Model Domain Mapping Function and Its Role in Model Composition

Authors: Liu Zhimu, Chen Zhen, Wang Xiaoshan, Liu Hui, Liu Zhimu

Date: 2025-12-03T00:00:00+00:00

Abstract

Based on the requirements of model composition on model domains, this paper further decomposes the model domain into variable domains and function codomains, and argues that the essence of model-restricted union operations lies in preserving the domains of functions and relations unchanged during the union operation, thereby ensuring that the functions and relations involved in the union model remain valid. Building upon this domain analysis, this paper proposes a domain mapping function method, which establishes a mutual data reference mechanism between models and enhances the flexibility and efficiency of data processing in model composition. Simultaneously, this paper also demonstrates that the domain mapping function method does not impede the implementation of model-restricted union operations, and analyzes the factors influencing the stability of various model compositions by the domain mapping function method along with their corresponding solutions.

Full Text

Model Domain Mapping Function and Its Role in Model Combinations

Liu Zhimo[1], Chen Zhen[2], Wang Xiaoshan[2], Liu Hui[3]
[1] (Xi' nan Computer Corp., Chongqing, 400060, China)
[2] (Beijing Institute of Technology, Beijing, 100081, China)
[3] (China Ordnance Industry Computer Application Technology Research Institute, Beijing, 100089, China)

Abstract: Based on the requirements of model combinations for model domains, this paper further decomposes the model domain into variable definition domains and function value ranges. It demonstrates that the essence of model restricted union operations is maintaining the definition domains of functions and relations unchanged during the union operation to ensure the validity of

functions and relations in the combined model. Building upon domain analysis, this paper proposes a domain mapping function method, establishes a mechanism for mutual data referencing between models, and enhances the flexibility and efficiency of data processing in model combinations. Meanwhile, the paper also discusses that the domain mapping function method does not affect the implementation of model restricted union operations, and analyzes the influencing factors and solutions for the stability of various model combinations.

Keywords: model domain, variable definition domain, function range, domain mapping function, data circular reference

1. Introduction

Artificial intelligence has entered a period of rapid development. Large models such as Transformer, BERT, ChatGPT-3, and ChatGPT-4 have emerged successively [1][2][3], achieving remarkable progress in natural language and multimodal information processing, enabling the processing and transformation of text, programming, images, speech, video, music, and other multimodal information. Generative AI technology built upon this foundation has made notable advances. Currently, pre-trained large models developed on the Transformer architecture can tune over one hundred billion parameters for deep learning, becoming the mainstream direction in AI. In recent years, China has also seen the emergence of large model applications represented by DeepSeek, achieving remarkable results and becoming a global focus in AI [4].

However, it should also be recognized that these large models learn by adjusting internal parameters, exhibiting high complexity and nonlinearity, with problems of interpretability and reproducibility [5]. Questions about their underlying principles and whether they can be directly applied to other intelligent agent device development are concerns for many, yet obtaining knowledge directly from current large models is quite difficult. Externally, they possess needed functionalities but lack specific mathematical formulas to describe them, with insufficient reproducibility, making them black boxes with specific functions [6][7]. People expect deeper theoretical and mathematical descriptions. As research in AI fields such as multimodal large language models, knowledge graphs, AI and multimodal mathematical reasoning, embodied intelligence, and cross-domain applications advances rapidly, several foundational theoretical studies need to keep pace to support faster and better development of AI technologies.

Undoubtedly, mathematical logic theory is an important component of current AI foundational theory. Model theory in mathematical logic provides strict definitions of model concepts and establishes a rigorous theoretical system that can serve as foundational theory for AI research. The model concept therein is defined through interpretation (mapping) by formal language, containing several functions and relations to form a complete logical whole. Since such models are strictly defined through formal language and satisfy formal language inference rules, the logical inference conclusions derived from them should apply to general

entity model reasoning. Therefore, studying mathematical logic models has important practical significance for AI system development.

Regarding models, a desirable scenario is that several simple models can be combined into larger models, or conversely, complex larger models can be decomposed into several smaller simple models for solution. This way, relatively simple and mature models can combine to produce complex, larger models, which is significant for AI systems and independent intelligent agent research. Achieving model combination requires model union operations. However, in mathematical logic model theory, model union operations generally do not satisfy closure requirements [8], making them undefinable and thus unimplementable. Existing model theory has not deeply studied this, representing a gap. In our paper "A Tree-Shaped Model Chain Construction Method and Its Application Prospects" [9], we proposed a model restricted union operation method in mathematical logic model theory, enabling model union operations under certain constraints, and discussed modeling approaches for combining small models into large models or decomposing large models into small ones. This provides theoretical support for AI device research. However, the validity of model restricted union operations is achieved through forced restrictions on domains during union operations, which seems somewhat inflexible for AI system construction and affects system operational efficiency. Therefore, this paper further explores and analyzes this issue from the perspective of model domains to improve the flexibility and operational efficiency of model combinations.

2. Review of Model Restricted Union Operations

To facilitate in-depth discussion of modeling issues using mathematical logic theory, we first briefly review the relevant mathematical logic model theory and model combination problems.

2.1 First-Order Language Symbol System

The first-order logic elements involved in this paper include: domain M, individual constant set C, relation symbol set R, function symbol set F, variable set $\{x, y, ...\}$, logical operator set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$, quantifier set $\{\forall, \exists\}$, and punctuation set $\{(,), , \}$. These can be found in general mathematical logic texts.

Except for the domain, the remaining sets constitute the first-order formal language $\mathcal{L}$:

$$\mathcal{L} = \{a, \dots, R, \dots, \approx, f, \dots, x, \dots, u, \dots, \neg, \dots, \forall, \exists, (,), , \}$$

where $\{\approx, x, \dots, u, \dots, \neg, \dots, \forall, \exists, (,), , \}$ are common across different systems and generally not listed in specific systems.

Definition 2.1: Let $C = \{c_1, c_2, \dots, c_\kappa\}$ be a set of individual constants, $R = \{r_1, r_2, \dots, r_\lambda\}$ be a set of relation symbols, and $F = \{f_1, f_2, \dots, f_\mu\}$ be a set of function symbols. Then $\mathcal{L} = \{C, R, F\} =$

$\{c_1, c_2, \ldots, c_\kappa, r_1, r_2, \ldots, r_\lambda, f_1, f_2, \ldots, f_\mu\}$ constitutes a specific language, where the cardinalities κ, λ, μ of C, R, F can be finite or infinite.

The definitions and formation rules of terms, formulas, sentences, and other elements of first-order logic used in this paper, as well as variable interpretation in domains, follow general literature [8][10][11][12].

2.2 Model Definition

Mathematical logic models are structural models. For discussion convenience, relevant definitions are listed in detail.

Definition 2.2: Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-structure is a binary pair $\mathcal{M} = (M, I)$, where M is a non-empty set called the domain of $\mathcal{M}$, and I is a mapping defined on $\mathcal{L}$ satisfying: (1) I interprets each constant symbol c in $\mathcal{L}$ as an element of M, i.e., $I(c) \in M$; (2) I interprets each n-ary relation symbol P in $\mathcal{L}$ as an n-ary relation on M, i.e., $I(P) \subseteq M^n$; (3) I interprets each n-ary function symbol f in $\mathcal{L}$ as an n-ary function on M, i.e., $I(f)$ is a mapping from M^n to M, $I(f) : M^n \to M$.

