Interoperability Semantics in Distributed Frame Hierarchy

Dmitri Soshnikov
Department of Numerical Mathematics and Programming,
Moscow Aviation Institute (State Technical University)
Moscow, Russia
dmitri@soshnikov.com

Abstract

The paper presents a formal semantics for distributed inference in the family of intelligent systems based on distributed frame hierarchy model. A formal semantics for a subclass of frame-production systems is presented, based on denoting a knowledgebase by a frame world state function $\mathcal{W}$, and then defining an operational semantics for logical inference in terms of transformations of this function. Distributed frame hierarchy is then represented by a set of state functions with either static or mobile references to each other, corresponding to static and mobile inheritance or aggregation. Two types of semantics are introduced for inference in distributed system: equivalent semantics, where distributed system is reduced to an equivalent state function by means of combination operator $\bigstar$, and extended semantics, defined on sets of state functions. By demonstrating equivalence of those semantics in the class of normal systems we can also show the equivalence of static and mobile inheritance.

Described semantics forms the basic functionality of JULIA toolkit for creating distributed intelligent systems. Some aspects of implementation are outlined in [1, 2].

Keywords: Distributed Frame Hierarchy, Semantics of Distributed Inference, Production-Frame Knowledge Representation, Remote Inheritance

Proceedings of the 4th International Workshop on Computer Science and Information Technologies CSIT'2002
Patras, Greece, 2002

1 Introduction

Nowadays, with the growing role of global computer networks and accumilation of vast amounts of information, there is a lot of research going on in the field of distributed intelligent systems. Introduction of certain level of intelligence into the operations of the World Wide Web, supported by the recent initiatives of the W3C, such as Semantic Web, will create much richer environment for distributed storage of not only information, but *knowledge*. Distributed knowledge sharing and reuse is also important in less "ambitious" smaller-scale areas, such as intelligent control of spatially distributed industrial processes, knowledge-based virtual corporations, etc.

In [1,2] we proposed a model of **distributed frame hierarchy**, which is effective for implementing systems that require distributed knowledge sharing and reuse. This model follows deliberative multi-agent architecture and serves as a direct physical implementation of extensible ontology model [4, pp.284–302], thus exhibiting natural taxonomy-based paradigm of distribution, making classical object-oriented [3] and semiotic [4] approaches to knowledge engineering applicable to creating distributed knowledgebases.

In this paper we present the formal semantics for logical inference in the distributed frame hierarchy. In defining this semantics there are two major milestones: formulating inference semantics for single frame-based system (so-called **frame world**) that can later be included as a part of the distributed hierarchy, and extending this semantics to the case of multiple interoperating frame worlds.

Definition of inference semantics for frame-based system differs from that of traditional programming languages [5], because the order of execution of different production rules does not depend on the order and stucture of statements in the knowledge representation language, but rather on the current *state* of the static portion of the knowlegdebase. Thus we propose two-step definiton of semantics for knowledge representation language: the input knowledgebase is denoted by a **frame-world state function** $\mathcal{W}$, that contains both structural (static) knowledge of the problem domain and dynamic production rules in defined expression syntax, and **evaluation function** $\|\cdot\|$ is defined on expressions containing slot references, that implies a series of transformations defined on the set of state functions $\mathcal{W}$.

To formulate the semantics of distributed inference we first define a **distributed system** as a set of individual frame worlds, and a set of operations ($\bigstar$, $\blacktriangleleft$) for constucting distributed systems from individual frame worlds. Each distributed frame system can be characterized by a single state function $\mathcal{W}$ of an equivalent system, that is a **combination** of state functions of individual frame worlds, on which inference semantics is already defined. However, to study interoperability of different frame worlds in a distributed hierarchy we need to define the semantic evaluation function on sets of state functions, and specifically describe treatment of **remote references**, i.e. references to frames and slots of remote frame worlds. By introducing static and mobile references we can define semantics of static and mobile inheritance and aggregation, and also demonstrate the equivalence of complex semantics to that of an equivalent system. From this follows the equivalence of static and mobile inference in distributed frame hierarchy.

