Knowledge Base of Intelligent Information System for Prediction of Phase Stability of Solid Solutions

Oleksii Kudryk, Oleg Bisikalo and Yurii Ivanov

Vinnitsa National Technical University, Khmel'nyts'ke sh., 95, Vinnytsya, Vinnyts'ka oblast', 21000, Ukraine

Abstract
This research work is devoted to the development of a knowledge base for solving the current problem of forecasting the phase stability of solid solutions. Knowledge bases mean a set of facts and inference rules that allow logical conclusion and purposeful processing of information. The most important property of information stored in knowledge bases is the reliability of specific and generalized information in the database and the relevance of the original information obtained using the rules of inference embedded in the knowledge base. The best knowledge bases include the most relevant, reliable and fresh information, have perfect information search systems, a carefully thought-out structure and format of knowledge. The expert system embodies the methodology of adapting the algorithm of successful solutions from one sphere of scientific and practical activity to another. With the spread of computer technologies, it is an identical intelligent computer program that contains the knowledge and analytical abilities of one or more experts in some field of application and is able to draw logical conclusions based on this knowledge, thereby providing a solution to specific tasks. An intelligent information system (IIS) is one of the types of automated information systems, which is a complex of software, linguistic and logical-mathematical tools for the implementation of the main task, which usually consists of data interpretation and forecasting. Data interpretation is one of the traditional tasks for expert systems. Interpretation means the process of determining the content of data, the results of which must be agreed and correct. A multivariate analysis of the data is usually assumed, and the findings from this model form the basis for probabilistic estimates. Forecasting allows you to predict the consequences of some events or phenomena based on the analysis of available data. A parametric dynamic model is usually used in the forecasting system, in which parameter values are set for a given situation. As a result of the development, a knowledge base model was built for predicting the phase stability of solid solutions using a set of production rules, predicates, functions and operators.

Keywords 1
Intelligent information system, knowledge base, production rules, predicates, operators.

1. Introduction, formulation problem in general

A knowledge base is a collection of systematized basic information related to a certain field of knowledge and the means by which knowledge is accumulated, stored, updated, and used.

The difference between a knowledge base and a database is the possibility of forming new knowledge [1].

The main functions of the knowledge base regarding automated design systems (CAD):

- description of the subject area of CAD;
- support for intellectual methods of solving problems that are part of CAD;
- realization of possibilities of expert analysis of project tasks.

COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems, April 20–21, 2023, Kharkiv, Ukraine
EMAIL: kydrikalex@gmail.com (O. Kudryk); obisikalo@gmail.com (O. Bisikalo), Yura881990@i.ua (Yu. Ivanov)
ORCID: 0000-0002-0592-6633 (O. Kudryk); 0000-0002-7607-1943 (O. Bisikalo), 0000-0003-2125-1004 (Yu. Ivanov)

Production rules are a form of representation of human knowledge in the form of sentences of the IF (condition), THEN (action) type. Rules provide a formal way to present recommendations, guidelines, or strategies. They are ideal in cases where knowledge of the subject area arises from empirical associations accumulated over years of work on solving problems in a particular field.

A production model is a set of production rules, which, on the one hand, is close to logical models, which allows you to organize effective derivation procedures on it, and on the other hand, reflects knowledge more clearly.

Production rules are used in artificial intelligence systems (for example, expert systems) [2], as one of the most common forms of knowledge representation, along with logical models, frames, and semantic networks [3].

An expert system is a methodology for adapting the algorithm of successful solutions from one sphere of scientific and practical activity to another. With the spread of computer technologies, it is an identical (similar, based on an optimizing algorithm or heuristics) intelligent computer program that contains the knowledge and analytical abilities of one or more experts in some field of application and is able to make logical conclusions based on this knowledge , thereby ensuring the solution of specific tasks (consulting, training, diagnosis, testing, design, etc.) without the participation of an expert (a specialist in a specific problem area). It is also defined as a system that uses a knowledge base to solve tasks (issuing recommendations) in a certain subject area. This class of software was originally developed by artificial intelligence researchers in the 1960s and 1970s and gained commercial use beginning in the 1980s. Often, the term knowledge-based system is used as a synonym for an expert system, however, the capabilities of expert systems are wider than the capabilities of systems based on deterministic (limited, currently implemented) knowledge [4].

An intelligent information system (IIS) is one type of automated information system, sometimes IIS is called a knowledge-based system. IIS is a complex of software, linguistic, and logical-mathematical tools for the implementation of the main task: supporting human activity and searching for information in the mode of extended dialogue in natural language [5].

A production model is a set of production rules, which, on the one hand, is close to logical models, which allows you to organize effective derivation procedures on it, and on the other hand, reflects knowledge more clearly.

Production rules are used in artificial intelligence systems (for example, expert systems) [2], as one of the most common forms of knowledge representation, along with logical models, frames, and semantic networks [3].

An expert system is a methodology for adapting the algorithm of successful solutions from one sphere of scientific and practical activity to another. With the spread of computer technologies, it is an identical (similar, based on an optimizing algorithm or heuristics) intelligent computer program that contains the knowledge and analytical abilities of one or more experts in some field of application and is able to make logical conclusions based on this knowledge , thereby ensuring the solution of specific tasks (consulting, training, diagnosis, testing, design, etc.) without the participation of an expert (a specialist in a specific problem area). It is also defined as a system that uses a knowledge base to solve tasks (issuing recommendations) in a certain subject area. This class of software was originally developed by artificial intelligence researchers in the 1960s and 1970s and gained commercial use beginning in the 1980s. Often, the term knowledge-based system is used as a synonym for an expert system, however, the capabilities of expert systems are wider than the capabilities of systems based on deterministic (limited, currently implemented) knowledge [4].

