INTELLIGENT AGENT FOR WEB WATCHING: LANGUAGE AND BELIEF SYSTEM

A. G. Kolonin

Aigents Group
630090, Novosibirsk, Russia

УДК 004.896

In this paper we describe pre-requisites for development of an intelligent computer software agent for watching information on the web in favor of human users. First, we introduce an „interlingua" language for textual and verbal communication between a human and an agent, sufficient to upload any user's beliefs into the agent ontology and to convey further interactions between the two. Next, we discuss construction of the agent foundation belief ontology and extend it to a specific web-watching domain. Finally, the language and belief are tested against real-world interaction scenarios.

Key words: artificial language, artificial intelligence, belief system, human-computer interaction, interlingua, ontology, software agent, web search.

Introduction. There are two complementary trends in the modern information technology development such as creation of artificial general intelligence agents [1] and emergence of the market of intelligent consumer devices or the so-called „Internet of Things" [2]. Both converging technologies mean wide spreading of semi-intelligent artificial software agents embodied in various software services and utilities as well as hardware consumer electronic devices, interacting with each other and human masters. At the same time, the interaction is assumed to be carried out in the context of knowledge structured by means of „Semantic Web" technology, where each of the agents has its own belief ontology while all communicating agents share some common foundation ontology. In such „Internet of Things", artificial and real human agents are talking about various „things", representing semantic entities with meaningful relationships between them, possibly including other agents. Then, in the beliefs of the peer agents, the agents are *„things"* themselves, so the things (cars, refrigerators, thermostats, computers, smartphones, people) are *„talking about things"* — everyone about each other.

In the following research we describe the requirements for development of a software agent specialized in watching the web and monitoring specific web resources, looking up for the topics of interest provided by a human master. For this work we will be assuming that the minimum capability of any agent of the kind would be to maintain bilateral conversation with a partner (for instance, human) using a simplistic semi-natural English dialect of „interlingua" Agent Language (AL) transported as a plain text on any protocol such as TCP/IP, HTTP, IRC, email, etc. Having this provided, the same language can be used as a communication protocol to create a top-level graphical user interface or a speech interface using third-party text-to-speech and speech-to-text technologies. Using this language, we construct agent's belief ontology, as a foundation which can be extended and enriched further in the course of communication between an agent and a human — so that the agent can gradually acquire the world of human knowledge and learn behavioral schemata usable for humans.

Requirements. Such an agent would need to be capable of knowing the list of sites or web pages to watch for the target data, as well as the list of topics to be tracked. The patterns or templates to be used for this can be learned by an agent in the course of experiential learning or be pre-set by a human operator. The agent would also need to have a list of its peer contacts that must be updated with the collected news. There could be also a variable for supplement of reward/punishment, so that templates as well as behavioral patterns can be learned by an agent by the trial-and-fail method with a feedback from the master. The acting capabilities of this agent, besides communication with human peers, would include downloading HTML pages from the list of sites, matching templates in the pages and extracting specific values of interest from the findings. The agent should be capable to carry out the following activities:

— Get familiar with new personalities (human *users*), represented by names (to personify), email addresses (to send notifications to), some specific information (like date of birth — in order to resolve full namesakes) plus some secret information (to confirm identity).

— Establish verbal (chat) conversations with human users (and possibly other agents of the kind) having the identity of the peer confirmed by secret information provided.

— Provide an ability to recall or reset the secret information (if forgotten) by email (if supplied).

— Accept specification of some number of the web *sites* of interest provided by a human participant of the conversation.

— Accept specification of some *things* interesting for a human participant of the conversation, associated with these sites.

— For any thing of interest, optionally specify some textual *patterns* indicating occurrence of these things in the web site text.

— For the patterns, be able to manually configure indicative combinations of keywords/tokens (e. g. „house sale“), variants and lemmas (e. g. large, huge, big, bigger biggest, etc.) and variable placeholders (e. g. [number], [date], [text]).

— Keep monitoring all sites of interests and respective things given by all familiar users and, if any new findings are discovered, provide users with *news* updates by chat (if there are open conversational sessions) and email, providing information about the time, site, particular thing and textual context of its experience.

— Be able to obtain feedback from a user supplied with the news in respect to applicability of the news — so the user can either confirm relevance and novelty (e. g. „good news!“) of the information or agree with relevance but deny novelty (e. g. „good, but don’t show it again anymore“) and finally deny relevance at all.

