на инструментальном языке программирования (C++) и могут быть абсолютно произвольными. При описании модели такие отношения становятся доступными сразу после загрузки соответствующего модуля.
- **Составные отношения** могут создаваться пользователем (точнее – инженером знаний). При этом он должен явно указать имя отношения, описать сигнатуру (типы аргументов и результата) и набор ограничений, из которых состоит данное отношение.

- Отношения, созданные с помощью механизма **обобщённых ограничений**.

4.6. Объектно-ориентированная парадигма описания модели

Парадигма объектно-ориентированного программирования (**ООП**) естественным образом вытекает из поддержки типизации и модульного строения системы. Понятие ООП также можно разложить на несколько составляющих:

1. **Типизация**. Это свойство означает наличие понятия «тип данных» («класс»), возможность прямо указывать класс каждого элемента модели и возможность создавать новые классы и отношения. API и язык поддерживает иерархию классов, как реализованных на инструментальном уровне, так и создаваемых пользователем.
2. **Модульность**. Система работает с двумя типами модулей: инструментальный модуль (динамически загружаемая библиотека со стандартизованным интерфейсом классов и процедур) и пользовательский (созданный самим пользователем) модуль. С точки зрения пользователя, они полностью эквивалентны.
3. **Инкапсуляция**. Инкапсуляция означает скрытие информации: данных и реализацию методов взаимодействия с объектом, - внутри самого объекта. Реализация этого понятия в системе Nemo имеет свои особенности:
    - В системе нет прав доступа к внутренним полям объекта – они все доступны (в терминах C++ - «публичные»).
    - Для типа данных можно указать **собственную подмодель** (см. подраздел 4.4. Типы данных), которая позволяет централизовать и скрывать функциональную информацию о связях между внутренними данными объекта.
    - Для типа данных можно описать набор методов для доступа к внутренним полям объекта этого типа (см. подраздел 4.5. Отношения).
4. **Наследование**. Система Nemo предоставляет методы построения новых классов при помощи механизма наследования. При этом имеются некоторые ограничения: запрещено множественное наследование, и базовый тип может быть только составным (массивом или записью).
5. **Полиморфизм**. Поддерживается возможность перегружать слоты и методы классов при наследовании, а также перегружать операции и функции для новых типов аргументов. Но так как нет виртуального наследования и виртуальных методов классов, то нельзя сказать, что полиморфизм реализован в полном объёме.

```
class Point2D; slots(x, y : real); end;
```

```
class Point3D: Point2D;
    slots(z : real);
end;
class Sphere: Point3D;
    slots(R : real);
    R >= 0.0;
end;
class Circle: Sphere;
    z = 0.0;
end;

method Point2D::Norm() : positive;
    Result^2 = This.x^2 + This.y^2;
end;
method Point3D::Norm() : positive;
    Result^2 = This.Norm() + This.z^2;
end;
```

Пример 5. Наследование в языке Nemo.

В примере создается иерархия типов данных: «Point2D» -> «Point3D» -> «Sphere» -> «Circle». Метод «Norm» создаётся для типа данных «Point2D» и перегружается для типа данных «Point3D».

Отметим, что конструкция «This.Norm()» внутри описания метода «Point3D::Norm()» указывает на метод родительского класса «Point2D::Norm()», так как для типа «Point3D» этот метод ещё не определён и, соответственно, не перегружен.

4.7. Механизм обобщённых классов и отношений

Существуют группы типов данных, отличающиеся друг от друга одним или несколькими параметрами. Такие типы данных можно сгруппировать в «метаклассы», которые мы будем называть **обобщёнными классами**. При этом, конкретный тип данных строится на основе обобщённого класса и набора параметров. Характерными примером такой конструкции является массив - для создания конкретного типа массива достаточно:

- Использовать обобщённый класс **generic array**.
- Указать тип элементов будущего массива.
- Указать размер будущего массива.