An intelligent information system (IIS) is one type of automated information system, sometimes IIS is called a knowledge-based system. IIS is a complex of software, linguistic, and logical-mathematical tools for the implementation of the main task: supporting human activity and searching for information in the mode of extended dialogue in natural language [5].

2. Formulation of the article aim

The aim of the article is to develop a knowledge base for an intelligent information system for forecasting the phase stability of solid solutions.

3. Building a knowledge base

Let's consider the model of the production-type knowledge base for predicting the phase stability of solid solutions, which consists of a set of relations, production rules, predicates, functions, and operators [6, 7]. Let's present the model being developed as

$$KnowledgeBase = \langle Re, Rule, Pr, Func, Op \rangle, \quad (1)$$

consisting of RE relations, production rules, predicates, functions, operators

$$Rule = \{ParamsR, ParamsChargeR, ParamsChargeCoordinationR, TypeAddR, TypeAdd\}, \quad (2)$$

$$Pr = \{A(pi), B(pcj), C(pccn), TypeAddj, Typej, PA(aj), AD(sd), AL(prmj)\}, \quad (3)$$

$$Func= \{ElementParams, ElementParamsCharge, ElementParamsCoordination, ElementTypeAdd, ElementType, PredAnalysis, AnalysisData, PredTypeAlgorithm\}, \quad (4)$$

$$Op = \{Oprint, Oanalysis, OdataCheck, Opred, OsaveResult, OreturnResult\}. \quad (5)$$

The following are the formal relationships that allow us to consider the informational component of the "Intelligent system of phase stability of solid solutions" in terms of the relational data model [8, 9]:

$$RE = \{element\_grp, sub\_element\_grp, element, elm2elm, charge\_element, cordination\_element, atom\_length, sum\_atom\_length, volume\_cell, structure\_solid, term\_system, tsys\_crde, stored\_system, stored\_result\}, \quad (6)$$

where *element_grp* is a relation for the characteristics of a group of elements; the attributes that make up this relation are: *elmg_id* (unique ID code of a certain group of elements), *elmg_name* (name of the group of elements).

$$element\_grp \subset elmg\_id \times elmg\_name; \quad (7)$$

sub_element_grp – relation for characteristics of a subgroup of elements; the attributes that make up this relationship denote: selmg_id (unique ID code of a certain subgroup of elements), elmg_name (name of the subgroup of elements).

$$sub\_element\_grp \subset selmg\_id \times selmg\_name; \quad (8)$$

element – relation to characterize a subgroup of elements; the attributes that make up this relationship are: *elm_id* (unique ID code of a certain element), *elm_selmg* (subgroup identifier), *elm_name* (element name), *elm_code* (chemical code), *elm_count* (number of element atoms).

$$element \subset elm\_id \times elm\_selmg \times elm\_name \times elm\_code \times elm\_count; \quad (9)$$

elm2elm – relation for the characteristic of a complex element; the attributes that make up this relationship are: *e2e_id* (unique ID code of a certain complex element), *e2e_che_par* (parent charge identifier), *e2e_che_ch* (child charge identifier), *e2e_sort* (sorting).

$$elm2elm \subset e2e\_id \times e2e\_che\_par \times e2e\_che\_ch \times e2e\_sort; \quad (10)$$

charge_element – relation for characterizing element charges; the attributes that make up this relationship denote: *che_id* (unique ID code of a certain element charge), *che_elm* (element identifier), *che_value* (charge value).

$$charge\_element \subset che\_id \times che\_elm \times che\_value; \quad (11)$$

cordination_element – relation for characterizing charge coordination numbers; the attributes that make up this relationship are: *crde_id* (unique ID code of a specific charge coordination number), *crde_che* (element charge identifier), *crde_value* (coordination number value).

$$cordination\_element \subset crde\_id \times crde\_che \times crde\_value; \quad (12)$$

atom_length – relation for characterizing the atomic lengths between element charges; the attributes that make up this relationship are: *atml_id* (unique ID code of a certain atomic length between element charges), *atml_che_from* (identifier of charge of element 1), *atml_che_to* (identifier of charge of element 2), *atml_value* (value of atomic lengths), *atml_strs* (solid solution structure identifier).

$$atom\_length \subset atml\_id \times atml\_che\_from \times atml\_che\_to \times atml\_value \times atml\_strs; \quad (13)$$

sum_atom_length – relation for characterizing the sum of atomic lengths between element charges; the attributes that make up this relation are: *satml_id* (unique ID code of a certain sum of atomic lengths between element charges), *satml_che_from* (identifier of charge of element 1), *satml_che_to* (identifier of charge of element 2), *satml_value* (value of sums of atomic lengths), *satml_strs* (solid solution structure identifier).

$$sum\_atom\_length \subset satml\_id \times satml\_che\_from \times satml\_che\_to \times satml\_value \times \quad (14)$$

$$\times \mathit{satml\_strs};$$

volume_cell – relation for characterizing the volume of cells between element charges; the attributes that make up this relation are: *volc_id* (unique ID code of a certain volume of cells between element charges), *volc_che_from* (element charge identifier 1), *volc_che_to* (element charge identifier 2), *volc_value* (volume value), *volc_strs* (identifier solid solution structures).

$$\mathit{volume\_cell} \subset \mathit{volc\_id} \times \mathit{volc\_che\_from} \times \mathit{volc\_che\_to} \times \mathit{volc\_value} \times \mathit{volc\_strs}; \qquad (15)$$

structure_solid – ratio for characterizing the structure of a solid solution; the attributes that make up this relation are: *strs_id* (unique ID code of a certain solution structure), *strs_name* (name of a group of elements).