Before going into the details of inference semantics, we will briefly introduce the main ideas behind the model of distributed frame hierarchy.

2 Distributed Frame Hierarchy Model

In distributed artificial intelligence the multi-agent paradigm is dominating, but it still leaves a lot of flexibility when the actual system architecture is concerned. There are several well-studied models of agent behaviours and interactions, as well as certain classifications of agents according to their properties and interoperability [6].

We can also look at the problem of knowledge sharing from the point of view of traditional distributed systems and component models. In studying and constructing complex systems an object-oriented approach [3] to decomposition turns out to be very productive, which in the field of knowledge engineering is adopted either as frame knowledge representation, and more recently as a model of **ontological systems** [4]. Ontologies and ontological systems as such have been mostly used in formulating an external knowledge of the problem domain for multi-agent systems, which can then be either translated to internal knowledge representation of deliberative agents in the society [7], or implicitly accounded for in algorithms for reactive agents.

A model of **distributed frame hierarchy** proposes to extend frame knowledge representation by allowing frame hierarchy to span several network nodes. Each node would then contain a frame sub-hierarchy called a **frame world**, and different frame worlds would interoperate to collectively solve a given problem. Such a model can be classified as deliberative collaborative multi-agent model, but it also possesses an important property of being **auto-ontological**, i.e. each frame sub-hierarchy defines a natural taxonomic ontology, which serves as both internal knowledge representation used in inference, and external explicit conceptualization of the problem domain.

There are two major types of relations between frame worlds in a distributed hierarchy: **inheritance** and **aggregation**. Those relations are achieved by using **remote refereneces** to frames or slots of a sub-hierarchy located on different network location. There are two types of references corresponding to two different types of interoperability: **static** and **mobile** references.

With static reference, when the underlying slot is to be computed a remote call is made to the network server for the target frame world, which takes over the inference until the value is obtained and retured to the original caller (so-called **invocation**). For static remote inheritance the situation is more complex, because often rules located on the remote host have to be applied to the data located in the local sub-hierarchy. In this case a series of remote call-backs is made, handled by two proxy frames on both sides of the remote inheritance link. Diring this process, dynamic rules remain on their original location, and slot values forming the static knowledge are transferred over the network, thus resulting in **remote knowledge application**.

Mobile references provide the opposite behaviour called **inclusion**: when such a reference is computed, the whole remote hierarchy is transferred through the network in some internal represenation and instantiated locally, thus reducing remote reference to a local one. In this case dynamic knowledge is also transferred through the network. Implementations of remote mobile inheritance presents no complications because both frame worlds after inclusion are located on the same host.

Presented outline of the distributed frame hierarchy model is by no means complete, but should provide the basis for the following sections. More detailed description can be found in [1, 2], where implementation issues of proposed approach in JULIA (Java Universal Library for Intelligent Applications) toolkit are also discussed, and also several areas of applications are outlined together with actual examples of distributed intelligent systems that have been implemented.

3 Local Inference Semantics

Semantics of distributed inference in the model of distributed frame hierarchy is based on inference semantics of systems with production-frame knowledge representation. In this section we present operational semantics for a family of production-frame systems with backward, forward and combined inference. JULIA toolkit, described in [1, 2], follows this semantics.

Unlike traditional programming languages, where in some cases denotates can be assigned directly to language constructs [5, 8], defining semantics for logical inference can be done in terms of transformations of knowledgebase *state*, denoted by a **state function** $\mathcal{W} \in \mathbb{W}$. Initial constructions of the knowledge representation language are mapped to the *initial state* $\mathcal{W}_0$ by means of **representation language semantics** defined by a funtion $\mathcal{L} : A^* \to \mathbb{W}$, where A^* — set of finite character strings in some alphabet A representing a knowledgebase text. **Inference semantics** is defined as a function of the form $\mathcal{E} : \mathbb{E} \times \mathbb{W} \to \mathbb{T} \times \mathbb{W}$, that computes a value of an expression $E \in \mathbb{E}$ giving a value $x \in \mathbb{T}$, leading to a series of changes in the knowledgebase state in the process of inference.