We often denote $I(c), I(P), I(f)$ as c^M, P^M, f^M respectively. Thus for such an $\mathcal{L}$-structure, we have $c^M \in M$, $P^M \subseteq M^n$, $f^M : M^n \to M$.

In some literature [8][12], the binary pair $\mathcal{M} = (M, I)$ is considered a model defined for formula A or formula set Σ in $\mathcal{L}$, i.e., when $\mathcal{M} \models A$ (or $\mathcal{M} \models \Sigma$), $\mathcal{M}$ is defined as a model of A (or Σ). In other literature [10], the binary pair (M, I) is viewed as an interpretation of language $\mathcal{L} = \{C, R, F\}$ on M. When this pair has interpreted $\mathcal{L}$ as $\{\bar{C}, \bar{R}, \bar{F}\}$, then $\mathfrak{A} = (M, \bar{C}, \bar{R}, \bar{F})$ is called a model of language $\mathcal{L}$, and M is called the domain of $\mathcal{L}$.

Structurally, the two are actually consistent. This paper only explores logical model problems from a data structure perspective, so we adopt the method from [10], treating the structure after interpretation on the domain as the language model.

Definition 2.3: Let language $\mathcal{L} = \{C, R, F\}$. If the binary pair (M, I) interprets the individual, relation, and function symbols of $\mathcal{L}$ in M as $\{\bar{C}, \bar{R}, \bar{F}\}$, then $\mathcal{M} = (M, \bar{C}, \bar{R}, \bar{F})$ is called a model of $\mathcal{L}$, and M is called the domain of $\mathcal{M}$ [10].

In the following discussion, when mentioning models of language $\mathcal{L} = \{C, R, F\}$, we assume by default that the individual, relation, and function symbols of $\mathcal{L}$ have been interpreted in M. For convenience, we also refer to $\mathcal{M} = (M, C, R, F)$ as a model of $\mathcal{L}$.

From the above definitions of mathematical logic models, the structural characteristics are evident. However, for general models in practical applications that target specific real-world objects and handle concrete problems, the two cannot be equated. Although mathematical logic models are structural models, they contain abstract data objects, logical relations and inference rules between

objects, and functional mapping relations between objects. These objects, relations, and functions also constitute a logical whole. Models of concrete objects generally also contain objects, data, functions, and inference methods, but involve specific object entities. While not equivalent to mathematical logic models, the modeling rules and inference rules obtained from mathematical logic model theory can be regarded as logical principles that should be followed in concrete model applications. Therefore, studying mathematical logic models has important practical significance for concrete model research.

2.3 Submodels

Definition 2.4: Let $\mathcal{M}$ and $\mathcal{N}$ be two models of the same language $\mathcal{L} = \{C, R, F\}$. We say $\mathcal{M}$ is a submodel of $\mathcal{N}$, denoted $\mathcal{M} \subseteq \mathcal{N}$, if: (1) $M \subseteq N$ (M, N are the domains of $\mathcal{M}, \mathcal{N}$ respectively); (2) For any constant symbol $c \in \mathcal{L}$, $c^M = c^N$; (3) For any n-ary relation symbol $R \in \mathcal{L}$ and any $a_1, \dots, a_n \in M$, $R^M(a_1, \dots, a_n)$ holds if and only if $R^N(a_1, \dots, a_n)$ holds; (4) For any n-ary function symbol $f \in \mathcal{L}$ and any $a_1, \dots, a_n \in M$, $f^M(a_1, \dots, a_n) = f^N(a_1, \dots, a_n)$.

For condition (2), it is implicitly required that $c^M \in M$.

2.4 Model Restricted Union Operation and Model Combination

A primary purpose of studying model problems is to achieve model combination. To realize model combination, model series and parallel connections are necessary, thereby fulfilling the vision of combining small models into larger, more complex ones. Reference [9] provides detailed discussion. For convenience, relevant definitions are compiled below.

For model intersection operation, we have:

Definition 2.5: Let models $\mathcal{M}_1, \mathcal{M}_2$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$, where $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. Define $\mathcal{M}_0 = \mathcal{M}_1 \cap \mathcal{M}_2 = (M_0, C_0, R_0, F_0)$ as their intersection operation, satisfying: (2) For any constant symbol $c \in \mathcal{L}$, $c^{M_0} = c^{M_1}$ and c^{M_2}; (3) For any n-ary relation symbol $R \in \mathcal{L}$ and any $a_1, \dots, a_n \in M_0$, $R^{M_0}(a_1, \dots, a_n)$ holds if and only if ($R^{M_1}(a_1, \dots, a_n)$ and $R^{M_2}(a_1, \dots, a_n)$); (4) For any n-ary function symbol $f \in \mathcal{L}$ and any $a_1, \dots, a_n \in M_0$, $f^{M_0}(a_1, \dots, a_n) = (f^{M_1}(a_1, \dots, a_n)$ and $f^{M_2}(a_1, \dots, a_n))$.

Reference [9] shows that model intersection operation is closed under constants, relations, and functions.

For model union operation, closure generally does not hold. Therefore, reference [9] proposed the model restricted union operation method to achieve model union operations under certain constraints.

Definition 2.6 (Model Restricted Union Operation): Let models $\mathcal{M}_1, \mathcal{M}_2$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$, where $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. Define $\mathcal{M}_0 = \mathcal{M}_1 \cup_{\lim} \mathcal{M}_2 =$

(M_0, C_0, R_0, F_0) as the restricted union operation of models $\mathcal{M}_1, \mathcal{M}_2$, if the following conditions hold: (2) For any constant symbol $c \in \mathcal{L}$, if $c^{M_1} = c^{M_0}$, then $c \in C_1$ and $c \in C_0$; if $c^{M_2} = c^{M_0}$, then $c \in C_2$ and $c \in C_0$; (3) For any n-ary relation symbol $R \in \mathcal{L}$ and any $a_1, \ldots, a_n$, if $a_1, \ldots, a_n \in M_1$, then $R^{M_1}(a_1, \ldots, a_n)$ holds if and only if $R^{M_0}(a_1, \ldots, a_n)$ holds; if $a_1, \ldots, a_n \in M_2$, then $R^{M_2}(a_1, \ldots, a_n)$ holds if and only if $R^{M_0}(a_1, \ldots, a_n)$ holds; (4) For any n-ary function symbol $f \in \mathcal{L}$ and any $a_1, \ldots, a_n$, if $a_1, \ldots, a_n \in M_1$, then $f^{M_1}(a_1, \ldots, a_n) = f^{M_0}(a_1, \ldots, a_n)$; if $a_1, \ldots, a_n \in M_2$, then $f^{M_2}(a_1, \ldots, a_n) = f^{M_0}(a_1, \ldots, a_n)$; (5) For any n-ary variable value combination $a_1, \ldots, a_i, \ldots, a_n \in M_0$, its value range only allows cases where $(\forall a_i \in M_1) \vee (\forall a_i \in M_2)$ (including $a_i \in M_1 \cap M_2$).

These conditions are called model restricted union operation conditions. Reference [9] shows the following lemma holds:

Lemma 2.1: Model restricted union operation is closed under relations and functions.

Reference [9] also lists definitions of submodels, model intersection operation methods, and defines submodel union operation methods.