$$\mathit{structure\_solid} \subset \mathit{strs\_id} \times \mathit{strs\_name}; \qquad (16)$$

term_system – relation for the characteristics of the thermodynamic system; the attributes that make up this relation are: *tsys_id* (unique ID code of a certain thermodynamic system), *tsys_crde* (identifier of the coordination element), *tsys_elm* (identifier of the element).

$$\mathit{term\_system} \subset \mathit{tsys\_id} \times \mathit{tsys\_crde} \times \mathit{tsys\_elm}; \qquad (17)$$

tsys_crde – relation for characterizing elements related to a certain thermodynamic system; the attributes that make up this relation are: *tsysc_id* (unique ID code of a certain element of the thermodynamic system), *tsysc_tsys* (identifier of the thermodynamic system), *tsysc_crde* (identifier of the coordination element).

$$\mathit{Tsys\_crde} \subset \mathit{tsysc\_id} \times \mathit{tsysc\_tsys} \times \mathit{tsysc\_crde}; \qquad (18)$$

stored_system – relation for the characteristics of the given system, which was calculated; the attributes that make up this relation are: *stds_id* (the unique ID code of certain data about the system that was calculated), *stds_strs* (the structure identifier of the system), *stds_ln1* (the identifier of lanthanide 1), *stds_ln2* (the identifier of lanthanide 2), *stds_anion* (the identifier anion), *stds_eps* (calculation step).

$$\mathit{stored\_system} \subset \mathit{stds\_id} \times \mathit{stds\_strs} \times \mathit{stds\_ln1} \times \mathit{stds\_ln2} \times \mathit{stds\_anion} \times \mathit{stds\_eps}; \qquad (19)$$

stored_result – relation to characterize data on system calculations; the attributes that make up this relation are: *stdr_id* (unique ID code of certain data about the calculated system), *stdr_stds* (identifier of the calculated system), *stdr_x* (result x), *stdr_x1* (result x1), *stdr_x2* (result x2), *stdr_t* (t result), *stdr_t_crit* (t_crit result), *stdr_q* (q result), *stdr_method* (method result), *stdr_err_message* (error text).

$$\begin{gathered}\mathit{stored\_result} \subset \mathit{stdr\_id} \times \mathit{stdr\_stds} \times \mathit{stdr\_x} \times \mathit{stdr\_x1} \times \mathit{stdr\_x2} \times \mathit{stdr\_t} \times \\ \times \mathit{stdr\_t\_crit} \times \mathit{stdr\_q} \times \mathit{stdr\_method} \times \mathit{stdr\_err\_message}.\end{gathered} \qquad (20)$$

The scheme of formal relations is in the Figure 1 [10].

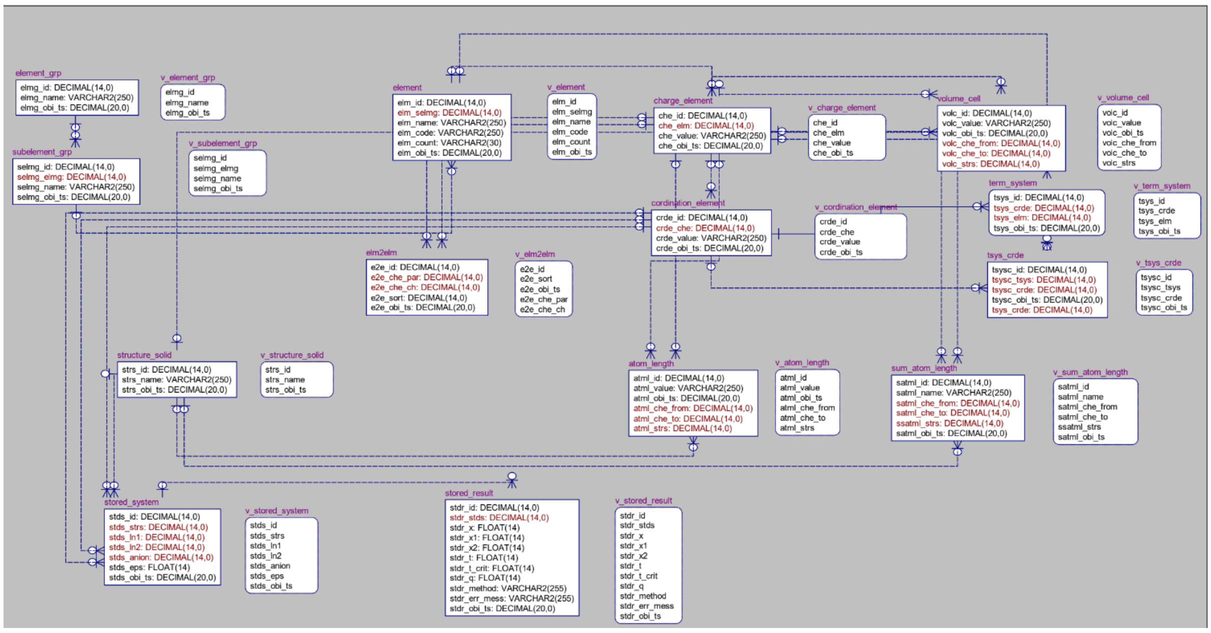


Figure 1: Scheme of formal relations

By the term element, we will understand the component, which is determined by certain parameters. In general, the parameters change depending on the type of element [11]. Using the id of the element from the element table, it is possible to connect several elements, which later form a complex element. For this, a knowledge base of the production type of the intelligent module "*Elements*" was developed, which consists of a set of production rules, predicates, functions and operators [12–15].