3.1 Representation of knowledgebase state

Knowledgebase state is denoted by one function $\mathcal{W}$ that contains both **static knowledge** in the form of slot values, and **dynamic rules** in some abstract syntax. Frames are denoted by identifiers from some set I, each having slots denoted by identifiers from I_F. We can identify a slot by complete id from a set $\mathbb{I} = \{\langle f, s\rangle | f \in I, s \in I_f\}$.

Each slot would have a complex structure $S = \langle v, u, \{Q_i\}, \{D_j\}, \{C_k\}, \sqsubseteq_q, \sqsubseteq_d, \alpha\rangle \in \mathcal{S}$ containing current value $v \in \mathbb{T}$, default value $u \in \mathbb{T}$, set of backward chaining queries $\{Q_i\}$, set of forward chaining daemons $\{D_j\}$, set of constraints $\{C_k\}$, rule selection strategies in the form of complete order relations on $\{Q_i\}, \{D_j\}$, and activity flag α. By $\mathbb{T}$ we would denote some set of values, possibly of different types, the only requirement for which is to form a complete lattice. [8] contains many examples of how such a set can be constructed, for example as a disjunctive sum of primitive types. It would in most cases include such types as integers, lists, references, etc.

State function $\mathcal{W} : \mathbb{I} \to \mathcal{S}$ would then map slot ids to the actual slot values. For simplicity, we would use constructions as $\mathcal{W}(f, s).value$, and even $\mathcal{W}.f.s.value$ to access individual components of a slot $\langle f, s\rangle$. We would also define the operator $[\cdot \leftarrow \cdot] : \mathbb{I}\times\mathbb{T} \to \mathbb{W} \to \mathbb{W}$ that changes a value of a given slot giving a new state function:

$$\mathcal{W}[f.s \leftarrow x] = \lambda\langle f_1, s_1\rangle.\langle f_1, s_1\rangle \neq \langle f, s\rangle \to \mathcal{W}(f_1, s_1), \langle x, \mathcal{W}(f, s).def, \mathcal{W}(f, s).queries, \ldots, \mathcal{W}(f, s).alpha\rangle$$

3.2 Expression syntax

In order to formulate rules and define inference semantics we would need to use some sort of **expressions** containing constants, references to slots, arithmetical and logical operations and some other features contained in the knowledge representation language. Such expressions may or may not be syntactically equivalent to the original knowledge representation language. Without describing syntax in detail we will adopt standard expression syntax, using $f.s$ to denote slot reference, and $b \to u, v$ as conditional operator that computes u if b is equivalent to $true$, and v otherwise.

3.3 Expression evaluation

Inference semantics is defined in terms of evaluation of arbitrary expressions $E \in \mathbb{E}$ that contain slot references. We would use the notation $\|E\|^C_{\mathcal{W}_1\to\mathcal{W}_2} = v \Leftrightarrow \langle v, \mathcal{W}_2\rangle = \mathcal{E}(E, \mathcal{W}_1, C)$, where C is so-called **evaluation context**. Note, that in this notation $\mathcal{W}_2$ is actually output value of a function, even though it is used on the left-hand side of the equality.

For all standard arithmetical and logical operations we would define short-circuit evaluation in the natural way, that is

$$\|E_1 \circledast E_2\|_{\mathcal{W}_1\to\mathcal{W}_2} = \begin{cases} \bot, \|E_1\|_{\mathcal{W}_1\to\mathcal{W}_2} = \bot \\ \|E_1\|_{\mathcal{W}_1\to\mathcal{W}'} \circledast \|E_2\|_{\mathcal{W}'\to\mathcal{W}_2} \end{cases}$$

where $\circledast \in \{+, -, *, /, \mathbf{and}, \mathbf{or}\}$. Constants are directly returned regardless of the state $\mathcal{W}$: $\forall x \in \mathbb{T}\ \|x\|_{\mathcal{W}\to\mathcal{W}} = x$.