Definition 2.7: Let models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$ of first-order language $\mathcal{L} = \{C, R, F\}$ be submodels of model $\mathcal{M}_0 = (M_0, C_0, R_0, F_0)$, with $\mathcal{M}_1 \subseteq \mathcal{M}_0$, $\mathcal{M}_2 \subseteq \mathcal{M}_0$. Define $\mathcal{M}_0' = (M_0', C_0', R_0', F_0') = \mathcal{M}_1 \cup_{\text{sub}} \mathcal{M}_2$ as the union model of submodels $\mathcal{M}_1, \mathcal{M}_2$, if: (2) The union operation of $\mathcal{M}_1, \mathcal{M}_2$ satisfies the model restricted union operation conditions in Definition 2.6; (3) $M_0' \subseteq M_0$.

Reference [9] points out that the submodel union operation $\cup_{\text{sub}}$ defined in Definition 2.7 is constrained by the restricted model union operation $\cup_{\text{lim}}$, and $\cup_{\text{sub}}$ better reflects the submodel characteristics of the union operation result.

3. Model Domain Decomposition

3.1 Requirements of Model Combination for Domains

As mentioned, the purpose of model combination is to combine several small models into larger ones to handle more complex problems. Such combinations should be meaningful, internally coordinated, and contradiction-free, rather than arbitrary patchwork. A prominent advantage of model combination is that models can mutually utilize each other' s data processing results—some models' data processing results are treated as intermediate results by the larger model and utilized by other models. This is clearly valuable for developing large model systems. Intuitively, for a model to utilize other models' data processing results, it must share the processed data from other models to achieve inter-model data transfer. From a model theory perspective, models capable of mutual data transfer must have intersecting data domains, leading to the following lemma:

Lemma 3.1: Let $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$ and $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$. If direct data transfer is to be realized between $\mathcal{M}_1$ and $\mathcal{M}_2$, the necessary condition is that the intersection of their domains M_1 and M_2 must be non-empty, i.e., $M_1 \cap M_2 \neq \emptyset$.

For model restricted union operations, the value range of n-ary variables is limited during union operations, which seemingly restricts the usable domain range, necessitating further in-depth discussion.

3.2 Domain Decomposition

Definition 3.1: Let $\mathcal{M} = (M, C, R, F)$ be a model of first-order language $\mathcal{L} = \{C, R, F\}$, with M as the domain of $\mathcal{M}$. Then $M = M_C \cup M_D \cup M_R$ is called the domain decomposition of M, where: M_C is the constant domain, containing general constants $a, b, ...$ and logical constants T (True), F (False); M_D is the definition domain, the value range of variables in the model; M_R is the range, the value range of function outputs in the model.

In Definition 3.1, M_R is only defined for functions because relations are logical expressions with only two possible values (true/false), which can be included in logical constants $\{T, F\}$.

Model definition domains and value ranges can be further decomposed.

Definition 3.2: The model definition domain M_D can be further decomposed as $M_D = M_{DR} \cup M_{DF}$, where: M_{DR} is the relation definition domain, the value range of variables in all relation expressions of the model; M_{DF} is the function definition domain, the set of variable values in all functions of the model.

In a model, the value ranges of relation variables $(x_1, x_2, ..., x_n)$ and function variables $(y_1, y_2, ..., y_m)$ may overlap, i.e., $M_{DR} \cap M_{DF} \neq \emptyset$ may exist.

Generally, a model may contain not just one function but a function set $F = \{f_i\}(1 \leq i \leq n)$. The definition domain of function f_i can be expressed as M_{Df_i}. Then according to Definition 2.2, we have the following lemma for function definition domains:

Lemma 3.2: Let the function set of model $\mathcal{M}$ be $F = \{f_i\}(1 \leq i \leq n)$, with function f_i' s definition domain as M_{Df_i}. Then the function definition domain of model $\mathcal{M}$ is $M_{DF} = \bigcup M_{Df_i}$.

Definition 3.3: The value range of function f is called the range of f, denoted M_{Rf}.

Based on Definition 3.3, we have the following lemma:

Lemma 3.3: Let the function set of model $\mathcal{M}$ be $F = \{f_i\}(1 \leq i \leq n)$. Then the function range of model $\mathcal{M}$ is $M_{RF} = \bigcup M_{Rf_i}$, and the model range $M_R = M_{RF}$.

In mathematical logic, both function definition domains and value ranges are structural definitions. Unlike concrete objects, they have broader meanings. For instance, regarding function definition domains: from a data attribute perspective, data in definition domains can be ordinary data or object data; from a data type perspective, data can be real numbers, imaginary numbers, integers; from a data structure perspective, data can be single data, vectors, matrices. For a concrete model, its data attributes, types, and structures in the definition domain are fixed, while for general models, these need not be specified—we only study definition domain characteristics from a structural perspective.

3.3 Requirements of Function Closure for Definition Domains and Value Ranges

After subdividing model domains into constant domains, definition domains, and value ranges, we can gain deeper understanding of function non-closure problems caused by general model union operations.

Reference [9] describes the function non-closure problem in model union operations as follows: Let models $\mathcal{M}_1, \mathcal{M}_2$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$, where $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. If we define $\mathcal{M}_0 = \mathcal{M}_1 \cup \mathcal{M}_2 = (M_0, C_0, R_0, F_0)$, we first require $M_0 = M_1 \cup M_2$. For some n-ary function $f(x_1, \dots, x_n) \in \mathcal{L}$ interpreted in both $\mathcal{M}_1$ and $\mathcal{M}_2$, for value combinations $\{(a_1, \dots, a_n) | a_i \in M_0\}$, there are several cases: (1) $f^{M_0}(a_1, \dots, a_n) \in M_0$, operation is closed; (2) $f^{M_0}(a_1, \dots, a_n) \in M_0$, operation is closed; (3) $M_1 \cap M_2$. In this case, the value of $f^{M_0}(a_1, \dots, a_i, \dots, a_j, \dots, a_n)$ cannot be determined to be in $M_0 = M_1 \cup M_2$, causing function non-closure.

Due to function non-closure, model union operations generally cannot be defined. From the function definition domain perspective, in cases (1) and (2) above, the arguments of functions $f^{M_1}(a_1, \dots, a_n)$ and $f^{M_2}(a_1, \dots, a_n)$ are within the function' s definition domain, so the functions of both models and the union model are defined within this range. In case (3), the arguments of function $f^{M_0}(a_1, \dots, a_i, \dots, a_j, \dots, a_n)$ exceed the definition domain ranges of known functions $f^{M_1}(a_1, \dots, a_n)$ and $f^{M_2}(a_1, \dots, a_n)$, thus causing the problem of undetermined function validity, i.e., function non-closure.

The function non-closure situation in model union operations can be extended to model relation operations. Since relation expressions in models also have variable definition domain issues, general model union operations may cause exceeding relation variable definition domain ranges, leading to undetermined relation expression validity.

From the above analysis, the following theorem holds:

Theorem 3.1: The essence of the model restricted union operation method is maintaining function and relation definition domains unchanged during union operations to preserve the validity of functions and relations in the union operation.

Theorem 3.1 explicitly states the essence and significance of model restricted union operations. From Theorem 3.1' s perspective, model restricted union operations can also be called domain-preserving model union operations.

3.4 Domain Extension in Single Models and Model Combinations

In mathematical logic model theory, maximization problems are often encountered. After subdividing model domains, the possibility of maximization becomes obvious. For any single model $\mathcal{M}$, its domain $M = M_C \cup M_D \cup M_R$. Assuming a member $m \in M_R$ and $m \notin M_D$, if we let $M'_D = M_D \cup \{m\}$, this is arbitrary domain expansion that generally does not hold. If domain expansion is needed, the expanded members must be verified from model functions and relations. Only when $M_R \subseteq M_D$ can we finally have $M'_D = M_D \cup \{m\}$.

