An Architecture of Distributed Frame Hierarchy for Knowledge Sharing and Reuse in Computer Networks

Dmitri Soshnikov
Department of Numerical Mathematics and Programming,
Moscow Aviation Technical University
dmitri@soshnikov.com

Abstract

An architecture for building distributed intelligent systems is considered, based on production-frame knowledge representation with distributed frame hierarchy, in which knowledge fragments located on different network nodes can be used cooperatively in the process of distributed or local inference. Frame knowledge representation can also serve as a common denominator when integrating knowledgebase with imperative components or relational data, because objects and component interfaces as well as relational tables can be transparently represented by frames or frame classes.

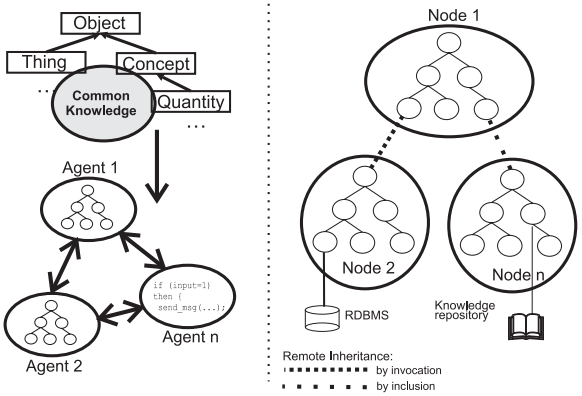

Figure 1. Agent architecture and distributed frame hierarchy

1. Introduction

Distributed intelligent systems can be used in many application areas. In many cases those areas can be classified as distributed knowledge sharing and reuse, when distributed knowledge is applied in some manner to the solution of a problem. Such tasks occur when creating knowledge repositories for distributed knowledge-intensive virtual corporations, collections of knowledgebases maintained by different specialists, when building large-scale expert systems, etc.

When using traditional multiagent approach [1] (see Fig. 1, left hand side) to distributed artificial intelligence, which is very broad and in many cases does not provide existing solution, a developer is faced with creating an application-specific architecture, models of agent communication and software tools for actual implementation. Moreover, complex asynchronous communication in agent society in some cases turns out to be too general, which complicates implementation. While in deliberative agent systems internal knowledge representation of an agent is separated from the external one that is used for communication and should be the same for the agent society, in some cases it makes sense to base knowledge sharing principles on unified knowledge representation and some remote communication mechanisms that evolve from classical notion of remote procedure call.

An approach presented in this paper is based on using production-frame knowledge representation with frame hierarchy distributed over computer network. Such system can be considered as one frame hierarchy that is distributed according to some remote communication protocol (see Fig. 1, right hand side), or as a collection of interoperating subhierarchies — in this case it can be classified as static deliberative collaborative multiagent architecture [1] [1]. Such system can also use mobile interoperability based on **inclusion**, as opposed to static remote-call interoperability based on **invocation**.

While for multiagent systems the structure of the problem domain is typically formulated using ontologies and then translated into internal representation, presented ap-

[1] Despite the fact that formally such architecture can be classified as multiagent, it can also be opposed to multiagent architecture due to the lower level of autonomy of individual subhierarchies.

proach is **autoontological**, i.e. system components themselves define hierarchy of problem domain concepts. Hierarchical combination of frame subhierarchies is in fact a natural implementation of taxonomic ontologies, that can also be naturally extended by production rules to define dynamics of the problem domain.

Hiearchical frame knowledge representation allows multiple **knowledge reuse** due to inheritance, in a way similar to object-oriented paradigm. Inheriting from remote hierarchies allows extending remote knowledgebases with more specific knowledge. Effective clasterisation of knowledge around frame slots minimizes problems of cross-knowledgebase consistency.

2. Distributed Frame Hierarchy Architecture

2.1. Production-Frame Knowledge Representation

Distributed frame hierarchy architecture is based on production-frame knowledge representation that exploits taxonomic hierarchy with inheritance relation and active slots with query procedures and daemons, around which production rules are clustered. Such an approach allows combining in one model static knowledge about the problem in the form of slot values, structural knowledge of the problem domain in the form of frame hierarchy, and dynamic knowledge in the form of attached procedures that drive logical inference.