3.4 Evaluation of slot references

The essence of inference semantics is defined by evaluation of slot references. $\|f.s\|$ should return the value of the slot, or intiate the process of logical inference, applying corresponding rules.

In order to describe application of rules we would define a funtion $\mu^{\sqsubseteq}_{\mathcal{W}\to\mathcal{W}'}(Q)$ that would compute a set of expressions Q ordered by a complete order $\sqsubseteq$ until the first non-null value is obtained:

$$\mu^{\sqsubseteq}_{\mathcal{W}\to\mathcal{W}'}(Q) = \begin{cases} \|\inf Q\|_{\mathcal{W}\to\mathcal{W}'}, & \|\inf Q\|_{\mathcal{W}\to\mathcal{W}'} \neq \bot \\ \mu_{\mathcal{W}''\to\mathcal{W}'}(\hat{Q}), & \|\inf Q\|_{\mathcal{W}\to\mathcal{W}''} = \bot \\ \bot, & Q = \varnothing\ (\mathcal{W}' = \mathcal{W}) \end{cases}$$

Here $\hat{Q} = Q \setminus \{\inf Q\}$, where ordering is defined with respect to $\sqsubseteq$. Correctness of this recurrent definition can be formally demonstrated by traditional methods of lattice theory.

Using this function, we can define semantics of slot evaluation in the following manner:

$$\|f.s\|_{\mathcal{W}\to\mathcal{W}'} =$$
$$\begin{cases} \mathcal{W}(f,s).value, \mathcal{W}(f,s).value \neq \bot \ (\mathcal{W}' = \mathcal{W}) \\ \bot, \mathcal{W}(f.s).busy = true \\ x = \mu^{\mathcal{W}(f,s).\sqsubseteq_q}_{\mathcal{W}[f.s.busy \leftarrow true]\to\mathcal{W}''}(\mathcal{W}(f,s).rules), \\ \qquad \mathcal{W}' = \Phi_{f.s}(\mathcal{W}''[f.s \leftarrow x, f.s.busy \leftarrow false]) \\ \qquad (x \neq \bot) \\ y = \mu_{\mathcal{W}'''\to\mathcal{W}''''}(\{p.s | p \in \|f.parent\|_{\mathcal{W}''\to\mathcal{W}'''}\}), \\ \qquad \mathcal{W}' = \Phi_{f.s}(\mathcal{W}''''[f.s \leftarrow y, f.s.busy \leftarrow false]) \\ \qquad (y \neq \bot) \\ \bot, \text{ otherwise} \end{cases}$$

If the slot value is contained in the current state, it is returned. Otherwise, we are trying to compute all query expressions in order to obtain the value, and then, if needed, evaluate slots with the same name of all parent frames, computed as $\|f.parent\|$. To avoid infinite loops, when the slot is in the process of evaluation its *busy* flas is set, and $\bot$ is returned upon another recurrent evaluation.

After some value is obtained, a **directed forward inference function** $\Phi : \mathbb{I} \to \mathbb{W} \to \mathbb{W}$ is applied, to see what can be derived by forward inference as a result of new slot value having been added to the static portion of the knowledgebase. This function is defined as a fixpoint of **one-step directed inference function** $\varphi : \mathcal{A} \times \mathbb{W} \to \mathcal{A} \times \mathbb{W}$, where $\mathcal{A}$ is a set of **activation functions** $\alpha : \mathbb{I} \to \{free, todo, done\}$ showing for each slot its current status with respect to forward inference. φ is monotonic with respect to natural ordering on $\mathcal{A}$ induced by complete order $free \sqsubseteq todo \sqsubseteq done$, and to ordering on $\mathbb{W}$ (which assumes that dynamic portion of the knowledgebase remains constant):

$$\begin{aligned} \mathcal{W}_1 \sqsubseteq \mathcal{W}_2 \quad &\Leftrightarrow \ \forall \langle f,s \rangle \in \mathbb{I} \\ &\mathcal{W}_1(f,s).value \sqsubseteq \mathcal{W}_2(f,s).value \end{aligned} \tag{1}$$