Тип элементов и размер задаются в виде параметров.

```
uses "NemNumbers";
uses "NemArrays";
uses "NemEnumIntegers";
uses "NemAbstractTypes";
main
    x,y: generic array <5, enum int>;
    i0, i1, i2, i3, i4 : enum int;
    i0=[0,4]; i1=[0,4]; i2=[0,4]; i3=[0,4]; i4=[0,4];
    x[0] = 3; x[1] = 1; x[2] = 7; x[3] = 5; x[4] = 0; // x = {3, 1, 7, 5, 0};
    x[i0] <= x[i1]; x[i1]<= x[i2]; x[i2] <= x[i3]; x[i3] <= x[i4];
    alldiff(i0,i1,i2,i3,i4);

    y[0] = x[i0];
    y[1] = x[i1];
    y[2] = x[i2];
    y[3] = x[i3];
    y[4] = x[i4];
end;
%exact = On;
```

Пример 6. Задача сортировки массива.

В примере создаются 2 массива (x и y). В результате вычислений не отсортированные значения из первого массива будут записаны в заданном порядке во второй массив.

В стандартной конфигурации системы реализованы несколько обобщённых классов:

- **generic array** служит для создания типов данных, описывающих одномерные массивы фиксированной длины.
- **generic scale** используется для создания перечислений.
- **generic grid** создаёт числовой тип данных с указанным шагом значений.
- **generic precision** создаёт вещественный числовой тип данных, все операции над которым выполняются с указанной точностью.

Все вышесказанное справедливо и для отношений: существуют группы отношений, отличающиеся друг от друга некоторыми параметрами; такие группы можно естественным образом сгруппировать в обобщённые классы отношения. В стандартной конфигурации системы реализованы следующие обобщённые классы отношений:

- **generic zero_partially** создаёт ограничения для решения задачи иерархического удовлетворения ограничений (РЗУО).
- **generic table** используется для создания табличных ограничений.
- **generic blackbox** используется для создания BlackBox-ограничений.

Механизм обобщённых классов и отношений поддерживается на уровне вычислителя системы Nemo, и далее в API и в языке Nemo.

4.8. Таблицы и BlackBox-ограничения

Отдельно отметим два вида универсальных отношений, с помощью которых можно смоделировать любое ограничение:

- **Табличные ограничения**. Как правило, ограничения на значения числовых объектов (целых, вещественных, логических) задаются интенсионально (посредством формул). Но в реальных задачах часто приходится иметь дело с ограничениями, заданными экстенсионально (посредством перечисления всех наборов значений, которые удовлетворяют данному ограничению). Каждое такое экстенсиональное ограничение удобнее всего представлять в виде таблицы, число столбцов которой совпадает с числом аргументов ограничения, а в каждой строке записан допустимый данным ограничением набор значений аргументов этого ограничения. На практике, табличные ограничения используются для задания нормативных параметров, для поддержки сложных перечислений и для взаимодействия с базами данных [14].
- **BlackBox-ограничения** (чёрный ящик). Существует класс задач, при описании и решении которых используются внешние, по отношению к системе Nemo, специализированные модули. Такие модули могут появляться в различных случаях:
  - Если для решения задачи необходимо использовать внешний модуль, реализующий некоторый закрытый алгоритм, взаимодействие с которым осуществляется только через заданный публичный интерфейс.
  - Если у пользователя есть готовые модули для решения подзадач и он хочет их использовать для задачи, решаемой системой Nemo. В этом случае пользователю может быть известна некоторая дополнительная информация, касающаяся поведения алгоритма. Например монотонность или дифференцируемость функции, вычисляемой алгоритмом.

  С точки зрения системы Nemo такие ограничения являются «чёрными ящиками» и для работы с ними создан специальный вид ограничений – BlackBox-ограничения [9, 18].

При использовании языка Nemo табличные и BlackBox-ограничения создаются с помощью механизма обобщённых классов:

```
function AB( abs1, abs2: int );
    table<"./", // path
            "Trains_from_a_to_b.nbt", // table name
            "1", // memory or file
            binary table, // table descriptor
            interval int, interval int, // column descriptors
            interval int, interval int> // argument descriptors
    ( abs1, abs2 );
end;

function Volume(x, y, z :real) : real;
    blackbox<
            "x1 * x2 * x3", "BlackBoxTest", "Calculator", 10000,
            "input single real[3]", "output single real", 0, 0,
            "NGradient exact", 1.e-5, 1, 0>
    (x, y, z ) = Result;
end;
```

Пример 7. Описание табличного и BlackBox отношений.