— Be able to learn from user feedback and infer the extensions of the user’s pattern on its own, so the former can be overridden eventually with the inferred pattern if it provides more relevant and novel news.

— Maintain a conversation with users letting them explore the sites and things of interest being operated by an agent, and let users to amend them — so the user can ask for lists of sites, things and links between them, add, remove or amend sites and things and their properties.

— Maintain a conversation with users retaining specific properties of humans such as notification frequency (hourly, daily, weekly, monthly) and the time to send the news.

— For the operations described above, keep the privacy of the users, isolating their private data from the others’ data so all addition, removal and amendment operations are applied to the image of a human user in the agent's belief system only — for each of the humans the agent is familiar with.

To support all of the above, we need to come up with a language capable to convey interactions between a user and an agent as well as to maintain extensible belief ontology of an agent's self.

Language. There are multiple studies in the area, targeting development of an „interlingua" usable for communication between software agents as well as between humans and these agents [3–7]. Currently, the long-term goal of many companies and projects is enabling computer systems to speak real human languages. However, we assume it will not only need substantial time for further research and development, but it will also require enormous computational resources which are typically not available in case of consumer devices. Respectively, we endeavor to come up with human-computer „interlingua" language trying to follow a more practical short-term approach. Below is a brief list of requirements for such language.

— The language has to possess *communication-wise symmetry* so that any peer agent of the conversation can transmit any kind of statements (declarative, imperative, interrogative, etc.) referring to the context of earlier statements made by any participant. It would be very different from the existing asymmetric client-server languages such as SQL, SPARQL, XML or JSON (unless bilateral implementations of such technologies are used).

— The language has to be *semantically expressive* being able to convey fundamental concepts of structured information such as class, object, attribute, value, name, set or array. It would complicate attempts to use synthetic human languages, for instance, simplified English or Lojban [3].

— The language should be *semantically open and ontologically transparent*, so that description and extension of the schema and object model of the subject area for any domain of communication could be done in the same linguistic structure using the same syntax as declarations, directives and interrogations about the data referring to that schema or model. This requirement cannot be met with XML [7] and JSON [5] since the schema is not expressed in the same medium and to some extent can be met only with RDF [7].

— The language should be *compact and easily parse-able* so large pieces of information can be transmitted and processed with less computational overhead which is especially important for wireless interactions with low bandwidth and embedded devices with low memory. Here is where XML-based encodings hardly have a promising future.

— The language should have the *structure congruent to one of prevailing human languages* (for instance, English) so the cost of learning it for humans can be kept to a minimum. Ideally, it should be comprehensible by humans with no special training at all, so that any intelligent consumer device could establish conversation with its owner out-of-the box. There are no good examples of such kind now, except Lojban+ [3] which still requires humans to learn a new natural language.

— The language should be capable of expressing any information contained in „Semantic Web" so any RDF [7] or Turtle [4] syntax can be transparently translated into it without any information loss. In addition, it should be possible to *express higher-order semantics or hypergraphs* with triples connected not only to the nodes (vertices) of a graph, but to the links (edges or arcs) of the graph as well.

In the following discussion we will consider how the above requirements can be satisfied with the Agent Language that we are suggesting.

Starting from the cases and examples described above and taking many other cases and examples into account, the following language called „Agent Language“ (AL) can be developed. Below is the language grammar specification briefly expressed in EBNF [8] form (without going into details such as defining <number>, <date>, <time> and <string>).

```
<message> := ( <statement> | <acknowledgement> )*
<acknowledgement> := ( ’ok’ | (’true’ | ’yes’ | <number>) |
                                                   (’no’ | ’false’ | 0) ) ’.’
<statement> := <interrogation> | <confirmation> | <declaration> | <direction>
<interrogation> := ’what’ ? <expression> ’?’   (* “open” incomplete graph *)
<confirmation> := ’if’ ? <expression-set> ’?’  (* “closed” complete graph  *)
<declaration> := ( <expression-set> ) ’.’      (* “closed” complete graph *)
<direction> := ’do’ ? <expression-set> ’!’     (* “closed” complete graph *)
<expression> := <term> (’ ’ <term>)*           (* separated by spaces *)
<expression-set> := <all-set> | <any-set> | <seq-set>
                                               (* different kinds of sets *)
<term> := <negation>? ( <anonymous>? | <self> | <peer> | <id> | <name> |
                                                     <value> | <qualifier> )
<qualifier> := <expression> | <expression-set>
<any-set> := <or-list> | ( ’{’ <or-list> ’}’ )
<all-set> := <and-list> | ( ’(’ <and-list>  ’)’ )
<seq-set> := <then-list> | ( ’[’ <then-list> ’]’ )
<or-list> := <expression> ( (’,’ | ’or’ ) <expression> )*
<and-list> := <expression> ( (’,’ | ’and’ ) <expression> )*
<then-list> := <expression> ( (’,’ | ’next’ ) <expression> )*
<negation> := ’not’ | ’no’ | ’~’
<anonymous> := (’there’ (’is’|’are’)) | ’any’ | ’anything’ ?
<self> := ’my’|’i’|’we’|’our’
<peer> := ’your’|’you’
<value> := <number> | <date> | <time> | <string>
```