In the process of creating an element, the possibility of adding general parameters to the element is taken into account (*ElementParams* function). So, for each element, it is necessary to ensure that indicators are obtained from the database. Therefore, if the parameter *pi* of some *i*-th element from *N* is not present in the system, the universal value *pdef* is chosen, otherwise it is necessary to enter the value *pi*. Let's formally define *ParamsR* products:

$$\text{If } A(pi) \text{ then } ElementParams(pi) \text{ else } ElementParams\,(pdef), \tag{21}$$

where the predicate *A*(*pi*) is defined as follows:

$$\exists p_i \mid i \in \{1,2,\dots,N\} \to A(p_i) = True. \tag{22}$$

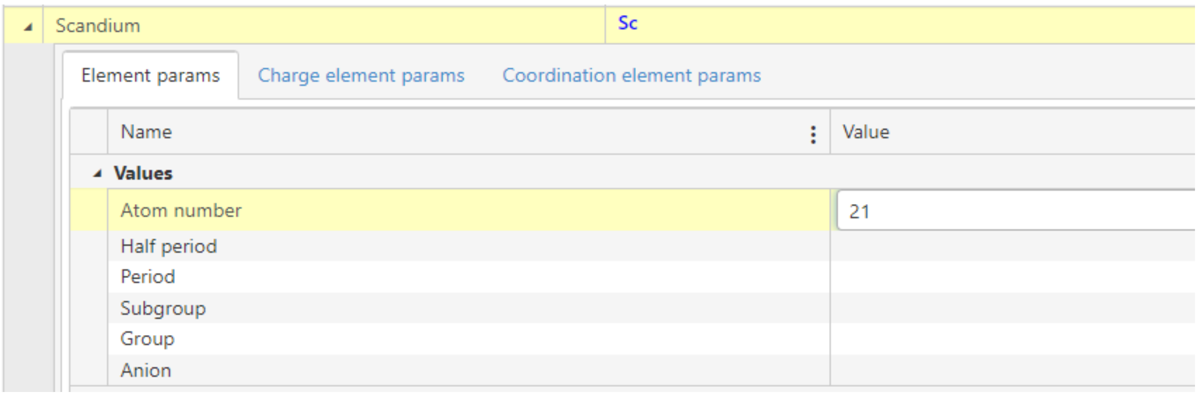
Figure 2 shows an example of entering general parameters.

In the process of creating an element, the possibility of adding charge parameters to the element is taken into account (*ElementParamsCharge* function). So, for each element, it is necessary to ensure the receipt of pointers from the database. Therefore, if the *pcj* parameter of some *j*-th element from *M* is not present in the system, the universal value of *pcdef* is chosen, otherwise it is necessary to enter the value of *pcj*. We will formally define *ParamsChargeR* products:

$$\text{If } B(pcj) \text{ then } ElementParamsCharge(pcj) \text{ else } ElementParamsCharge(pcdef), \tag{23}$$

where the predicate *B*(*pcj*) is defined as follows:

$$\exists pc_j \mid j \in \{1,2,\dots,M\} \to B(pc_j) = True\,. \tag{24}$$



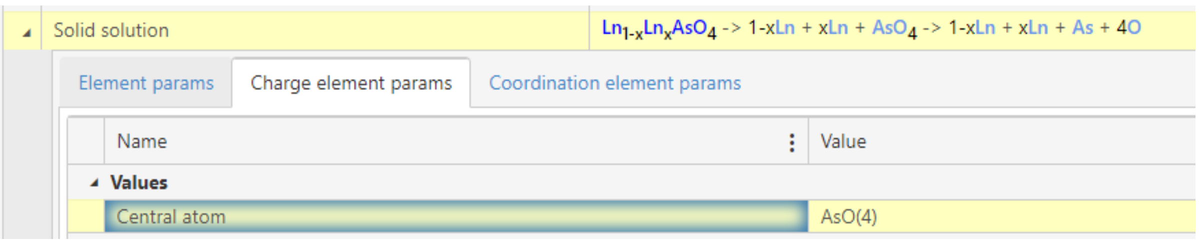
Scandium | Sc

Element params | Charge element params | Coordination element params

Name	Value
Values	
Atom number	21
Half period	
Period	
Subgroup	
Group	
Anion	

Figure 2: Fill in general parameters

Figure 3 shows an example of entering the element charge parameters.



Solid solution | $Ln_{1-x}Ln_xAsO_4$ -> 1-xLn + xLn + AsO_4 -> 1-xLn + xLn + As + 4O

Element params | Charge element params | Coordination element params

Name	Value
Values	
Central atom	AsO(4)

Figure 3: Entering of element charge parameters

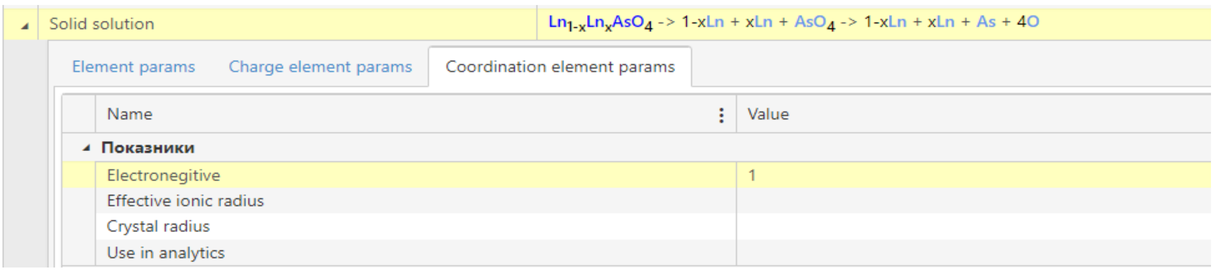
In the process of creating an element, the possibility of entering parameters of the coordination number to the element is taken into account (*ElementParamsChargeCoordination* function). Yes, for each element, it is necessary to ensure that indicators are obtained from the database. Therefore, if the parameter pccn of some *n*-th element from *K* is not present in the system, the universal value *pccdef* is chosen, otherwise it is necessary to enter the value *pccn*. Let's formally define *ParamsChargeCoordinationR* products:

If *C*(*pccn*) then *ElementParamsChargeCoordination*(*pccn*)
else *ElementParamsChargeCoordi-nation* (*pccdef*), (25)

where the predicate *C*(*pccn*) is defined as follows:

$$\exists pcc_n \mid n \in \{1,2,...,K\} \rightarrow C(pcc_n) = True. \tag{26}$$

Figure 4 shows an example of entering the element charge parameters.



Solid solution | $Ln_{1-x}Ln_xAsO_4$ -> 1-xLn + xLn + AsO_4 -> 1-xLn + xLn + As + 4O

Element params | Charge element params | Coordination element params

Name	Value
Показники	
Electronegitive	1
Effective ionic radius	
Crystal radius	
Use in analytics	

Figure 4: Entering the parameters of the coordination number to the element

System users can independently edit the type of element they need. In order to support flexibility, a drop-down list in the element window has been developed: "Complex" and "Simple". Through the “Complex” menu, users can select the *j*-th tapes *rj* from the *LR* element table, which are needed to build a complex element (2+ elements) *ElementTypeAdd*, and through the “Simple” menu, it is possible to select elements (up to 2 elements) in certain *j*-th fields from the *LF* table element by the *ElementType* function. Let's formally define the *TypeAddR* and *TypeR* products:

If *TypeAddj* then *ElementTypeAdd*(*rj*) else *ElementTypeAdd*(*Null*), (27)

where the *TypeAddj* predicate is defined as follows:

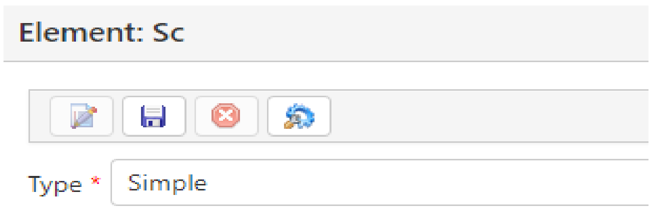
$$\exists TypeAdd_j \mid j \in \{1,2,...,LR\} \rightarrow TypeAdd_j = True. \tag{28}$$

If *Typej* then *ElementType*(*rj*) else *ElementType*(*Null*), (29)

where the *Typej* predicate is defined as:

$$\exists Type_j \mid j \in \{1,2,...,LF\} \rightarrow Type_j = True. \tag{30}$$

Figure 5 shows an example of changing the type of an element.



Element: Sc

Type * Simple

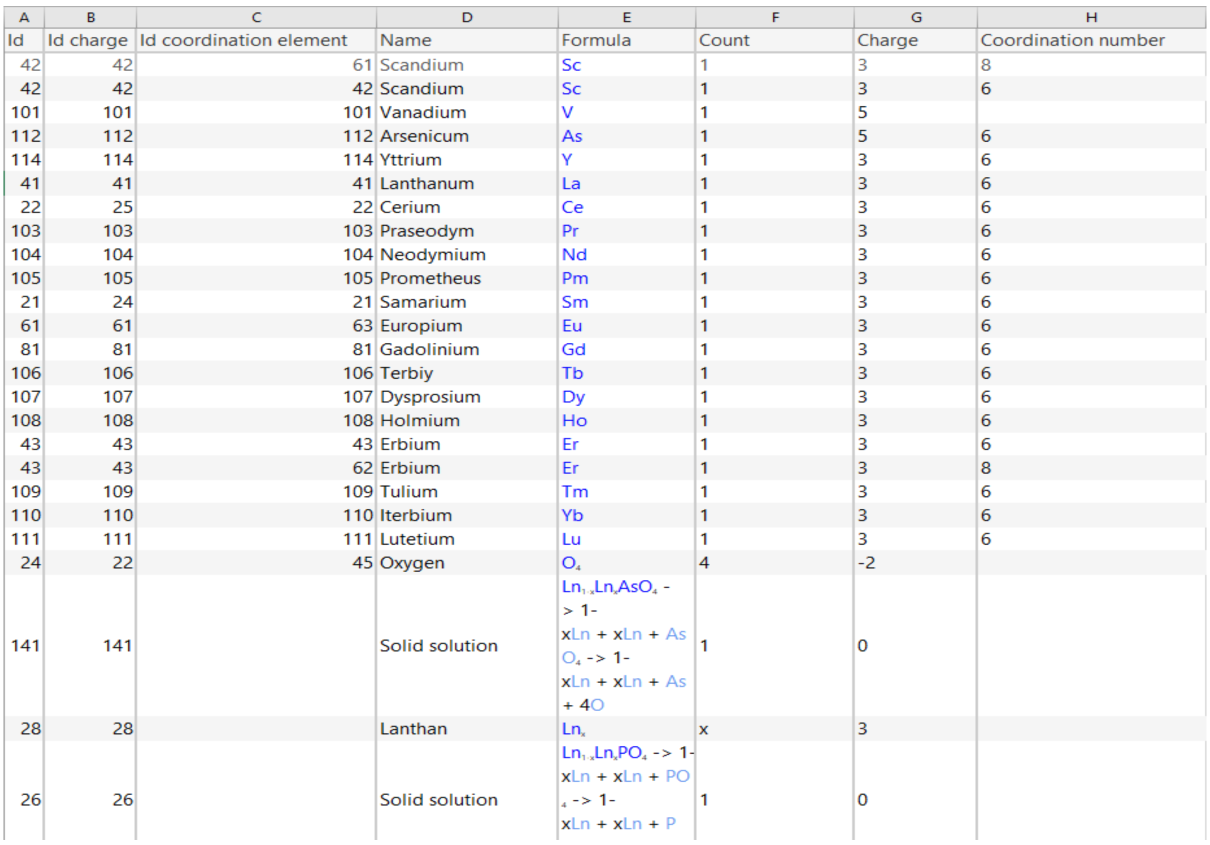
Figure 5: Changing the type of an element