В примере создаются 2 ограничения:

- таблица, загружаемая из файла «Trains_from_a_to_b» (расписание поездов) и содержащая 2 целочисленных столбца;
- BlackBox-ограничение с основе функции «Calculator» из библиотеки «BlackBoxTest» с 3 вещественными аргументами и вещественным результатом.

В обоих случаях для создания конкретного ограничения с помощью обобщённого отношения, приходится указывать множество дополнительных константных параметров, которые не влияют на дальнейшее использование созданных ограничений. Поэтому, для упрощения их использования, ограничения создаются внутри составных отношений «AB» и «Volume».

4.9. Классы решаемых задач

Базовый алгоритм, реализованный в системе Nemo, служит для решения задачи достижения локальной совместности. С точки зрения пользователя, он находит внешнюю

оценку множества решений ЗУО. Тем не менее, с помощью специализированных ограничений, система позволяет решать следующие задачи:

- Поиск точного решения ЗУО с заданной точностью.
- Поиск точного решения, ближайшего к текущему значению аргументов.
- Поиск нескольких/всех точных решений.
- Поиск частичного решения.
- Решение задач оптимизации.
- Решение задач иерархического удовлетворения ограничений (Partial Constraint Satisfaction Problem).

Кроме того, пользователю предоставляется доступ к некоторым параметрам базового алгоритма, которые позволяют управлять процессом поиска решений. В частности можно задавать следующие параметры:

- Задавать начальное значение переменной (**%***var***=**...).
- Задавать точность найденного решения (**%eps=**...).
- Задавать невязку ограничений для решения (**%discrepancy=**...).
- Задавать число требуемых решений (**%solutions=**...).
- Сохранять лучшее частичное решение (**%partial=**...).
- Включать все переменные модели в механизм перебора с откатами (**%exact=**...).
- Задавать целевое значение переменной (**%***var***~**...).
- Задавать шаг решения по переменной (**%eps**(*var*)**=**...).

```
uses "NemNumbers";
uses "NemAbstractTypes";
main
    x1,x2,x3,x4,x5 : real;
    3*x1*(x2 - 2*x1) + 0.25*x2^2 = 0.0;
    3*x5*(20 - 2*x5 + x4) + 0.25*(20 - x4)^2 = 0.0;
    3*x2*(x3 – 2*x2 + x1) + 0.25*(x3 - x1)^2 = 0.0;
    3*x3*(x4 – 2*x3 + x2) + 0.25*(x4 - x2)^2 = 0.0;
    3*x4*(x5 – 2*x4 + x3) + 0.25*(x5 - x3)^2 = 0.0;
end;
//
%solutions=10
%x1 ~ 0.0
```

```
%x2 ~ 0.0


%partial=On
%discrepancy = 1e-8


%eps(x1) = 0.1
%eps(x2) = 0.1
%eps(x3) = 0.1
%eps(x4) = 0.1
%eps(x5) = 0.1
```

Пример 8. Управление механизмом поиска решений.

В примере заданы следующие параметры алгоритма:

- вычисления останавливаются после нахождения 10 решений («%solutions=10»);
- решения ищутся последовательно, начиная от окрестности целевой точки {x1, x2} = {0.0, 0.0};
- сохраняется лучшее частичное решение («%partial=On»);
- все ограничения удовлетворяются с невязкой не более 1e-8 («%discrepancy = 1e-8»);
- расстояние между найденными решениями будет не менее 0.1 по каждой из переменных.

4. Заключение

В статье описывается система программирования в ограничениях Nemo и её особенности, позволяющие эффективно описывать и решать ЗУО.

Система Nemo предоставляет пользователю широкие возможности:

- для декларативного описания задачи;
- для настройки на предметную область с помощью создания собственных типов данных и ограничений;
- для повышения уровня представления данных с помощью методов объектно-ориентированной и обобщённой парадигм программирования;
- для расширения классов решаемых задач с помощью собственных специализированных ограничений и типов данных;
- для интеграции в состав сторонних систем.

Несмотря на долгую историю развития, возможности дальнейшей модернизации системы далеко не исчерпаны. Благодаря заложенным на этапе проектирования принципам система

Nemo может использоваться как инструмент для решения сложных задач, как среда для разработки новых алгоритмов для решения задач в области ЗУО (например: нахождение 3В-совместности, быстрое решение подсистем линейных уравнений, использование производных при поиске точного решения), как вычислительный модуль в составе сторонних систем.