Here is a brief description of the language, under an assumption that the purpose of the language is to express the full set of operations on a semantic hyper-graph consisting of terminal nodes and typed links between the nodes, links and sets of nodes and/or links.

— *Message (Document)* — a series of semantically consistent statements and optional acknowledgments to the former statements, bounded physically by means of external communication protocol or storage medium.

1) Acknowledgement — used to provide response to the statements other than interrogation. For declarations and directions „ok“ is returned. In case of „if“-style confirmations „false“, „no“ or zero can be returned for failed assertions, while „true“, „yes“ or a number of evidences (count of the applicable subgraphs) satisfying the assertion (operating like COUNT(*) function in SQL) — for successful assertions.

2) Statement (Sentence) — an association of semantically connected expressions, separated by delimiters such as commas, exclamation and interrogation symbols (’.’, ’!’, ’?’) or by external formatting markup (e.g. HTML blocks, list items, table cells, etc.). There can be four kinds of statements, varying in the degree of certainty in respect to the contained expressions. That is, interrogation means a speaker has no idea about the complete matter of an expression being

asked about, confirmation means the matter is guessed and there is a need to have it confirmed, declaration expresses the matter with some confidence while direction is intended to force the listener to accept the matter communicated by the speaker.

a) *Interrogation* statement is denoted with question mark at the end and can be used to perform a query against the specified incomplete (or „open“) subgraph to retrieve specific parts of it, effectively representing complementary subgraphs needed to make the expression in the statement complete (similarly to SQL or SPARQL query). It can be preceded with „what“ (i.e. „which“) keyword as a clue for text parser.

b) *Confirmation* statement is also denoted with the question mark at the end and can be used to check the existence of certain complete (or „closed“) subgraph or a set of subgraphs or truth value of an assertive expression or expressions encompassing them. It can be preceded with „if“ (i.e. „whether“) keyword as a clue for text processor, so it could be distinguished from interrogation without semantic analysis of the query.

c) *Declaration* statement is denoted with the period mark at the end and is used in conversation purely for the declaration purposes, so the receiver of the message can handle it at its discretion. It contains a single expression or a set of expressions encompassing complete assertions or „closed“ subgraphs. This is what is returned upon execution of „what“-style interrogations or can be used to load knowledge into an agent’s belief system. It is the default kind of statement to be expected by a language processor, so it is not supplied with a clue keyword.

d) *Direction* statement is denoted with the exclamation mark. Beyond just declaring the essence of the matter like the above-mentioned declaration does, it is used to force the partner receiving the message to accept the assertive expression or expressions, so they are incorporated in their belief graph. It can be preceded with „do“ keyword as a clue for the text processor.