In practical applications, using mature models often encounters functional expansion problems. One is expanding model function capabilities in new data environments to handle new data and enhance processing capabilities; the other is adapting to new model environments to work collaboratively with other models to complete tasks. Both may involve model modifications. By modifying model implementation functions to adapt to changes in function and relation definition domains and function value ranges, the result is improved model functionality and performance. To confirm these functional changes, the most direct method is verification through model usage. If verification shows these modifications are feasible, the model changes are successful; otherwise, they are not feasible.

For single models, domain changes should be handled cautiously, and domains generally cannot be arbitrarily expanded, especially definition domains. However, for model combinations, the situation differs. For combined models achieved through model restricted union operations, their domains should at least contain the domains of all submodels in the combination, naturally expanding domain scale.

Let combined model $\mathcal{M}_0 = (M_0, C_0, R_0, F_0)$ be a union model composed of several submodels: $\mathcal{M}_0 = (\cup_{\text{sub}})_{i=1}^{n}(M_i, C_i, R_i, F_i)$, i.e., model combination $\mathcal{M}_0$ is composed of submodels $\mathcal{M}_i (i = 1, \ldots, n)$ through model restricted union operations, where the domain of $\mathcal{M}_0$ is $M_0 = \bigcup M_i$.

Clearly, the domain of combined model $\mathcal{M}_0$ is expanded compared to each submodel. However, since the model union operation here is restricted, by Theorem 3.1, the functions and relations of each submodel in combined model $\mathcal{M}_0$ will not exceed their definition domain ranges during operations. Therefore, the combined model is valid within its domain $M_0 = \bigcup M_i$.

For union models not composed through model restricted union operations, since union operations generally may not hold, the union model may not be valid. The reason is that as domains expand during union operations, submodels' definition domains exceed their originally determined ranges, causing functions and

relations in the union operation to potentially not hold. Therefore, for practical situations requiring non-restricted union operations, verification methods are needed to confirm whether the union model holds and for further model modifications.

4. Domain Mapping Function and Its Role in Model Combinations

4.1 Domain Mapping Function

As mentioned, our purpose for model combination is to enhance model problem-processing capabilities, combining several small mature models into larger, more complex models to handle more complex practical problems. Of course, these combined models should form an organic whole, working collaboratively to fully leverage the overall performance of the model combination. To achieve this, models must be able to share and utilize each other' s processed data results—some models can use other models' data processing results to enhance their own problem-processing capabilities. Under this interaction mechanism, the overall performance of the entire model combination is improved, which is exactly what we expect from model combination.

Reference [9] proposed the model restricted union operation method to achieve model combination. As discussed above, the essence of model restricted union operations is maintaining function and relation definition domains unchanged during model union operations, which seemingly restricts the application scope of model restricted union operations. In practical applications, to solve the problem of limited application scope due to domain restrictions in general model union operations, there are two solutions: for pure software systems, needed submodel modules can be added anywhere in the model combination to achieve model combination; for systems composed of both hardware and software, such as embodied intelligent systems, it' s generally impossible to combine and load a submodel module through "copying."Instead, we hope to achieve functional combination by referencing the processed data of that model module where needed. Especially in complex model networks composed of numerous submodels, using copying to obtain its data processing results is neither advisable nor feasible. In view of this, this paper proposes a domain mapping function method to enable one model to reference data processed by other models.

Definition 4.1: Let models $\mathcal{M}_1, \mathcal{M}_2$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$, where $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. Let $M_{1k} \subseteq M_1$ be a subdomain of M_1, and $M_{2p} \subseteq M_2$ be a subdomain of M_2. The function set $F_{M_2-M_1}$ is called a domain mapping function from domain M_2 to M_1 if $F_{M_2-M_1}$ maps M_{2p} to M_{1k}, i.e., $F_{M_2-M_1} : M_{2p} \to M_{1k}$.

From Definition 4.1, domain mapping functions map part of other models' domains into this model' s domain, providing the possibility for this model to directly reference other models' data processing results. Domain mapping is

essentially data correspondence or transformation, converting some data forms into others. Of course, such transformations should have definite meanings and be feasible in practical applications. Definition 4.1 does not specifically emphasize whether domain mapping is between value ranges or definition domains, considering that when mapping value ranges to another model' s definition domain, simultaneous mapping of part of the definition domain is also allowed.

Generally, M_{2p} in Definition 4.1 mainly refers to certain function value ranges of model $\mathcal{M}_2$, while M_{1k} mainly refers to function and relation definition domains in model $\mathcal{M}_1$.

For example, calling another software module in a software system can be viewed as mapping the called module' s output data into the calling module' s definition domain for continued use. In this case, the function mapping can essentially be implemented through module calls. Another example: if a system contains both binary logic and fuzzy logic systems, the fuzzy logic system' s processing results can be adopted by the binary logic system. Here, the fuzzy logic system' s output data must undergo appropriate transformation before being passed to the binary system. The function mapping then includes referencing fuzzy logic system output data and appropriate data transformation [13]: when mapping fuzzy logic system variable x_A's value range to binary logic system x_B's definition domain, the membership degree μ_{x_A} corresponding to x_A must be transformed from value range $[0,1]$ to x_B' s input values $\{0,1\}$, with transformation $x_{B-In} = \{...\}$.

Since domain mapping functions are introduced in model combinations, it is necessary to appropriately modify model definitions to reflect the role and positioning of domain mapping functions in models.

Definition 4.2 (Model Definition with Domain Mapping Function): Let a model of first-order language $\mathcal{L} = \{C, R, F\}$ be $\mathcal{M} = (M, C, R, F)$. We call $\mathcal{M}' = (M, C, R, F \cup F_{M_o - M})$ the model after incorporating domain mapping functions into $\mathcal{M}$.

Here, domain mapping function $F_{M_o - M}$ is a mapping function set satisfying Definition 4.1. Its role is only to map part of other models' domains into model $\mathcal{M}$' s domain M, without participating in other data processing within model $\mathcal{M}$.

For convenience, when we later mention models containing domain mapping functions, we directly use $\mathcal{M} = (M, C, R, F \cup F_{M_o - M})$, where the model' s function set is augmented with domain mapping function $F_{M_o - M}$ to distinguish it from general models. For general models not using domain mapping functions, we still use the form $\mathcal{M} = (M, C, R, F)$.

After adding domain mapping functions to a model, this model' s functions and relations can reference data from other models mapped into this model' s definition domain. Of course, when referencing data from other models mapped into this model' s definition domain, users should understand that these data

correspond to data in the referenced model and utilize them accordingly. This referencing of other models' output data effectively expands the data processing range of the model, potentially achieving overall performance improvement.

Meanwhile, to achieve performance improvement, partial modifications to relation sets and function sets may be involved. This issue of modifying relations and functions in the model involves specific model composition, which is beyond this paper' s scope and left for future research.

4.2 Impact of Domain Mapping Functions on Model Restricted Union Operations

After introducing domain mapping functions into models, the first question that comes to mind is whether model restricted union operations still hold after adding domain mapping functions. The following theorem gives an affirmative answer.

Theorem 4.1: After introducing domain mapping functions into models, if the models' relations and functions remain unchanged, then model restricted union operations based on such models still hold valid.