Thus, by Knaster-Tarski theorem we can conclude the existence of a family of fixpoints $\tilde{\Phi}_\alpha(\mathcal{W}) = \bigsqcup_i \varphi^i(\alpha, \mathcal{W})$, that define $\Phi_{f.s}(\mathcal{W}) = \tilde{\Phi}_{\alpha_0[f.s \leftarrow todo]}(\mathcal{W})$.

3.5 Dynamic inheritance

Inheritance relation : in the presented semantics is dynamically defined by the *parent* slot: $f : g \Leftrightarrow g \in \|f.parent\|$. This significantly enriches the semantics, allowing dynamic frame specification by using rules or specific search functions, but also has to be accounted for in the definition of evaluation function: whenever a parent has to be computed, a state change has to be taken into account.

To model the inheritance we use **context** of evaluation function, and substitute references to the container frame in all rules by specific token *this*, with specific evaluation rule

$$\|this.s\|^f_{\mathcal{W}\to\mathcal{W}'} = \|f.s\|^f_{\mathcal{W}\to\mathcal{W}'}$$

In this manner, when rules from the parent frame are applied, child frame is passed through evaluation context, and thus all references in rules are made to slots of a child frame.

3.6 Concluding remarks

Presented semantics is based on partial order defined on $\mathbb{W}$ by (1), and thus does not allow rulebase modification in the process of inference, as well as metarules for dynamically changing rule selection strategies and other inference properties, because such modifications would lead to incomparable state values and break the monotonicity of inference operator. It also has to be noted that considered inference is strictly monotonic, which leads to monotonic inference operators, and allows to apply Knaster-Tarski fixpoint theorem to define the result of forward inference, as well as to demonstrate existence of evaluation for backward inference.

4 Semantics of Distributed Inference

There are different approaches to study interoperability in distributed problem solving. One of the important aspects that can be investigated is **knowledge-theoretic** aspect dealing with notions of **distributed** and **common knowledge** in distributed system [9]. Another possibility is to approach the problem from the point of view of state space search [10] and partitioning it between individual sub-spaces.

In the proposed distributed frame hierarchy model problem state is formed by slot values, and thus is distributed. Arcs of state space search graph are also distributed, because different rules are located on different network nodes. However, search process in this graph takes place synchronously, with either inference process being transferred from one frame world to another, or required part of the graph being transferred.

In this paper, instead of studying search process in the state graph, we would adopt and approach similar to LogicWeb [11], where semantics is defined in terms of interoperability of logic programs using specific combination operators. We would introduce such combination operator for frame hierarchy, and then extend semantic function $\mathcal{E}$ to be defined on sets of state functions describing the whole distributed system.

4.1 Definition of Distributed System

By **distributed frame system** we would mean a set $\{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$ of state functions of individual frame worlds, extended by constructions called **remote references**. Remote reference can be either **static** (denoted by $\blacklozenge\mathcal{W}_i(f,s)$) or **mobile** ($\lozenge\mathcal{W}_i(f,s)$), and correspond to different modes of evaluation (invocation or inclusion respectively). Also, we would without loss of generality assume that sets of frame ids in all frame worlds do not overlap, i.e. $\forall i \neq j \; I_i \cap I_j = \varnothing$.