— *Expression (Phrase or Proposition)* — an elementary constituent of a sentence. It is an ordered association of terms separated by spaces and bound to the same semantic entity, or a subgraph encompassing the entity in the following way. 1. The subgraph represented by expression can be „closed“ or complete, so there is no ambiguity in its expression and no hidden variables implied. The subgraph can also be „open“ or incomplete, encompassing unresolved variables or „hanging links“ in the graph, so it can be used to create „what“-style interrogations being asked to resolve these variables. 2. The subgraph can be defined like a locally restricted path in a hyper-graph, most likely a set of mutually interconnected RDF or Turtle expressions. 3. Unlike RDF or Turtle expressions, the length of the single expression path is not restricted, so besides subject, predicate and object arguments, an expression can represent subject-predicate-predicate-...-predicate-... inference chains of arbitrary length (such as „*my favorite bank customer mother maiden name*“ for instance). 4. Unlike Turtle, which can have multiple predicate-object clauses as well as multiple object arguments, an expression can have multiple subjects and predicate verbs as well, which may be grammatically supported with using implicit or explicit parentheses (’(’ and ’)’), braces (’{’ and ’}’) and brackets (’[’ and ’}’) instead of using Turtle’s colons and semi-colons (making the „*you and me will work and live together forever*“ possible). 5. Unlike Turtle, potentially any terms can be complete expressions on themselves, rather than static resources, with some restrictions to be implied by particular implementations on the language (so an English phrase „*anyone taller than 1.8 meters is tall*“ is a valid AL expression „*(taller 1.8) is tall*“). 6. Expressions can be composed together into an expression set, qualifying composite semantic entities, or composite qualifiers — as is described further.

— *Argument (Term)* — an atomic constituent of an expression, denoting either a node or a link in a graph or a whole set of nodes or links representing a respective similar semantic entity. Any term can be preceded with unary negation operator (expressed as ’no’, ’not’ or ’~’) inverting the subgraph scoping from inclusion to exclusion. The following terms can be used. 1. Anonymous term denoting an unlabeled entity not unassociated with any prior experiences, to be identified solely by further arguments of an expression. 2. Self-reference grammatically denoted as „My“ or „I“ (or „We“ or „Our“), identifying the first-person of an agent. 3. Peer-reference identified by „Your“ or „You“, identifying the partner of the communication. 4. Reference by Id or URI which may be used to refer to any subjects and objects in the course of internal non-human communications between non-human agents. 5. Reference by Name is intended to identify any named entities including persons, classes, properties of objects, verbs and operator symbols such as ’+’, ’-’, ’=’, ’>’, ’<’. 6. Value can be encompassing any semantic terminals such as finite numbers, literal strings, characters, times and dates. 7. Qualifier is intended to refer to a complete semantic entity or a set of entities applying a hierarchy of expressions that restrict the graph down to the target set of relationships in the graph. It could be either singular expression (like „*big tree*“) or a composite qualifier represented by an expression set (like „*big green tree on the other side of the street next to the parking lot*“).

— There can be three kinds of expression sets or composite qualifiers where each of them can associate a list of qualifying expressions recursively — with different logical and sequential operations implied to associate the expressions in the set, as follows. 1. Disjunctive qualifier representing „OR“-style logical association where any expression in the list can be used to qualify the entire term. 2. Conjunctive qualifier representing „AND“-style logical association where all expressions in the list have to be resolved in order to qualify the term. 3. Successive (Ordered) qualifier representing „NEXT“-style logical and sequential association where all expressions in the list have to be resolved strictly in the specified order.

Trying to use the grammar of AL language to generate some kind of a human-friendly speech might get little weird for a native English speaker. This is because the roles of terms in the expressions in English are driven by the grammatical order, prepositions and forms of the verb „to be“. However, it might be easier to accept for Portuguese and Russian speakers where the same grammatical statement can have a different mood (interrogative, declarative, imperative) depending on the tonal modulation, as shown in the following table.

Table

English	**AL (no clues)**	**Russian (with tonal modulation)**
What is your feeling?	*Your feeling?*	*Твое ощущение? (tone up)*
If your feeling is good?	*Your feeling good?*	*Твое ощущение хорошее? (tone up)*
Your feeling is good.	*Your feeling good.*	*Твое ощущение хорошее. (tone neutral)*
Have your feeling good!	*Your feeling good!*	*Твое ощущение хорошее! (tone down)*

Use of composite qualifiers can be presented with the following mapping where expressions on the left are proper AL expressions built with use of braces, brackets and parentheses (which might be easier for computer text parser) while the expressions on the right are glued with conventional prepositions (which might be more convenient for human ear). However, it should be noted that the right-side expressions present the ambiguity of the logical term grouping that cannot be resolved by a simple parser, incapable to detect semantic roles of predicate and object terms.

I (can (eat, sleep), want (dance, sing)). I can eat and sleep and want dance and sing.

I {can (eat, sleep), want (dance, sing)}. ⇔ I can eat and sleep or want dance and sing.

I (can {eat, sleep}, want {dance, sing}). ⇔ I can eat or sleep and want dance or sing.