Thus, frame system state can be represented as $\mathcal{W} : I^2 \to \mathcal{S}$, where I — a set of identifiers, $\mathcal{S}$ — set of slots of the form $\langle v, d, \{Q_i\}, \{D_j\}, \{C_k\}, \sqsubseteq_q, \sqsubseteq_d, \alpha\rangle$, that include current slot value $v \in \mathbb{T}$, default slot value $d \in \mathbb{T}$, set of query procedures $\{Q_i\}$ and set of daemons $\{D_j\}$ with corresponding rule selection strategies in the form of complete order relations $\sqsubseteq_q$ and $\sqsubseteq_d$, set of constraints $\{C_k\}$, etc. Query procedures Q_i are in fact expressions from a set $\mathbb{E}$, constructed according to some defined syntax, and daemons are represented by functions that change system state $D_j : \mathbb{W} \to \mathbb{W}$ (where $\mathbb{W} \ni \mathcal{W}$ denotes the set of frame model state functions) [2]. A set of slot values $\mathbb{T}$ can be of arbitrary structure (it can be, for example, as set of types with dynamic or strict typisation) — for describing inference semantics it would be convenient to consider $\mathbb{T}$ being a complete lattice.

Inheritance relation ":" is induced by a slot with reserved name **parent**: $F : G \Leftrightarrow \|F(\mathbf{parent})\| = G$, which allows formalising **dynamic inheritance**, in which frame parent can be infered in the process of logical inference. Typical for frame systems operation of pattern frame specification can be implemented by including dynamic inheritance rule $F(\mathbf{parent}) \leftarrow \mathbf{match}(F, G)$. Multiple inheritance can also be considered, in which case **parent** slot is assumed to be of list type, and $F : G \Leftrightarrow \|F(\mathbf{parent})\| \ni G$.

Inference semantics is defined in terms of semantic function $\|\cdot\| : \mathbb{C} \times \mathbb{E} \times \mathbb{W} \to \mathbb{T} \times \mathbb{W}$, that is recurrently defined to describe backwards inference, and a function of directed forward inference $\Phi_{\langle f,s\rangle} : \mathbb{W} \to \mathbb{W}$. On a set $\mathbb{W}$ a partial order relation $\sqsubseteq$ is naturally induced by corresponding relation on $\mathbb{T}$, with respect to which $\mathbb{W}$ forms a complete lattice. Since we only consider monotonous inference (which leads to the semantic functions being monotonous with respect to frame states), there exists natural fixpoint semantics based in consecutive application of $\|\cdot\|$ and Φ. Process of combined inference can also be illustrated as a search process in finite state bigraph, defined on a set of states factored by the relation of state equivalence with respect to all rule premises in the knowledgebase. This set can be proved to be finite, which leads to the finite set of steps in any logical inference process.

Semantics of knowledge representation language is defined by function $\mathcal{L} : A^* \to \mathbb{W} \to \mathbb{W}$, that transforms language statements to set transformations, which being applied to empty state $0_{\mathcal{W}}$ leads to the initial state $\mathcal{W}_0 \in \mathbb{W}$, on which $\|\cdot\|$ evaluation function is applied to start the inference.

2.2. Frame model integration with relational databases

Integration of relational databases into the frame model is based on implicit definition of frame set $\mathcal{F}(\mathcal{R})$ by a relational table $\mathcal{R}$, where each table row $\mathcal{R} = \{\langle x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\rangle\}$ defines a frame named $x_1^{(i)}$ (we can without loss of generality consider first key column or $\mathcal{R}$ to be a *naming attribute*) and slots $s_2, \ldots, s_n$ (where $\langle s_1, \ldots, s_n\rangle$ — metadata of $\mathcal{R}$), with values $x_2, \ldots, x_n$ accordingly (i.e.. $\|x_1^{(i)}(s_j)\| = x_j^{(i)}$).

We can also consider an approach where one frame corresponds to a family of table rows accoding to some criteria (for example, an arbitrary SQL subexpression — in this case semantics is defined in terms of oracle function that incapsulates the semantics of SQL statements).

Suggested approach for integration of relational databases and reasoning is in a way a compromise between classical models of **active databases** [3], where relational model is extended by forward inference triggers, and **dedutive databases**, that are rather first-order logic representation of relational structures. Presented approach allows using existing RDBMS software for reasoning over data contained there by inheriting implicit database frames from parent frames that encapsulate dynamic knowledge used in inference.

[2] To describe forward inference that allows one-dimensional inheritance parametrisation we need to define execution context of a daemon in the form of base frame reference — in this case daemon functions would be defined as $D_j : \mathbb{C} \to \mathbb{W} \to \mathbb{W}$.