Proof: Let $\mathcal{M}_1, \mathcal{M}_2$ be two models of first-order language $\mathcal{L} = \{C, R, F\}$, where $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. Let $\mathcal{M}_0 = \mathcal{M}_1 \cup_{\lim} \mathcal{M}_2 = (M_0, C_0, R_0, F_0)$ be the model restricted union operation result of $\mathcal{M}_1, \mathcal{M}_2$ satisfying Definition 2.6 conditions. According to Lemma 2.1, model restricted union operations based on $\mathcal{M}_1, \mathcal{M}_2$ are closed under relations and functions, i.e., the relations and functions R_0, F_0 in $\mathcal{M}_0$ hold and have definite values after union operations.

Now introduce domain mapping function $F_{M_2-M_1} : M_{2r} \to M_{1q}$ into $\mathcal{M}_1$, where $M_{1q} \subseteq M_1$ is a subdomain of M_1, and $M_{2r} \subseteq M_2$ is a subdomain of M_2. $F_{M_2-M_1}$ maps model $\mathcal{M}_2$' s subdomain M_{2r} to model $\mathcal{M}_1$' s subdomain M_{1q}. At this point, model $\mathcal{M}_1$ becomes $\mathcal{M}_1' = (M_1, C_1, R_1, F_1 \cup F_{M_2-M_1})$, while $\mathcal{M}_2$ remains unchanged. Next, examine the restricted union operation $\mathcal{M}_0' = \mathcal{M}_1' \cup_{\lim} \mathcal{M}_2$ of models $\mathcal{M}_1'$ and $\mathcal{M}_2$. Since domain mapping function $F_{M_2-M_1}$ only maps $\mathcal{M}_2$' s subdomain M_{2r} to $\mathcal{M}_1$'s subdomain M_{1q}, compared to $\mathcal{M}_1$, $\mathcal{M}_1'$ has not changed in other aspects; $M_0' = M_0$, $C_0' = C_0$, $R_0' = R_0$, and $F_0' = F_0 \cup F_{M_2-M_1}$. Thus, for union model $\mathcal{M}_0'$, M_0 is unchanged; R_0 and F_0 are closed with definite values, also unchanged; only F_0' has changed, adding $F_{M_2-M_1}$ with definite meaning. Therefore, we can assert that union operation $\mathcal{M}_0' = \mathcal{M}_1' \cup_{\lim} \mathcal{M}_2$ holds valid.

According to Theorem 4.1, after introducing domain mapping functions into model combinations, model restricted union operations remain functionally closed and can be used confidently from a union operation perspective. However, if domain mapping functions are adopted in more than one place in model combinations, is the scheme still feasible? According to Theorem 4.1, we can be certain that model restricted union operations can still be performed. But model combinations are wholes, and after introducing domain mapping

functions in multiple places, the overall performance of model combinations may be affected. Therefore, it is necessary to deeply analyze this situation.

4.3 Restrictions and Influences of Model Definition Domains and Value Ranges on Domain Mapping Functions

Generally, domain mapping functions map other models' value range data into this model' s definition domain data for this model to adopt other models' data processing results. Regarding the correlation degree of domain data between two related models, there are cases of domain overlap and non-overlap. Let models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$ and $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$. Now introduce domain mapping function $F_{M_2-M_1}$ into $\mathcal{M}_1$, making model $\mathcal{M}_1$ become $\mathcal{M}_1 = (M_1, C_1, R_1, F_1 \cup F_{M_2-M_1})$, where $F_{M_2-M_1} : M_{2p} \to M_{1k}$. We discuss separately based on whether M_{2p} and M_{1k} overlap.

Case 1: $M_{2p} \cap M_{1k} \neq \emptyset$, where overlap exists. This is further divided into two subcases.

Subcase 1.1: $M_{2p} \subseteq M_{1k}$. For any $a \in M_{2p}$, there exists $b \in M_{1k}$ such that $a = b$, thus $a \in M_{1k}$. Here $\mathcal{M}_1$ can directly reference data in M_{2p}. The following theorem holds:

Theorem 4.2: Let models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1 \cup F_{M_2-M_1})$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$, where $\mathcal{M}_1$ contains domain mapping function $F_{M_2-M_1} : M_{2p} \to M_{1k}$, and $M_{2p} \subseteq M_{1k}$. Then $\mathcal{M}_1$ can directly use data from $\mathcal{M}_2$ through the domain mapping function.

This is the most ideal and concise situation.

Subcase 1.2: $M_{2p} \supset M_{1k}$. There exists $a \in M_{2p}$ such that $a \notin M_{1k}$. In this case, when referencing value range data obtained from $\mathcal{M}_2$ through the domain mapping function in $\mathcal{M}_1$, there exists a situation exceeding $\mathcal{M}_1$' s definition domain. Direct usage would cause function non-closure and relation invalidity. Therefore, to make referenced data usable in $\mathcal{M}_1$, M_{2p} data must be compressed.

Now implement this through a data compression function F_{comp} that compresses M_{2p} to not exceed M_{1k}' s range, i.e., $F_{\text{comp}} : M_{2p} \to M'_{2p}$, making $M'_{2p} \subseteq M_{1k}$ hold. Then reference $\mathcal{M}_2$' s processed data in $\mathcal{M}_1$ through domain mapping function $F_{M_2-M_1} : M'_{2p} \to M_{1k}$. According to Theorem 4.1, this is feasible. Thus we obtain:

Theorem 4.3: Let models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1 \cup F_{M_2-M_1})$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$, where $\mathcal{M}_1$ contains domain mapping function $F_{M_2-M_1} : M_{2p} \to M_{1k}$, and $M_{2p} \supset M_{1k}$. The domain mapping in this case can be implemented in two steps: first use data compression function F_{comp} to compress the domain, $F_{\text{comp}} : M_{2p} \to M'_{2p} \subseteq M_{1k}$; then implement domain mapping through domain mapping function $F'_{M_2-M_1} : M'_{2p} \to M_{1k}$.

The domain mapping function in Theorem 4.3 can be viewed as a composite function: $F_{M_2-M_1} : M_{2p} \to M_{1k} = F'_{M_2-M_1} \circ (F_{\text{comp}} : M_{2p} \to M'_{2p}) \to M_{1k}$.

Since data compression is employed in the mapping function, some data information obtained by the referenced model will inevitably be lost, which should be fully noted in usage.

Case 2: $M_{2p} \cap M_{1k} = \emptyset$. In this case, the domain data of the two models have no intersection, and one model cannot directly reference another model' s processed data results. Can we indirectly utilize another model' s processing results? This is a question worth exploring. This paper proposes a "data proxy" method to achieve domain mapping under non-intersecting domain conditions.

Definition 4.3 (Domain Data Proxy): Let two models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$ of first-order language $\mathcal{L} = \{C, R, F\}$ have subdomains $M_{1k} \subseteq M_1$, $M_{2p} \subseteq M_2$. Under the condition $M_{2p} \cap M_{1k} = \emptyset$, if for some member $b_i \in M_{2p}$ in M_{2p} we can designate a member $a_i \in M_{1k}$ in M_{1k} to correspond to it, and this designation is meaningful for referencing $\mathcal{M}_2$' s data processing results in $\mathcal{M}_1$, then a_i is called a domain data proxy in domain M_1 for domain M_2' s data member b_i.