You [eat {rice, meat}, drink {juice, water}]! ⇔ *You eat rice and meat next drink juice and water!*

From the perspective of using the language to represent semantic expressions, the following example demonstrates how the AL can be used to express statements in conventional term logic or Turtle syntax. In the last two examples, no conversion to Turtle is applicable at all.

AL		**Term logic**		**Turtle**
A C (D, E).	⇔	*A C D. A C E.*	⇔	*A C D, E.*
A (C D, F G).	⇔	*A C D. A F G.*	⇔	*A C D; F G.*
A (C (D, E), F (G, H)).	⇔	*A C D. A C E. A F G. A F H.*	⇔	*A C D, E; F G,H.*
(A, B) C D.	⇔	*A C D. B C D.*		
(A, B) (C (D, E), F (G, H)).	⇔	*A C D. A C E. B C D. B C E. A F G. A F H. B F G. B F H.*		

The language by itself defines nothing but the ways to express certainty and modality, the structure of statements and the means to refer to the subject in the first person, in the second person or any subject in the third person referred by name, identifier or qualifier. In order to augment communication with the features specific to possession, inheritance, time and place, there is a need for a minimum set of predicate verbs supported by an ontology employed by an agent speaking the language. In the next section we will build up such a „belief system“ for the agent of the purpose.

Belief Ontology. In order to construct agent's domain-specific ontology, first we define a foundation ontology used to express everything else [6]. First of all, we assume any thing (semantic entity) must have a unique id (owned by the entity) and possibly may have one or more names (potentially shared with other namesake things). Further, we rely on such semantic relationships between things as „is“ (being an instance of something), „has“ (possessing certain properties) and „does“ (be capable of doing specific actions), as expressed in AL below. The semantic relationships are represented by properties (effectively — typed semantic links or ternary relationships) which can be potentially assumed obligatory for a thing (so the thing must have at least one relationship of a type). Also, some of the properties may reflect others being reverse to them by meaning. Bold text and capitalization in the following statements do not convey syntactic meaning and are used solely for the illustration purposes, distinguishing terms in subject, predicate verb and predicate object roles.

— Thing **has** Id, Name, Is, Has, Does, Times.

— Id **is** Property, Owned, Number, Obligatory.

— Property **has** Type, Source, Target.

— Type, Source, Target **is** Thing.

— Name **is** Property, Shared, String.

— Is, Has, Reflects **is** Property, Shared, Thing.

— Does **is** Property, Shared, Action.

— Action **is** Thing, Executable.

— Times **is** Property, Shared, Time.

— Time **is** {Today, Yesterday, Tomorrow, Date-time, Date, Month, Year}.

— Date-time **has** Daytime, Date.

— Time **has** Events.

— Events **is** Property, Shared, Thing, **reflects** Times.

In terms of object-oriented design, the is/has/does relationships identify such relationships as inheritance (and opposing instance), attributes (or member variables) and methods (or member functions) — respectively. It should be noted that, unlike many other ontologies, we do not attempt to distinguish different kinds of inheritance (such as inheritance and instance) explicitly, so that instances of classes and objects are all subclasses of one generic abstract thing [6]. Also, we assume the action is just a specific executable kind of thing, with the lifespan restricted by the execution time and runtime variables being member attributes. The meaning of other things such as name, number, string, daytime and date involved below should be obvious. On the basis of the foundation ontology described above, we can construct the following domain ontology.

— Agent **is** Thing, **has** Peers (**is** Property, Agent).

— Agent **has** Feels (**is** Property, Shared, {Good, Bad}).

— Self, User **is** Agent.

— User **has** Surname, Birth date, Email, Secret question, Secret answer, Update time, Update period, Things, Shares, Likes.

— User has Sensitivity threshold (is Percentage (is Number, is {0, 1, ..., 99, 100})), Seeing shares (is Toggle (is {On, Off})), Keeping days (is Number), Basic privacy (is Toggle), Check cycle (is {Hour, Day, Week, Month}), Update time (is Daytime), Telling news (is Toggle), Emailing news (is Toggle).

— Surname, Email, Secret question, Secret answer **is** Property, Shared, String.

— Birth date **is** Property, Shared, Date.

— Update time **is** Property, Shared, Time.

— Update period **is** Property, Shared, Period.

— Email **is** Property, Shared, String, Obligatory.

— Things **is** Property, Shared, Thing.