Список литературы

1. Ботоева Е.Ю., Костов Ю.В., Петров Е.С. Универсальный решатель UniCalc // Информационный бюллетень рабочего семинара «Наукоемкое программное обеспечение», Новосибирск: ИСИ СО РАН, 2006, С.42-45.
2. Булгаков. С.В. Подход к построению мульти-агентной системы содержательного поиска во множестве разнородных структурированных источников данных // КИИ’2004. Труды 9-й национальной конференции по искусственному интеллекту с международным участием. Том 2. Москва: Физматлит, 2004. С. 706-714.
3. Гончар А.М., Загорулько Г.Б., Рубан М.Н., Рябков А.Н. Экспертная система поддержки диагностики, профилактики и лечения элементозов на основе коррекции питания // КИИ’2006. Труды 10-й национальной конференции по искусственному интеллекту с международным участием. Том 3. Москва: Физматлит, 2006. С. 849-857.
4. Загорулько Г.Б., Сидоров В.А., Телерман В.В. и др. НеМо+: Объектно-ориентированная среда программирования в ограничениях на основе недоопределённых моделей // КИИ'98. Шестая национальная конференция с международным участием. Сборник научных трудов в трёх томах. Том I. Пущино, 1998. С. 524–530.
5. Загорулько Г.Б., Сидоров В.А., Телерман В.В. и др. Обстановка для программирования в ограничениях на основе недоопределённых моделей NeMo+ (язык, архитектура, интерфейс). // Научно-техн. отчёт N 7 / Российский НИИ искусственного интеллекта, Институт систем информатики им. А. П. Ершова СО РАН. Москва, Новосибирск, 1998. 107 с.
6. Загорулько Ю.А., Попов И.Г., Костов Ю.В. Интеграция технологии баз знаний с агентной технологией и методами программирования в ограничениях. // Материалы международной научно-технической конференции "Информационные системы и технологии" (ИСТ '2000). Новосибирск: Издательство НГТУ, 2000. Т.3. С. 508.
7. Нариньяни А.С. Недоопределенность в системе представления и обработки знаний // Изв. АН СССР. Техническая кибернетика. 1986. № 5. С. 8-11.
8. Нариньяни А.С. Недоопределённые модели и операции с недоопределёнными значениями // Новосибирск, 1982. 33 с. (Препринт / АН СССР. Сиб. отд-ние. ИЦ; № 400).
9. Сидоров В.А. Программирование в ограничениях с чёрными ящиками. Новосибирск, 2003. 39 с. (Препринт / ЗАО Ледас; №2).

10. Сидоров В.А. Сравнение способов представления неточных значений в методе недоопределённых вычислений. // Системная информатика. 2015. № 6. С. 53-66.
11. Телерман В.В., Ушаков Д.М. Недоопределенные модели: формализация подхода и перспективы развития // Проблемы представления и обработки не полностью определенных знаний / Под ред. И.Е. Швецова. Москва-Новосибирск: РосНИИ ИИ, 1996. С. 7–30.
12. Golomb S.W. , Baumert L.D. Backtrack programming // J. of the ACM. 1965. Vol 12, №. 4. P. 516–524.
13. Huffman D.A. Impossible objects as nonsence sentences // Machine Intelligence. 1971. Vol. 6. P. 295–323.
14. Lipski S., Sidorov V., Telerman V., Ushakov D. Database Processing in Constraint Programming Paradigm Based on Subdefinite Models // Joint Bulletin of NCC&IIS,. 12 (1999), NCC Publiher. Novosibirsk, 1999.
15. Mackworth A.K. Consistency in Networks of Relations. // Artificial Intelligence. 1977. № 8. P. 99–118.
16. Nilsson N.J. Principles of artificial intelligence. // Numerical Computation Guide. Mountain View, USA, November, 1995.
17. Shvetsov I., Kornienko V., Preis S.. Interval spreadsheet for problems of financial planning // PACT`97, England, London, 1997. P. 373-385
18. Sidorov V., Telerman V. Industrial Application of External Black-Box Functions in Constraint Programming Solver. // Perspectives of System Informatics: Proc./ Ed. by M.Broy, A.Zamulin. Berlin a.o.: Springer-Verlag, 2003. P. 415 – 422. (Lect. Notes Comput. Sci.; 2890).