In order to visually easily show how an element was created and which elements were entered into the system, a function was created to print the table of *PrintElement* elements. Actually, each element is formed from a certain set of fields: *elm_id* (unique id code of a certain element), *elm_selmg* (subgroup identifier), *elm_name* (element name), *elm_code* (chemical code), *elm_count* (number of element atoms). Formally, the element print operator

O_{print} : (*elm_id, elm_selmg, elm_name, elm_code, elm_selmg*) → *PrintElement.* (31)

Figure 6 shows an example of printing a table of elements.

By the term forecasting, we will understand the component that begins with the moment of analysis, data processing and ends with the output of data on the user interface. Using data from the tables: *element*, *charge_element*, *coordination_element*, *volume_cell*, *term_system*, *atom_length*, *sum_atom_length*, *stored_system*, *stored_result*, analyzing and processing them, it is possible to obtain data that will be displayed on the user interface in the form of a graph and a table. For this, a knowledge base of the production type of the intelligent module "Forecasting" has been developed, which consists of a set of production rules, predicates, functions and operators.



A	B	C	D	E	F	G	H
Id	Id charge	Id coordination element	Name	Formula	Count	Charge	Coordination number
42	42	61	Scandium	Sc	1	3	8
42	42	42	Scandium	Sc	1	3	6
101	101	101	Vanadium	V	1	5	
112	112	112	Arsenicum	As	1	5	6
114	114	114	Yttrium	Y	1	3	6
41	41	41	Lanthanum	La	1	3	6
22	25	22	Cerium	Ce	1	3	6
103	103	103	Praseodym	Pr	1	3	6
104	104	104	Neodymium	Nd	1	3	6
105	105	105	Prometheus	Pm	1	3	6
21	24	21	Samarium	Sm	1	3	6
61	61	63	Europium	Eu	1	3	6
81	81	81	Gadolinium	Gd	1	3	6
106	106	106	Terbiy	Tb	1	3	6
107	107	107	Dysprosium	Dy	1	3	6
108	108	108	Holmium	Ho	1	3	6
43	43	43	Erbium	Er	1	3	6
43	43	62	Erbium	Er	1	3	8
109	109	109	Tulium	Tm	1	3	6
110	110	110	Iterbium	Yb	1	3	6
111	111	111	Lutetium	Lu	1	3	6
24	22	45	Oxygen	O_4	4	-2	
141	141		Solid solution	$Ln_{1-x}Ln_xAsO_4$ -> 1-xLn + xLn + As O_4 -> 1-xLn + xLn + As + 4O	1	0	
28	28		Lanthan	Ln_x	x	3	
26	26		Solid solution	$Ln_{1-x}Ln_xPO_4$ -> 1-xLn + xLn + PO_4 -> 1-xLn + xLn + P	1	0	

Figure 6: Print table of elements

In the forecasting process, data analysis first takes place (*PredAnalysis* function). For each system, it is necessary to ensure that the indicators of each element are obtained from the database and analyze whether it is possible to make a forecast. Therefore, if the parameter *ai* of some *i*-th element from *AN* is not present in the system, the universal value *adef* is chosen, otherwise it is necessary to enter the value *ai*. Let's formally define Analysis products:

$$\text{If } PA(ai) \text{ then } PredAnalysis(ai) \text{ else } PredAnalysis(adef), \tag{32}$$

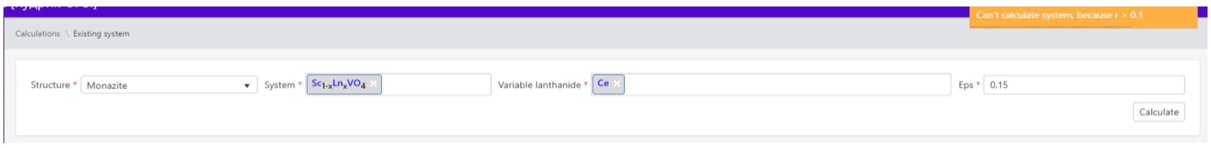
where the predicate *PA*(*ai*) is defined as follows:

$$\exists a_i \mid i \in \{1,2,\ldots,AN\} \rightarrow PA(a_i) = True. \tag{33}$$

In order to be able to visually easily show which errors occurred during data analysis, a function was created – the output of *PredAnalysisErr* errors. The function must be given a certain set of fields: *elm_id* (unique ID of a certain system element), *strs_id* (identifier of the solution structure), *elm_id_ln* (id of a certain lanthanide), *eps* (calculation step). Formally, the error output operator during system analysis

$$O_{analysis} : (elm\_id, strs\_id, elm\_id\_ln, eps) \rightarrow PredAnalysisErr. \tag{34}$$

Figure 7 shows an example of an error output during system analysis.