— Shares **is** Property, Shared, Thing.

— Site **is** Thing, **has** Links (**is** Property, Shared, Site).

— Thing **has** Users (**is** Property, Shared, User, **reflects** Things).

— Thing, Site **has** Patterns, Times, Users.

— Patterns **is** Property, Shared, Pattern, Obligatory, {String, Lemma, Frame, Thing}.

— Pattern, Lemma, Frame **has** Patterns.

— Site **has** Topics (**is** Thing).

— Things **has** Origins (**is** Site).

— Topics **reflects** Origins.

— User **things** Site.

There are some key points in the ontology worth specific mentioning. Fist of all, we will assume that an agent would be able to maintain internal belief of a peer (be it a human or another computer agent) it is interacting with, so the things property is used to connect a user (representing an agent's peer) to any thing in the agent's self mind. That is, the belief subgraph of a user can be built using that property. Next, for the simplicity of further implementation, we will associate the context of communications between an agent and its peers into a belief sub-graph having it to keep all what is typically maintained in communication sessions (it means that no more than one communication session can be established between an agent and its peer). Specifically, the interaction graph for a user would be represented by a behavioral schema of its does subgraph, described in the implementation section below.

The part of the agent's ontology dedicated to operations with text can be described as pattern, which is a fundamental entity connecting things in the ontology with their textual representations or templates. In general, pattern can be anything starting with an individual symbol or glyph and ending with a complex hierarchy of other patterns. In our design, we consider that pattern could be either a text string (of 1 or more characters or glyphs), a lemma (either linguistic lemma representing a base word plus all its inflections or psycholinguistic lemma representing a set of synonyms conveying some particular meaning), a frame (a set of patterns to be stated in order, as defined per [9]) or any thing placed in the parent pattern as a variable to be resolved by pattern instance in a real text. Notably, the lemma and frame can have patterns so hierarchical textual templates are possible.

Testing Language and Belief Together. When an agent possesses the belief system described above, it enables support to the following communication scenario, expressed with the Agent Language grammar. The following dialogues denote a human user as H: and an agent as A:. Italic text is used to mark omittable parts of textual communication, which can be inferred by the conversational context. On the right side, comments after double slashes indicate what is going on during the conversation.

H: *My* login. // Login

A: *Your* name, surname, birth date, email? // . . . possible identifiers asked

H: *My* name John, email john@doe.org. // . . . unique ones supplied

A: *Your* secret question, secret answer? // Registration

H: *My* secret question „pet name“, secret answer „Toby“. // . . . combination retained

A: *Your* pet name Toby? // . . . „password“ re-entered

H: Yes. // . . . „password“ matched

A: Ok.

. . . // Conversation

H: *My things* site http://www.weather.com, http://www.accuweather.com .

A: Ok. // . . . sites retained

H: *My things* site no http://www.accuweather.com, http://weather.yahoo.com .

A: Ok. // . . . sites updated

H: *My things s*ite?

A: *Your things site* http://www.weather.com, http://weather.yahoo.com/.

H: *My things* temperature, storm. // . . . adding new things to belief

A: *Your things* temperature, storm has, patterns? // . . . asking for unresolved details

H: Temperature has value (is number), scale (is {'°C ', '°F'}), patterns [temperature scale Novosibirsk Russia].

A: Ok. Storm has, patterns? // . . . keep resolving details

H: Storm has region. Region is {Northwest, Southwest, Midwest, North Ural, South Ural, Northwest Siberia, Southwest Siberia, Northeast Siberia, Southeast Siberia, North Far East, South Far East}.

A: Ok. Ok. Storm patterns? // . . . keep resolving details

H: Storm patterns [{'storm', 'storms'}, {approach, move} region]. Approach is {'approach', 'approaching', 'approaches'}]. Move is {'move', 'moving', 'moves'}.