4.2 Combination of Frame World Functions and Equivalent System

For two frame world functions $\mathcal{W}_1$ and $\mathcal{W}_2$ we would define their **combination** as

$$\mathcal{W}_1 \bigstar \mathcal{W}_2 = \lambda f, s. \begin{cases} \mathcal{W}_1(f,s), & \text{if } \langle f,s\rangle \in \mathbb{I}_1 \\ \mathcal{W}_2(f,s), & \text{if } \langle f,s\rangle \in \mathbb{I}_2 \\ \bot, & \text{otherwise} \end{cases}$$

This operation is symmetric and associative, which allows us to extend it for arbitrary number of arguments. In particular, for distributed frame system $\tilde{\mathcal{W}} = \{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$ we would define $\bigstar\tilde{\mathcal{W}} = \bigstar_{i=1}^{n} \mathcal{W}_i$, and call it a **state function of a distributed system**. This state function, with remote reference operations removed, would correspond to some non-distributed system called an **equivalent system**.

In addition to combination, we can also define **static** and **mobile hierarchy inheritance** operations:

$$\begin{aligned} \mathcal{W}_1 \underset{f}{\blacktriangleleft} \mathcal{W}_2 &= \{\mathcal{W}_1, \mathcal{W}_2[\mathcal{W}_2.\mathit{object.parent} \leftarrow \blacklozenge\mathcal{W}_1.f]\} \\ \mathcal{W}_1 \underset{f}{\triangleleft} \mathcal{W}_2 &= \{\mathcal{W}_1, \mathcal{W}_2[\mathcal{W}_2.\mathit{object.parent} \leftarrow \lozenge\mathcal{W}_1.f]\} \end{aligned}$$

Those operations produce distributed frame systems from sub-hierarchies, but they can also be used in the context of state function, meaning the state function of a resulting distributed system.

4.3 Equivalent and Extended Semantics

Equivalent system for distributed hierarchy is essentially a non-distributed frame hierarchy described by ordinary state function, on which inference semantics has already been defined. We would call this semantics an **equivalent semantics** of distributed system. It defines how different slots are computed, but does not take into account interoperability between frame worlds in different modes of computation of remote references.

To take this interoperability into account, we would define an **extended semantic function** $\tilde{\mathcal{E}} : \mathbb{E} \times \tilde{\mathbb{W}} \times \mathbb{C} \to \mathbb{T} \times \tilde{\mathbb{W}}$, also denoted by $v = \llbracket E \rrbracket^{C}_{\tilde{\mathcal{W}} \to \tilde{\mathcal{W}}'} \Leftrightarrow \tilde{\mathcal{E}}(E, \tilde{\mathcal{W}}, C) = (v, \tilde{\mathcal{W}}')$. We would also extend the definition of distributed system by adding a notion of **current frame world**. We would denote it by $\{\mathcal{W}_1, \ldots, \overline{\mathcal{W}}_i, \ldots, \mathcal{W}_n\} = \langle\{\mathcal{W}_1, \ldots, \mathcal{W}_n\}, i\rangle$, and set of such systems as $\tilde{\mathbb{W}}$.

Evaluation function for expressions without remote references would be reduced to local inference semantics in the natural manner:

$$\llbracket E \rrbracket_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\} \to \{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}'_i,\ldots,\mathcal{W}_n\}} = \|E\|_{\mathcal{W}_i \to \mathcal{W}'_i}$$

Distributed inference would actually be described by evaluation of remote references.

4.3.1 Mobile Interoperability Semantics

To evaluate a mobile reference the remote frame world is combined with the current one by means of $\bigstar$ operator, thus removing all mobile references between them and reducing computation of remote reference to the local one:

$$\begin{aligned} &\llbracket \lozenge\mathcal{W}_j(f,s) \rrbracket_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} = \\ &\quad = \llbracket f.s \rrbracket_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}_i \bigstar \mathcal{W}_j},\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} \end{aligned} \tag{2}$$

If only mobile references are used throughout the whole hierarchy, then current frame world is never changed, but it is extended with other frame worlds as needed. This semantics is very similar to approach presented in [11], where logic programs are loaded and combined into one contextual rulebase to be used by one inference engine.