2.3. Integration with imperative program code

Embedding imperative algorithmic components into frame model is based on similarities between frame knowledge representation and object-oriented and component approach, which allows representing objects and components as frames according to some naming conventions. Semantics of such frames is represented by some external computable oracle function $\mathcal{F}$: $\|F(s_i)\| = \mathcal{F}(s_i)$.

From implementation point of view it is convenient to attach procedures written in host programming language to frame slots. Describing semantics of such procedures is complicated, because of side-effects that can break monotonicity of inference, but under some restrictions they can also be describes as oracle-functions of certain types.

2.4. Distributed frame system

We would call **distributed frame system** a collection of state functions of individual subhierarchies $\tilde{\mathcal{W}} = \langle\{\mathcal{W}_1, \ldots, \mathcal{W}_n\}, i\rangle$, where one subhierarchy is designated as current, which we would also denote by $\{\mathcal{W}_1, \ldots, \overline{\mathcal{W}}_i, \ldots, \mathcal{W}_n\}$. Expression syntax $\mathbb{E}$ would be extended by mobile $\Diamond$ and static $\blacklozenge$ **remote references**. We would also without loss of generality suppose that frames in subhierarchies are uniquely named. If it is not the case, we can switch to complete frame naming, where frame name is prepended by subhierarchy index.

We would define **combination operator** for state functions as follows:

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

This operator is symmetric and associative, which allows to extend it to arbitrary number of subhierarchies. We would call $\mathcal{W} = \bigstar\tilde{\mathcal{W}} = \bigstar_{i=1}^{n}\mathcal{W}_i$ a **state function** of distributed frame system. Such a function would be a state function of some local frame hierarchy[3], which we would call **equivalent induced local system** for the original distributed system.

For combining hierarchies we can also define operations of hierachy inheritance as follows:

$$\mathcal{W}_1 \underset{f}{\blacktriangleleft} \mathcal{W}_2 = \{\mathcal{W}_1, \mathcal{W}_2[object.parent \leftarrow \blacklozenge\mathcal{W}_1.f]\}$$
$$\mathcal{W}_1 \underset{f}{\triangleleft} \mathcal{W}_2 = \{\mathcal{W}_1, \mathcal{W}_2[object.parent \leftarrow \Diamond\mathcal{W}_1.f]\}$$

2.5. Distributed inference semantics

Logical inference in the distributed system can be defined in terms of equivalent local system, or by introducing a semantic function for distributed inference $[\![\cdot]\!] : \mathbb{C} \times \mathbb{E} \times \tilde{\mathbb{W}} \to \mathbb{T} \times \tilde{\mathbb{W}}$, that operate over sets of state functions.

For computing static and mobile references this function would define interoperability of individual subhierarchies:

$$[\![\Diamond\mathcal{W}_j(f,s)]\!]_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'} = [\![f.s]\!]_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}_i \bigstar \mathcal{W}_j},\ldots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'}$$
$$[\![\blacklozenge\mathcal{W}_j(f,s)]\!]^{F}_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'} = [\![f.s]\!]^{\blacklozenge\mathcal{W}_i(F)}_{\{\mathcal{W}_1,\ldots,\mathcal{W}_i,\ldots,\overline{\mathcal{W}}_j,\ldots,\mathcal{W}_n\}\to\tilde{\mathcal{W}}'}$$

while for expressions that do not contain remote references it would be defined by local semantics:

$$[\![E]\!]_{\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}_i,\ldots,\mathcal{W}_n\}\to\{\mathcal{W}_1,\ldots,\overline{\mathcal{W}}'_i,\ldots,\mathcal{W}_n\}} = \|E\|_{\mathcal{W}_i\to\mathcal{W}'_i}$$

It can be demonstrated that for systems that use only mobile or only static interoperability distributed inference semantics is equivalent to that of local for equivalent induced system. For combined interoperability in the general case it is not true, however, for some classes (for example, for purely polymorphic subhierarchies, where all rules correspond only to one frame, without direct references to slots of other frames, and all slot references are made through current frame reference $this$) equivalence still holds.

3. Implementation Details

Distributed frame hierarchy architecture is implemented in JULIA toolkit (Java Universal Library for Intelligent Applications), that has been developed by the author. The toolkit is implemented in Java (more than 13000 lines of code) and is registered in the Russian Patent agency (certificate No. 2002610609 from 25.04.2002).