Using the domain data proxy method, data referencing between models can be achieved under non-intersecting domain conditions. Of course, as clarified in Definition 4.3, data proxy designation is determined for referenced data and has specific meaning in the proxy' s model for data processing—i.e., it only makes sense when the referencing model needs to utilize the referenced model' s processed data results.

For example, if one model' s domain contains a team composed of Zhang San, Li Si, Wang Wu, etc., we can use numbers 1, 2, 3, etc. in another model' s domain to correspond one-to-one. In the model using numbers, the numbered digits implicitly correspond to team members from the original model. This is just a metaphorical illustration.

Cases where model domains have no intersection yet need to reference each other' s processed data should be relatively rare. In most cases, for models to mutually utilize processed data, there should be data overlap in domains, i.e., the cases listed in Section 4.3.1.

4.4 Impact of Domain Mapping Functions on Model Combination Stability

Domain mapping functions bring much flexibility and convenience for mutual data referencing in model combinations, but after adopting them, is the model combination network stable? This is a question that must be answered after introducing domain mapping functions into model combinations. In reference [9], model combinations have several forms: model chains, model trees, and model networks. We analyze each separately.

4.4.1 Impact on Model Chains Reference [9] introduced the composition of model chains. Let a series of models of language $\mathcal{L} = \{C, R, F\}$ be $\mathcal{M}_\Gamma$:

$\mathcal{M}_\gamma, \mathcal{M}_{\gamma-1}, \ldots, \mathcal{M}_1, \mathcal{M}_0$, where $\mathcal{M}_{i-1}$ is a submodel of $\mathcal{M}_i$, i.e., $\mathcal{M}_{i-1} \subseteq \mathcal{M}_i$ ($\gamma \geq i \geq 1$, $\gamma \in \mathbb{I}$). This series forms a submodel ascending chain:

$$\mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \ldots \subseteq \mathcal{M}_\gamma \quad (4.1)$$

[Figure 4: see original paper] shows the structure of the model chain in equation (4.1). Model $\mathcal{M}_0$ is at the lowest level, $\mathcal{M}_\gamma$ at the highest level, with lower-level models being submodels of higher-level models.

If the model chain forms a data processing whole, from the data transfer direction perspective, lower-level model output data becomes higher-level model input, transferring level by level, and this data transfer is unidirectional.

Now examine the situation of adding domain mapping functions to submodel chains. Let $\mathcal{M}_i, \mathcal{M}_j$ be two models in the model chain, $\mathcal{M}_i \subseteq \mathcal{M}_j$ ($i < j$). We examine adding domain mapping functions from $\mathcal{M}_i$ and $\mathcal{M}_j$ respectively.

Case a) Adding domain mapping function in high-end model $\mathcal{M}_j$ to reference domain data from low-end model $\mathcal{M}_i$

The two models can be expressed as $\mathcal{M}_i = (M_i, C_i, R_i, F_i)$, $\mathcal{M}_j = (M_j, C_j, R_j, F_j \cup F_{M_i - M_j})$, where $F_{M_i - M_j} : M_i' \to M_j'$ and $M_i' \subseteq M_i$, $M_j' \subseteq M_j$. Since lower-end model data effects will definitely transfer to higher-end models in model chains, and because $\mathcal{M}_i \subseteq \mathcal{M}_j$, referencing lower-end model domain data in higher-end models through domain mapping functions only affects higher-end models quantitatively and in advance timing, without affecting model chain stability. Thus we have:

Lemma 4.1: In model chains, referencing low-end model domain data in high-end models through domain mapping functions does not affect model chain stability.

Case b) Adding domain mapping function in low-end model $\mathcal{M}_i$ to reference domain data from high-end model $\mathcal{M}_j$

As shown in [Figure 4: see original paper], adding domain mapping function to low-end model $\mathcal{M}_i = (M_i, C_i, R_i, F_i \cup F_{M_j - M_i})$ to introduce domain data from high-end model $\mathcal{M}_j = (M_j, C_j, R_j, F_j)$: $F_{M_j - M_i} : M_j' \to M_i'$, where $M_i' \subseteq M_i$, $M_j' \subseteq M_j$.

Definition 4.4: Let there be a submodel ascending chain $\mathcal{M}_\Gamma : \mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \ldots \subseteq \mathcal{M}_\gamma$, with $\mathcal{M}_i \in \mathcal{M}_\Gamma$, $\mathcal{M}_j \in \mathcal{M}_\Gamma$, $\mathcal{M}_i \subseteq \mathcal{M}_j$ ($i < j$). If a domain mapping function referencing high-end model $\mathcal{M}_j$' s value range data is added to low-end model $\mathcal{M}_i$, i.e., $\mathcal{M}_i = (M_i, C_i, R_i, F_i \cup F_{M_j - M_i})$, where $F_{M_j - M_i} : M_j' \to M_i'$, $M_i' \subseteq M_i$, $M_j' \subseteq M_j$. In this case, referenced data effects will transfer through the model chain: $\mathcal{M}_i$' s output data will transfer to $\mathcal{M}_{i+1}$, eventually to $\mathcal{M}_j$' s input, affecting $\mathcal{M}_j$' s output, then reverse transfer to $\mathcal{M}_i$' s input, forming a cycle. This data cycle referencing through domain mapping functions in model chains is called circular reference.

From Definition 4.4, adding domain mapping functions to low-end model $\mathcal{M}_i$ to reference domain data from high-end model $\mathcal{M}_j$ involves data circular referencing, which may cause system instability. Particularly, in Definition 4.4, if we let $i = j$, then $\mathcal{M}_i$ will reference its own output data in input, similar to "data feedback" in automatic control systems, potentially causing unstable operation. Thus we have:

Theorem 4.4: In model chains, referencing high-end model domain data in low-end models through domain mapping functions involves data circular referencing and may cause system instability.

From Theorem 4.4, in cases where low-end models reference high-end model domain data through domain mapping functions in model chains, due to potential system instability factors, technical solutions must be verified. For unstable systems, technical solutions should be adjusted to ensure stable operation.

Case c) Impact on model structure after adding domain mapping functions to model chains

After adding domain mapping functions, the domains, relations, and other functions of each model in the model chain remain unchanged. Therefore, adding domain mapping functions does not affect model combination structure. However, data transfer paths are changed, affecting the overall data processing effect of the model combination. In other words, it affects data transfer structure, i.e., data flow structure. [Figure 4: see original paper] shows data transfer relationships after adding domain mapping functions to model chains.

At first glance, [Figure 4: see original paper] shows little difference from the previous figure. However, when domain mapping functions are added in more than one place, we cannot consider only the original model chain but must also consider the data transfer path structure formed after previously added mapping functions. In this case, data transfer graphs should be used to examine the impact of newly added domain mapping functions on model chains. This issue is not prominent in single model chains but cannot be avoided in model trees and model networks with multiple model chains.

4.4.2 Impact on Model Trees [Figure 5: see original paper]a shows a situation where domain mapping functions are added to a submodel tree $\mathcal{M}_\Gamma$. In the figure, model $\mathcal{M}_i$ adds domain mapping function $F_{M_k - M_i} : M_k \to M_i$ (for convenience and clarity, domain representation is directly used here, though generally it should be subdomain mapping), mapping model $\mathcal{M}_k$' s output to $\mathcal{M}_i$' s input; model $\mathcal{M}_q$ adds mapping function $F_{M_j - M_q} : M_j \to M_q$, mapping model $\mathcal{M}_j$' s output to $\mathcal{M}_q$' s input.