Remote inheritance also involves passing the context in the form of reference to the base frame to remote hierarchy. However, in mobile inheritance this reference does not become remote, since it refers to the original world that has been combined, but still remains as a part of the current one.

4.3.2 Static Interoperability Semantics

When evaluating static reference, the process of inference is transferred to another frame world that becomes active, and further inference uses another state functions and associated rules:

$$\begin{aligned} &\llbracket \blacklozenge\mathcal{W}_j(f,s) \rrbracket_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} = \\ &\quad = \llbracket f.s \rrbracket_{\{\mathcal{W}_1,\ldots,\mathcal{W}_i,\ldots,\overline{\mathcal{W}}_j,\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} \end{aligned}$$

Static inheritance is slightly more complicated, because base frame reference that is passed through the evaluation context becomes static remote reference, which leads to another remote reference appearing in the evaluation context on the right hand side:

$$\begin{aligned} &\llbracket \blacklozenge\mathcal{W}_j(f,s) \rrbracket^{F}_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} = \\ &\quad = \llbracket f.s \rrbracket^{\blacklozenge\mathcal{W}_i(F)}_{\{\mathcal{W}_1,\ldots,\mathcal{W}_i,\ldots,\overline{\mathcal{W}}_j,\ldots,\mathcal{W}_n\} \to \tilde{\mathcal{W}}'} \end{aligned} \tag{3}$$

In the inference process driven by $\mathcal{W}_j$ state function, whenever *this* has to be computed, another remote call would be made to compute this remote reference, that would refer to the original frame world $\mathcal{W}_i$. In this process $\mathcal{W}_i$ would again become active, and remote reference would be removed according to the following rule:

$$[\![\blacklozenge\mathcal{W}_i(F)]\!]_{\{\mathcal{W}_1,\dots,\overline{\mathcal{W}}_i,\dots,\mathcal{W}_n\}\to\{\mathcal{W}_1,\dots,\overline{\mathcal{W}}_i,\dots,\mathcal{W}_n\}} = \\ = \|F\|_{\mathcal{W}_i\to\mathcal{W}_i} = \mathcal{W}_i(F)$$

4.4 Equivalence of Mobile and Static Interoperability

Given a distributed system, we can compute any slot using equivalent semantics, as though the system was non-distributed, or using extended semantics that takes interoperability into account. It would be natural to expect the results to be identical, however, it is true only in some class of distrubuted systems that we would call **normal systems**. The following statement express this idea more accurately.

Statement 1 (on static and mobile reference evaluation). *For the normal system* $\tilde{\mathcal{W}}$

$$\begin{aligned} [\![\lozenge\mathcal{W}_j(f,s)]\!]_{\{\mathcal{W}_1,\dots,\overline{\mathcal{W}}_i,\dots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'} &= \|f.s\|_{\mathcal{W}\to\mathcal{W}'} \\ [\![\blacklozenge\mathcal{W}_j(f,s)]\!]_{\{\mathcal{W}_1,\dots,\overline{\mathcal{W}}_i,\dots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'} &= \|f.s\|_{\mathcal{W}\to\mathcal{W}'} \end{aligned}$$

where $\mathcal{W} = \mathcal{W}_1 \bigstar \dots \bigstar \mathcal{W}_n$ *— equivalent state function for the distributed system. Also, upon evaluation* $\mathcal{W}' = \bigstar\tilde{\mathcal{W}}'$*.*

This statement can be generalised to an arbitrary expression $E \in \mathbb{E}$, giving $[\![E]\!]_{\tilde{\mathcal{W}}\to\tilde{\mathcal{W}}'} = \|E\|_{\mathcal{W}\to\mathcal{W}'}$ with $\mathcal{W} = \bigstar\tilde{\mathcal{W}}$, and upon evaluation $\mathcal{W}' = \bigstar\tilde{\mathcal{W}}'$.