Can't calculate system, because r > 0.1

Calculations \ Existing system

Structure * Monazite

System * $Sc_{1-x}Ln_xVO_4$

Variable lanthanide * Ce

Eps * 0,15

Figure 7: Error output during system analysis

During data analysis, it is also checked whether such a system with the same set of input data has already been calculated (*PredAnalysisData* function), if it has been calculated, then we output the found data. Analysis and comparison of data from the database, namely in the tables *stored_system* and *stored_result*, is carried out. So, if the *sd* data of some *d*-th element from *SR* is available in the system, we pass the *sd* value, otherwise *sfound*. We formally define the *AnalysisData* product:

$$\text{If } AD(sd) \text{ then } AnalysisData(sd) \text{ else } AnalysisData(sfound), \tag{35}$$

where the predicate *AD*(*sd*) is defined as follows:

$$\exists s_d \mid d \in \{1,2,...,AD\} \rightarrow AD(s_d) = True. \tag{36}$$

A function was also created – *PredAnalysisDataCheck*, which checks whether the system was previously calculated. A certain set of fields must be passed to the function: *elm_id* (unique ID of a certain system element), *strs_id* (solution structure identifier), *elm_id_ln* (id of a certain lanthanide), *eps* (calculation step). Formally, the error output operator during system analysis

$$O_{dataCheck} : (elm\_id,\ strs\_id,\ elm\_id\_ln,\ eps) \rightarrow PredAnalysisDataCheck. \tag{37}$$

After data analysis, the prediction algorithm is selected (*PredTypeAlgorithm* function). The system analyzes and compares the presence of parameters in the element from the database, the following parameters are taken into account: interatomic lengths, sum of interatomic lengths, unit cell volume and ionic radius. Therefore, if the *prmj* parameter of some *j*-th element from PARAM is missing in the system, the algorithm using prmion ionic radii is selected, otherwise the *prmj* algorithm is set. Let's formally define *Algorithm* products:

$$\text{If } AL(prmj) \text{ then } PredTypeAlgorithm(prmj) \text{ else } PredTypeAlgorithm(prmion), \tag{38}$$

where the predicate *AL*(*prmj*) is defined as follows:

$$\exists prm_j \mid j \in \{1,2,...,PARAM\} \rightarrow AL(prm_i) = True. \tag{39}$$

For data processing, after setting the type of forecasting algorithm, the *Prediction* function was created. A certain set of fields must be passed to the function: *elm_id* (unique id of a certain element of the system), *strs_id* (identifier of the structure of the solution), *elm_id_ln* (id of a certain lanthanide), *eps* (calculation step), *algorithm_tp* (algorithm type). Formally, the prediction operator has the form

$$O_{pred} : (elm\_id,\ strs\_id,\ elm\_id\_ln,\ eps,\ algorithm\_tp) \rightarrow Prediction. \tag{40}$$

After data processing, we save the results in specially developed tables *stored_system* and *stored_result*. These tables will later be used in the data analysis phase to check whether calculations have been performed previously with the same input data. So, function *SaveResult* was created.

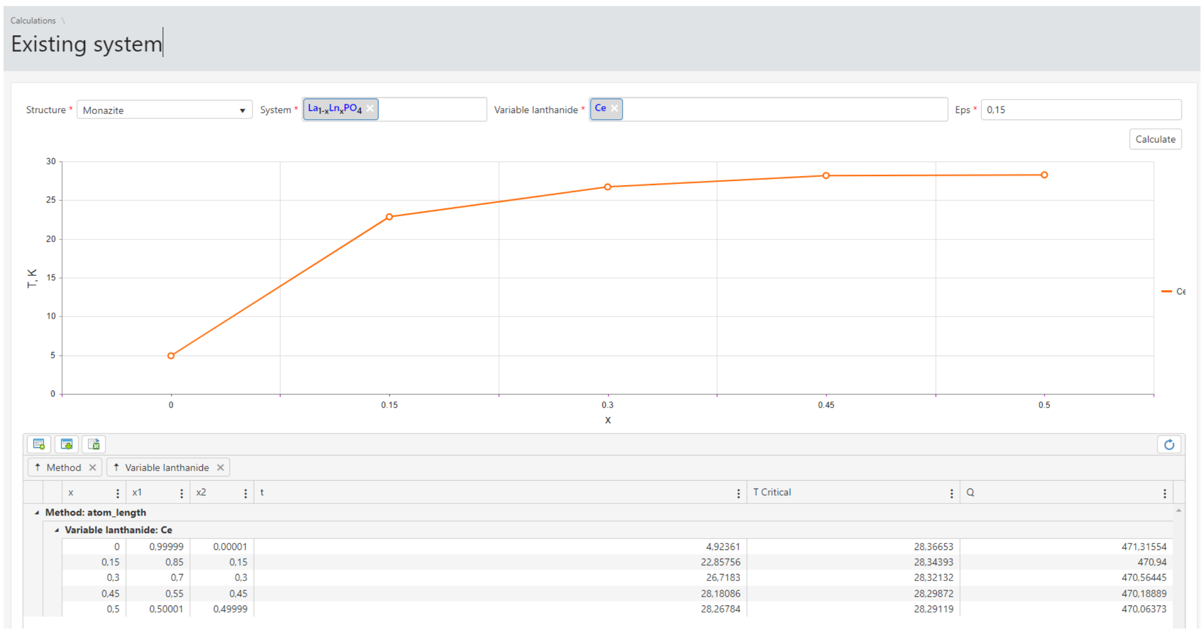
A certain set of fields must be passed to the function: *stds_strs* (system structure identifier), *stds_ln1* (lanthanide 1 identifier), *stds_ln2* (lanthanide 2 identifier), *stds_anion* (anion identifier), *stds_eps* (calculation step), *stdr_stds* (calculated system identifier), *stdr_x* (result x), *stdr_x1* (result x1), *stdr_x2* (result x2), *stdr_t* (result t), *stdr_t_crit* (result t_crit), *stdr_q* (result q), *stdr_method* (result method), *stdr_err_message* (error text). Formally, the prediction operator has the form:

$$O_{saveResult} : (stds\_strs,\ stds\_ln1\ stds\_ln2,\ stds\_anion,\ stds\_eps,\ stdr\_stds,\ stdr\_x,\ stdr\_x1, \\ stdr\_x2\ stdr\_t,\ stdr\_t\_crit,\ stdr\_q,\ stdr\_method,\ stdr\_err\_method) \rightarrow SaveResult. \tag{41}$$