3.1. Toolkit Architecture

For defining knowledgebases specific knowledge representation language had been developed, called JFMDL (Julia Frame Model Definition Language). Knowledgebase formulated using that language is then translated into a collection of Java objects representing frame hierarchy and attached rules. Backward-chaining production rules and user queries are translated into query procedures, and forward chaining rules — into daemons, that form a sort of Rete network. Toolkit uses modified version of Rete algorithm, where slot values themselves serve as α-memory, and β-memory is replaced by using special expression trees with intermediate result caching.

An internal representation obtained after translation (so-called **frame world**) can be directly used for logical inference, or can be serialized into external file or stored in object database for further use in the logical inference. Loading such a model from file or database can be performed using few JULIA API calls.

[3] Omitting $\Diamond$ and $\blacklozenge$ operations that do not correspond to local syntax.

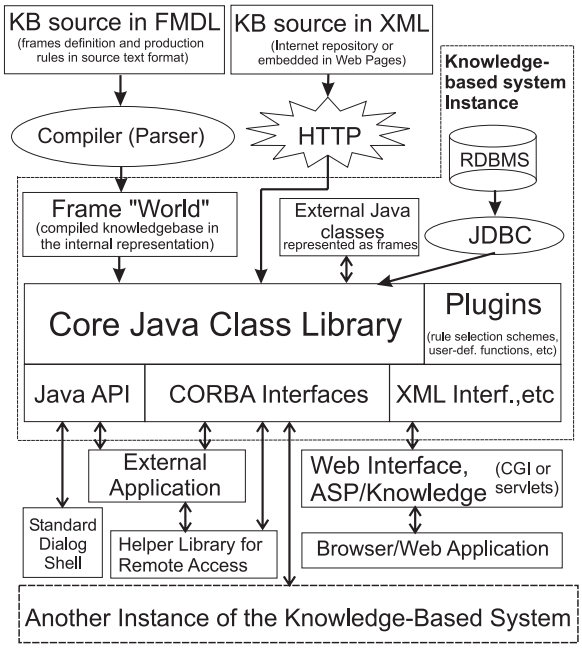

Figure 2. JULIA Toolkit Architecture

Apart from relational database and imperative object interfaces described above, the toolkit allows base programming language implementation of arbitrary rule selection strategies for forward and backward inference, arbitrary daemons and query actions, user functions that can be used in expressions and forward-chaining actions, etc. Moreover, due to the open nature of JULIA API, it can be extended in any way to suite desired functionality (for example, arbitrary data type or fuzzy inference can be implemented).

3.2. Implementation of remote inference

A set of all frames located on one network node (more precisely, in one namespace) form a **frame world**. When computing static remote reference a **remote call** is made using some remote interoperability protocol (CORBA, RMI). For each remote frame a local object called **proxy frame** is created, that forwards all slot requests to the remote network node. Such proxy frame can also cache slot values, thus minimizing network traffic when slot value is queried more then once.

When frame F **statically inherits** frame G on some other node, the inference process can synchronously move from one node to another. In the process of inference F can query the slot value for some slot of remote parent frame G, giving execution control to remote inference engine and passing remote self-reference as a base frame. F in the remote hierarchy is, in turn, a remote frame, and assigning values to its slots would be made via remote callback. Thus, during inference dynamic production rules associated with G will be used, but only static knowledge in the form of slot values would be exchanged over the network.

Distributing forward inference is more complex task, because all daemons have to be linked into Rete-like network prior to starting inference. If forward inference rules are limited by a frame world boundaries, forward inference is similar to backward: when a slot value is assigned daemons for all parent frames up the hierarchy are executed, causing remote inference when needed. However, when there are rules that use slots from different frame worlds in rule premises, all such worlds have to register corresponding daemons upon initial connection, and see if some rules can be executed. Modified algorithm used in the toolkit limits the used of cross-frame forward inference rules, thus providing higher efficiency and remote inheritance transparency.

Described functionality of remote inference requires active two-way communication between network nodes using some remote call protocol like CORBA or RMI. In some cases it is desirable to create passive repositories of structured knowledge that can be loaded on demand using HTTP or FTP standard protocols. In such cases **mobile inheritance** is useful, in which dynamic knowledge in the internal representation is transferred over the network and then used by the local inference engine. It can be implemented by loading the whole remote hierarchy (according to the semantics described above), parent frame, or only required rules. Apart from mobile inheritance, it is possible to use references to remote knowledge repositories or rule collection. In the latter case the internal representation of a knowledgebase would consist of a central hierarchy skeleton and a set of files, one for each rule, that would be loaded through the network on demand. Such approach can be used for minimizing network traffic in client-server remote consultations.