Most of the systems used in practical applications are normal. They include, in particular, systems based only on one type of interoperability, and fully polymorphic systems (in which no direct references to remote slots are made, and the only type of remote referencing is inheritance).

Since semantics of both mobile and static reference evaluations correspond to equivalence semantics, we can also conclude that mobile and static interoperability in distributed frame hierarchy are equivalent in terms of results obtained upon evaluation:

Corollary 2.

$$[\![\lozenge\mathcal{W}_i(f,s)]\!]_{\tilde{\mathcal{W}}_1\to\tilde{\mathcal{W}}_1'} = [\![\blacklozenge\mathcal{W}_i(f,s)]\!]_{\tilde{\mathcal{W}}_2\to\tilde{\mathcal{W}}_2'}$$

where $\bigstar\tilde{\mathcal{W}}_1 = \bigstar\tilde{\mathcal{W}}_2$*, and upon evaluation* $\bigstar\tilde{\mathcal{W}}_1' = \bigstar\tilde{\mathcal{W}}_2'$*.*

4.5 Taking Network Failures into Account

Network interoperability in real systems is not failure-proof, which in some cases prevents remote calls or data transfers from functioning properly. If we want to account for such problems in the formal semantics, we can, as it is done in [11], use **oracle functions** in equations (2) and (3). It has to be noted that in this case the result of any evaluation that involves remote references becomes non-deterministic, thus making it difficult to formulate precise results about defined semantics.

5 Conclusion

In this paper we have described semantics of logical inference in production-frame systems, and then extended it to define the semantics of distributed inference in the model of distributed frame hierarchy. Extended semantics takes into account two modes of interoperability in distributed frame system (invocation and inclusion), and happens to be equivalent to natural semantics, where distributed hirarchy is viewed as a non-distributed one. Defined semantics allows us to state the equivalence of invocation and inclusion-based approaches to distributed inference in normal systems, and defines the behaviour of software implementations [1, 2].

References

[1] Soshnikov D. Software Toolkit for Buiding Embedded and Distributed Knowledge-Based Systems. In *Proceedings of the 2nd International Workshop on Computer Science and Information Technologies, Vol.1*, USATU Publishing, Ufa, 2000. pp. 103–111.

[2] Soshnikov D. JULIA Toolkit for Creating Distributed Intelligent Systems based on Production-Frame Knowledge Representation. *Electronic Journal "Trudi Mai"*, N 7, 2002. (In Russian) `http://www.mai.ru/mai_works`

[3] Booch G. Object-Oriented Analysis and Design with Applications. Addison-Wesley Publishing Company, 1994.

[4] Gavrilova T.A., Khoroshevsky V.F. Knowledgebases and Intelligent Systems. Piter Publishing House, 2000. (In Russian)

[5] Donahue J.E. Complementary Definitions of Programming Language Semantics, Lecture Notes in Computer Science, 42, 1976.

[6] Hyacinth S. Nwana, Software Agents: An Overview, *Knowledge Engineering Review*, Vol. 11, No.3, 1996. pp. 1–40.

[7] Gruber T. R., A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2), 1993. – pp. 199-220.

[8] Wolfengagen V.E. Constructions of Programming Languages. Methods of Description. Moscow, JurInfoR Publishing, 2001.

[9] Wooldridge M. A Knowledge-Theoretic Approach to Distributed Problem Solving. Proceedings of the 13th European Conference on Artificial Intelligence. John Wiley, August 1998.

[10] Lesser, V.R. An Overview of DAI: Viewing Distributed AI as Distributed Search, *Journal of Japanese Society for Artificial Intelligence*, Vol. 5, No. 4, 1990.

[11] Seng Wai Loke, Adding Logic Programming Behaviour to the World Wide Web, PhD Thesis, Department of Computer Science, The University of Melbourne, Australia, 1998.