[Figure 5: see original paper]a also shows that models $\mathcal{M}_k$ and $\mathcal{M}_i$ are not on the same model chain, while models $\mathcal{M}_j$ and $\mathcal{M}_q$ are on the same model chain. Based on the above analysis, since $\mathcal{M}_k$ and $\mathcal{M}_i$ are not on the same model chain, no model data circular referencing occurs, thus not affecting system

stability; while $\mathcal{M}_j$ and $\mathcal{M}_q$ are on the same model chain, with $\mathcal{M}_q$ as the low-end model, $\mathcal{M}_q \subseteq \mathcal{M}_j$, data circular referencing exists, potentially causing system instability. Thus we have:

Corollary 4.1: In model trees, if two models connected through domain mapping functions for data referencing are in the same submodel chain, and high-end model output is referenced to low-end model input, data circular referencing exists, potentially causing system instability.

Data referencing with potential stability issues must be verified. If verification confirms system instability, technical solutions must be adjusted to ensure stable operation.

As mentioned in the previous section, when domain mapping functions are added in more than one place, newly added mapping functions must consider the impact of already-added mapping functions, verifying whether system stability is affected from a data transfer relationship perspective. As shown in [Figure 5: see original paper]b, based on the model tree with mapping functions already added in [Figure 5: see original paper]a, another domain mapping function $F_{M_{pm}-M_q} : M_{pm} \rightarrow M_q$ is added, forming a new data transfer cycle path: $\mathcal{M}_q \rightarrow \mathcal{M}_{pm} \rightarrow \cdots \rightarrow \mathcal{M}_j$, creating new data circular referencing issues, thus requiring stability verification again. Therefore we have:

Corollary 4.2: In model trees with domain mapping functions already added, if newly added mapping functions combine with original data paths to form new data transfer paths, creating new paths where low-end models reference high-end model domain data, then new data circular referencing exists after adding new domain mapping functions, potentially causing system instability, requiring stability verification.

Corollary 4.2 points out that new stability issues requiring verification can arise in submodel trees with more than one model chain. If new instability occurs, technical solutions must also be adjusted to ensure stable operation.

4.4.3 Impact on Model Networks [Figure 6: see original paper] shows an example of adding domain mapping functions to model network $\mathcal{M}_\Gamma$. In $\mathcal{M}_\Gamma$, two domain mapping functions are added: $F_{M_{pnj}-M_{qml}} : M_{pnj} \rightarrow M_{qml}$ and $F_{M_{p01}-M_{rsj}} : M_{p01} \rightarrow M_{rsj}$. Here, $F_{M_{p01}-M_{rsj}}$ maps model $\mathcal{M}_{p01}$' s output to model $\mathcal{M}_{rsj}$' s input, while $F_{M_{pnj}-M_{qml}}$ maps model $\mathcal{M}_{pnj}$' s output to model $\mathcal{M}_{qml}$' s input.

In the figure, models $\mathcal{M}_{pnj}$ and $\mathcal{M}_{qml}$ are not on the same submodel chain, so no data circular referencing exists between them, thus mapping $F_{M_{pnj}-M_{qml}}$ does not affect system stability. However, models $\mathcal{M}_{p01}$ and $\mathcal{M}_{rsj}$ are on the same submodel chain, with $\mathcal{M}_{p01}$ as the high-end model. Therefore, mapping function $F_{M_{p01}-M_{rsj}}$ involves data circular referencing and may affect system stability.

Based on the above analysis of adding domain mapping functions to model chains and model trees, we obtain:

Corollary 4.3: In model networks, if two models connected through domain mapping functions for data referencing are in the same submodel chain, and high-end model output is referenced to low-end model input, data circular referencing exists, potentially causing system instability.

In model networks, data referencing with potential stability issues must be verified. If verification confirms system instability, technical solutions must be adjusted to ensure stable operation.

As mentioned, model networks also face the issue of newly added domain mapping functions causing new cyclic data paths. [Figure 6: see original paper] shows this situation. In [Figure 6: see original paper], based on [Figure 5: see original paper], a new domain mapping function $F_{M_{q0k}-M_{pnj}} : M_{q0k} \rightarrow M_{pnj}$ is added, forming a new data transfer cycle path: $\mathcal{M}_{pnj} \rightarrow \mathcal{M}_{qml} \rightarrow \mathcal{M}_{p01} \rightarrow \mathcal{M}_{rsj} \rightarrow \mathcal{M}_{q0k} \rightarrow \mathcal{M}_{pnj}$, creating new instability possibilities. Therefore, technical verification of this mapping function' s addition is required. We have:

Corollary 4.4: In model networks with domain mapping functions already added, if newly added mapping functions combine with original data paths to form new data transfer paths, creating new paths where low-end models reference high-end model domain data, then new data circular referencing exists after adding new domain mapping functions, potentially causing system instability, requiring stability verification.

In summary: In model combinations, if low-end models on the same submodel chain reference high-end model data, data circular referencing problems exist; when adding domain mapping functions to model combinations, using data transfer graphs to analyze data circular referencing is more convenient; for cases with data circular referencing, stability verification is required, and if system instability factors exist, technical solutions must be adjusted to ensure stable operation.

5. Application Prospects of Domain Mapping Function Methods

5.1 Applications to Model Combination

We have analyzed methods for adding domain mapping functions to model chains, model trees, and model networks. Results show that: (1) Between any two models in a model combination, as long as they are not on the same submodel chain, domain mapping functions can be added for data referencing without stability issues; (2) Even if two models are on the same submodel chain, as long as the domain mapping is not from high-end model to low-end model, system stability will not be affected by domain mapping function referencing; (3) Only when two models are on the same submodel chain and data is referenced

from high-end model to low-end model may data circular referencing occur, potentially causing system stability issues requiring verification. Therefore, in structurally complex model networks, to determine whether data circular referencing will occur when adding domain mapping functions, using data transfer graph methods is necessary, as shown in [Figure 5: see original paper]b and [Figure 6: see original paper].

Model restricted union operations achieve model combination. Theorem 3.1 reveals that the essence of model restricted union operations is maintaining function and relation definition domains unchanged during union operations to preserve function and relation validity, achieving model combination. The introduction of domain mapping function methods greatly facilitates mutual utilization of processed data between models in combinations, enabling greater performance improvements. We expect domain mapping function methods to be widely applied in model combinations.

5.2 Applications to Intelligent Agent Research

Currently, multimodal intelligent systems [14][15][16] and multi-agent systems [17][18][19] based on large models are important development directions in AI, achieving remarkable results. Both involve data conversion, interaction, and collaboration between multiple models. The submodel combination method proposed in reference [9] reflects logical relationships between different models from a logical structure perspective, providing technical support for multimodal embodied intelligent systems and multi-agent systems. The domain mapping function method proposed in this paper provides feasible solutions for mutual data referencing between models. Therefore, combining submodel combination methods with domain mapping function methods will provide better support for intelligent agent research.

For embodied intelligent agents and multi-agent systems, we can view them as composed of several relatively independent subsystems or modules. From a logical structure perspective, these subsystems or modules can be seen as individual models, thus combining them into larger systems or models. The domain mapping function method proposed in this paper indicates that processed data can be mutually referenced between these component models and specifies logical principles to follow, as well as monitoring and handling methods for data circular referencing. Obviously, domain mapping function methods provide a flexible and efficient implementation scheme for data utilization between models, which plays an important role in improving overall system data processing efficiency. This paper only proposes the principle scheme for this cross-model data referencing, awaiting further application and refinement in actual systems.