To display the data on the user interface, the *ReturnResult* function was created. It is necessary to transfer a certain set of fields to the function: *stdr_id* (unique ID code of data about the calculated system). Formally, the data display operator on the user interface

$$O_{returnResult} : (stdr\_id) \rightarrow ReturnResult. \tag{42}$$

Figure 8 shows an example of displaying data on the user interface.


Calculations \

Existing system

Structure * Monazite

System * $La_{1-x}Ln_xPO_4$

Variable lanthanide * Ce

Eps * 0,15

Method: atom_length

Variable lanthanide: Ce

x	x1	x2	t	T Critical	Q
0	0,99999	0,00001	4,92361	28,36653	471,31554
0,15	0,85	0,15	22,85756	28,34393	470,94
0,3	0,7	0,3	26,7183	28,32132	470,56445
0,45	0,55	0,45	28,18086	28,29872	470,18889
0,5	0,50001	0,49999	28,26784	28,29119	470,06373

Figure 8: Displaying data on the user interface

4. Conclusions

In this work, for the first time, a knowledge base model was created for predicting the phase stability of solid solutions, which, unlike the existing ones, takes into account the unique limit of substitutions of each lanthanide with the help of a set of production rules, predicates, functions and operators, which allows to increase the accuracy calculation of the energy of mixing and to calculate the exact, and not the general, critical decomposition temperature of the corresponding lanthanide.

5. References

[1] Z. Nagy, Artificial Intelligence and Machine Learning Fundamentals, Packt Publishing, 2018.

[2] O. Naum, L. Chyrun, O. Kanishcheva, V. Vysotska, Intellectual System Design for Content Formation, in: Computer Science and Information Technologies, 2017, pp. 131–138. doi:10.1109/STC-CSIT.2017.8098753.

[3] Z. Ma, Intelligent Databases: Technologies and Applications, Idea Group Publishing, 2006.

[4] P. Jackson, Introduction to Expert Systems, 2nd ed., Addison Wesley, 2001.

[5] D. Calvaneze, Optimizing Ontology-Based Data Access, Technical University Vienna, 2013. URL: https://pdfs.semanticscholar.org/f53a/40dc026d442cf5058e5b6a4b5c9f2f522d6b.pdf.

[6] A. T. Coerbett, J. R. Anderson, Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge 4 (1995) 253–278. doi:10.1007/BF01099821.

[7] O. Bisikalo, I. Bogach, V. Sholota, The Method of Modelling the Mechanism of Random Access Memory of System for Natural Language Processing, in: IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2020, pp. 472–477. doi: 10.1109/TCSET49122.2020.235477.

[8] A. Alamri, The Relational Database Layout to Store Ontology Knowledge Base, in: International Conference on Information Retrieval & Knowledge Management, Kuala Lumpur, Malaysia, 2012, pp. 74–81. doi: 10.1109/InfRKM.2012.6205039.

[9] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (1997) 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[10] O. V. Kudryk O. V. Bisikalo, Yu. A. Oleksii, S. V. Radio, Intelligent Information System for Predicting Chemicals with Interactive Possibilities, in: Computational Linguistics and Intelligent Systems. CoLInS 2021. URL: http://ceur-ws.org/Vol-2870/paper68.pdf.

[11] V. Vysotska, V. Lytvyn, Y. Burov, P. Berezin, M. Emmerich, V. B. Fernandes, Development of Information System for Textual Content Categorizing Based on Ontology, in: CEUR Workshop Proceedings, 2019. URL: https://ceur-ws.org/Vol-2362/paper6.pdf

[12] N. Khairova, S. Petrasova, A. P. S. Gautam, The Logical-Linguistic Model of Fact Extraction from English Texts, in: Proceedings of the International Conference on Information and Software Technologies, 2016, pp. 625–635. doi:10.1007/978-3-319-46254-7_51.

[13] O. Orobinska, J. H. Chauchat, N. Sharonova, Methods and Models of Automatic Ontology Construction for Specialized Domains (case of the Radiation Security), in: Proceedings of the 1st International Conference Computational Linguistics and Intelligent Systems, Kharkiv, Ukraine, 2017, pp. 95–99. URL: https://oldena.lpnu.ua/bitstream/ntb/39462/1/012-095-099.pdf.

[14] O. Bisikalo, Yu. Ivanov, V. Sholota, Modeling the Phenomenological Concepts for Figurative Processing of Natural-Language Constructions, in: Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Systems, Kharkiv, Ukraine, 2019, pp. 1–11. URL: http://ceur-ws.org/Vol-2362/paper1.pdf.

[15] M. Nokel, N. Loukachevitch, An Experimental Study of Term Extraction for Real Information-Retrieval Thesauri, in: Proceedings of 10th International Conference on Terminology and Artificial Intelligence, Paris, France, 2013, pp. 69–76.