When combining knowledge from several remote sources multiple inheritance can be used — in which case remote calls to several remote hierarchies would be made consecutively (according to some programmatically defined criteria) until the solution is obtained. In some cases instead of multiple inheritance **pseudo-multiple**, or consecutive inheritance can be used, where frame has only one parent at any given moment, that is changed according to some forward-chaining rules, meta-rules or external algorithm. Such an approach is complimentary to multiple inheritance and can be used together with it, however, using pseudo-multiple inheritance together with forward chaining rules should be avoided, because of less transparent semantics that does not directly correspond to the multiple inheritance.

3.3. Toolkit Interfaces

JULIA toolkit provides the developer of an intelligent system with JULIA API, which can be used for performing logical inference from any Java-based applications. Availability or CORBA interfaces provides means for interfacing corporate business logic and software systems developed in any CORBA-aware programming language. In both cases the interface is bidirectional, i.e. not only the toolkit can be invoked from information systems, but also imperative components and databases can be used as a part of frame hierarchy, providing seamless means to reason about corporate data.

The toolkit also includes utilities for using it as an expert system shell for creating distributed dialog expert systems in the form of Java applets, servlets or applications. Small run-time library footprint (less then 100 Kb) allows using distributed reasoning functionality in mobile devices like PDAs and cell phones that conform to MIDP or CDC profiles. To fully exploit wireless functionality for building distributed knowledge spaces with mobile devices as terminals some further development of the toolkit is required to use different XML-based distribution protocols (SOAP), but conceptually proposed model scales very well into the mobile environment. Moreover, we believe that eventually this architecture can be integrated into other evolving standards and protocols within modern computer networks, including Semantic Web initiative of the W3C.

4. Conclusion

Presented architecture of distributed frame hierarchy can be effectively used in solving problems of distributed knowledge sharing and reuse. JULIA toolkit provides a cross-platform implementation of such an architecture, and has been used in a number or real-life projects. For example, it has been used in the development of intelligent information system for diagnosing and planning treatment tactics for patients with benign prostatic hyperplausea (BPH), which is now used in the urology department of Botkin state hospital in Moscow [5]. Based on the toolkit, some prototype systems have been created, including ontological search system for annotated hypermedia collections [6], systems of distance education in the field of Logic Programming, expert system for web resource promotion campaign planning, and some more.

Architecture of distributed frame hierarchy is very scalable, and allows constructing traditional expert systems with remote consultation facilities, taxonomic knowledge repositories that can be later extended with more specific knowledge and used collectively in solving specific domain problems, and in the nearest future to create wireless distributed knowledge spaces with mobile and handheld devices used as user terminals, and central servers holding most accurate ontological problem domain knowledge. Since interaction semantics for nodes in distributed frame hierarchy is well-defined, it provides natural means of using knowledge repositories in e-commerce B2B and B2C systems, where certain price can be charged per consultation, per number of rules used, or on subscription basis. With clear auto-ontological production-frame knowledge representation and relatively straightforward cross-platform implementation, ability to interface legacy code, relational and object-oriented databases, as well as corporate business logic, we believe that distributed frame hierarchy architecture and JULIA toolkit can be effectively used in problems of knowledge sharing and reuse in evolving intelligent computer networks.

References

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

[2] 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.

[3] Hanson N.H.,Widom J. An Overview of Production Rules in Database Systems. In *the Knowledge Engineering Review*, Vol.8, No.2, 1993. pp.121–143.

[4] Ramakrishnan K., Ullman, J. A Survey of Research on Deductive Database Systems, *Journal of Logic Programming*, 23(2), 1995. – pp. 125–149.

[5] Lukianov I.V., Soshikov D.V. et al. Expert analysis using Intelligent Information System for Diagnosing and Developing Treatment Tactics for Patients with Infravesical Obstruction. Moscow Medical Journal, No. 5–6. pp. 29-31.

[6] Sizikov E.V., Soshnikov D.V. Jewel: Ontology-based Search System for Intelligent Search in Internet and Intranet Networks. Moscow, Electronic Journal “Trudy MAI”, No.7, 2002. (In Russian)

[7] Soshnikov D. Logical Inference based on Remote Invocation and Inclusion in the Systems with Distributed Frame Hierarchy. Moscow, Vuzovskaya Kniga, 2002. (In Russian)