5.3 Prospects for Application to AI Foundational Theory

Currently, technical research in AI fields such as multimodal large language models, knowledge graphs, AI and multimodal mathematical reasoning, cross-

domain applications, and embodied intelligent systems has made rapid progress and remarkable achievements [14]-[19]. Simultaneously, several foundational theoretical research efforts need to proceed in parallel. The model restricted union operation method proposed in reference [9] enables studying model combination problems from a mathematical logic perspective; the domain mapping function method proposed in this paper makes this model combination method more efficient and effective in data processing. We expect it to be widely applied in AI model construction.

From a data transfer perspective, the aforementioned domain mapping function method is where a model in a model combination actively references another model' s processed data. In practical applications, we can imagine that data processed by one model can also actively provide support to other relevant models. For example, coordinate detection data obtained by a motion intelligent agent' s position detection module should be actively submitted in real-time to relevant modules for use. In this case, from a domain mapping perspective, the logical effect of data transfer is equivalent to actively referencing other models' data as described previously. Therefore, if systems adopt schemes where data is actively submitted to other models, data circular referencing issues also exist, along with factors affecting system stability. Only in this case, analysis of data circular referencing should be conducted from the perspective of the referencing model, while the model actively submitting data need not consider its effects.

The domain mapping function method proposed in this paper is a domain data transfer control method that can be viewed as a data referencing method. From a practical usage perspective, is functional referencing possible in large model combinations? This is worth in-depth research. From a model perspective, it can be described as follows: Let two models $\mathcal{M}_1 = (M_1, C_1, R_1, F_1)$, $\mathcal{M}_2 = (M_2, C_2, R_2, F_2)$ of first-order language $\mathcal{L} = \{C, R, F\}$. Now suppose $\mathcal{M}_1$ provides domain data $M_{1k} \subseteq M_1$ to $\mathcal{M}_2$ for processing, borrowing $\mathcal{M}_2$' s function set F_2 and relation set R_2 for data transformation and inference, finally returning result data $M_{2p} \subseteq M_2$ to $\mathcal{M}_1$. This is described as a two-step implementation: first add mapping vector $\langle F_{M_{1k}-M_2}, G_{1-2}\rangle$ to $\mathcal{M}_1$, where $F_{M_{1k}-M_2}$ is data submitted by model $\mathcal{M}_1$ to $\mathcal{M}_2$, and G_{1-2} is functional processing requirements submitted by $\mathcal{M}_1$ to $\mathcal{M}_2$, including data transformation and inference requirements, i.e., specifying processing functions and inference relations in $\mathcal{M}_2$. After $\mathcal{M}_2$ completes processing requirements submitted by $\mathcal{M}_1$, mapping function $F_{M_{2p}-M_1}$ returns results to $\mathcal{M}_1$. This is equivalent to enabling external models to call $\mathcal{M}_2$' s internal functions. In complex model combinations, this working mode allows models to mutually utilize processing resources, which is a model inter-data processing method worth exploring. In multi-agent systems, this functional referencing is worth further research.

Mathematical logic provides foundational theoretical support for AI system research. In AI foundational technology research, including modeling theory, knowledge representation and reasoning, machine learning theory, and cross-domain applications, all 离不开 mathematical logic theory support. For AI sys-

tem research, model restricted union operations and domain mapping function methods provide a feasible model combination method from a mathematical logic model theory perspective, which we expect to be widely applied as a foundational technology for AI system model construction.

6. Conclusion

Based on model restricted union operations from reference [9], this paper, through in-depth analysis of model domains, reveals that the essence of model restricted union operations is maintaining function and relation definition domains unchanged during model union operations to ensure union operation validity. Through further analysis, this paper proposes the domain mapping function method. This method is a feasible solution for data referencing between models in model combinations, improving model data processing efficiency. From the perspective of the entire model combination, it can more fully utilize system resources and improve overall model combination performance, making it a system optimization method worth adopting in model construction.

In AI research, foundational research is indispensable. Model restricted union operations and domain mapping function methods, as mathematical logic research topics, can be regarded as a new research direction in mathematical logic model theory, providing important support for AI model construction, multimodal embodied intelligence applications, cross-domain applications, etc.

References

[1] Vaswani A, Shazeer N, Parmar N, Uszkoreit U, Jones L, Gomez A N, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, USA: Curran Associates Inc., 2017. 6000−6010.

[2] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2018. 4171−4186.

[3] Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Florencia B, et al. GPT-4 technical report. arXiv preprint arXiv: 2303.08774, 2023.

[4] 童云海, 陈建龙. DeepSeek 热潮下的双重变革: 大模型的技术革新与高校图书馆服务范式的重构. 大学图书馆学报, 2025 年第 1 期.

[5] 孟小峰等. 科学发现中的机器学习方法研究. 计算机学报, 2023, 第 5 期, 877-895.

[6] Hutson M. Artificial intelligence faces reproducibility crisis. Science, 2018, 359(6377): 725-726.

[7] Khan S, Naseer M, Hayat M, Zamir S W, Khan F S, Shah M. Transformers in vision: A survey. arXiv preprint arXiv: 2101.01169, 2021.

[8] 赵希顺. 简明数理逻辑. 科学出版社, 2021.11.

[9] 刘志模, 刘徽, 王晓山, 陈振. 一种树形模型链构建方法及其应用展望. ChinaXiv:202503.00280.

[10] 罗里波. 模型论及其在计算机科学中的应用. 北京师范大学出版社, 2012.1.

[11] Michael Huth, Mark Ryan. Logic in Computer Science: Modelling and Reasoning about Systems, Second Edition. ISBN 978-7-111-21397-0.

[12] 孙希文. 数理逻辑. 高等教育出版社, 2019.11.

[13] 雷英杰等. 模糊逻辑与智能系统. 西安电子科技大学出版社, 2016.5.

[14] Wang J Q, Wu Z H, Li Y W, Jiang H Q, Shu P, Shi E Z, et al. Large language models for robotics: Opportunities, challenges, and perspectives. arXiv preprint arXiv: 2401.04334, 2024.

[15] Yang Z Y, Li L J, Lin K, Wang J F, Lin C C, Liu Z C, et al. The dawn of LMMs: Preliminary explorations with GPT4V(ision). arXiv preprint arXiv: 2309.17421, 2023.

[16] 王文晟等. 基于大模型的具身智能系统综述. 自动化学报, 2025 年第 1 期.

[17] Zhao W S, Queralta J P, Westerlund T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: Proceedings of 2020 IEEE Symposium Series on Computational Intelligence. IEEE, 2020: 737-744.

[18] Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in Neural Information Processing Systems, 2017, 30: 6382-6393.

[19] Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning. In: Proceedings of the 36th International Conference on Machine Learning. 2019: 5261-5274.

(Corresponding author: Liu Zhimo, E-mail: hylj2010@sohu.com)

Author Contributions:

Liu Zhimo: Proposed research ideas, drafted paper, finalized manuscript.
Chen Zhen: Revised paper, compared with entity models, analyzed application prospects and proposed revision suggestions.
Wang Xiaoshan: Revised paper, proposed revision suggestions on application prospects.
Liu Hui: Revised paper, revised theoretical discussions, proposed revision suggestions on application prospects.

Note: Figure translations are in progress. See original paper for figures.