A: Ok. Ok. Ok. // . . . resolving is complete

H: *My* things? // . . . „if“-style asking count

A: *Your things* 3.
H: *My* things name?
A: *Your things* name site, name temperature, name storm.
H: Temperature patterns?
A: *Temperature patterns* [{'storm', 'storms'}, {approach, move} region].
. . . // Conversation
H: *My things site* http://www.weather.com, http://weather.yahoo.com topics temperature.
A: Ok.
H: *My things* storm origins http://www.weather.com .
A: Ok.
H: *My things site* http://weather.yahoo.com topics?
A: *Your things site* http://weather.yahoo.com *topics* temperature, storm.
H: *My things* storm origins?
A: *Your things storm origins* http://www.weather.com .
. . . // . . . pushing news
A: Storm times today, region Southwest Siberia, origins http://www.weather.com?
H: Yes. // . . . positive feedback, true belief
A: *Your* feels? // . . . indirect feedback polled
H: *My feels* good.
. . . // Conversation
H: Storm times yesterday, region Southeast Siberia. // . . . authoring news, true belief
A: Ok.
H: Storm times today, region Southwest Siberia, origins http://www.weather.com!
A: Ok. // . . . learning pattern
H: Storm times yesterday, region? // . . . polling news
A: *Storm times yesterday, region* Southeast Siberia.
H: Ok. // . . . positive feedback, true belief
H: Storm times today, region, sources? // . . . polling news
A: Storm times today, region Northeast Siberia, origins http://www.weather.com.
H: No. // . . . negative feedback, false belief
H: *You* feels good?
A: Yes.
H: *You* feels bad! // . . . indirect feedback pushed
A: Ok.
H: No storm times today, region Northeast Siberia. // . . . false belief, decrement evidence
A: Ok.
H: *My things site* http://www.weather.com time today, yesterday topics?
A: *Your things time* http://www.weather.com *time today, yesterday topics* storm times today, region Northeast Siberia, Southeast Siberia.

The presented level of human — agent communication will not only enable a human user to control an agent monitoring the web for the user's benefit, but will also let the agent possibly learn new text patterns and semantic associations and evolve a behavioral schemata made of elementary actions instead of having them explicitly specified by the user.

Conclusion. The AL language suggested for communication between a computer software agent and a human user seems compact enough for transmission and visual comprehension, easy to read and write for an average human (without special computer knowledge) and easy to parse into semantic graph operations for computer programs. In the written form, the ambiguity can be easily resolved with use of clue keywords and braces and parentheses. In the spoken form, however, should one be implemented, or while being typed in by a human, there may be a need for ontology-based disambiguation techniques so only expressions valid in terms of the current ontology are accepted by the parsing process using the underlying ontology while building the parse tree.

Given the proposed foundation belief ontology, it seems possible to extend it to any complex beliefs in various practical domains, enabling users to specify the targets to watch on the web as well as explicit matching templates and provide feedback to an agent — so the latter can use artificial general intelligence techniques to evolve the desired behavioral schemata in the course of experiential learning.

The claims above are expected to be practically verified in the further work, with real implementation and testing of the designated agent.

References

1. Looks M., Goertzel B., Pennachin C. Novamente: An integrative architecture for general intelligence. AAAI fall symposium, achieving human-level intelligence, 2004. [Electron. res.]: Available online: `http://www.aaai.org/Papers/Symposia/Fall/2004/FS-04-01/FS04-01-010.pdf`

2. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions // Future Generation Computer Systems. 29. Elsevier. 2013. [Electron. res.]: `http://www.buyya.com/papers/Internet-of-Things-Vision-Future2013.pdf`

3. Goertzel B. Lojban++: An Interlingua for Communication between Humans and AGIs / Proceedings of 6^{th} International Conference, AGI 2013. Beijing. China, August 2013. [Electron. res.]: `http://goertzel.org/agi-13/Lojban_paper_v1.pdf`

4. World Wide Web Consortium: Turtle — Terse RDF Triple Language. November 2012. [Electron. res.]: `http://www.w3.org/TR/2012/WD-turtle-20120710/`

5. JSON-RPC Working Group // JSON-RPC 2.0 Specification. March 2010. [Electron. res.]: `http://www.jsonrpc.org/specification`

6. Kolonin A. Object-Relational Language and modern tradeoffs in software technology / International Andrei *Ershov* Memorial Conference. *PSI 2006.* Program understanding workshop. Russia, Novosibirsk, June 2006. [Electron. res.]: `http://www.webstructor.net/papers/2006ObjectRelationalLanguageAndModernTradeoffsInSoftwareTechnology.pdf`

7. Decker S., Melnik S., Van Harmelen F., etc. The Semantic Web: The roles of XML and RDF // IEEE Internet Computing. N 15 (3). October 2000. P. 63–74. [Electron. res.]: `http://www.cs.vu.nl/~frankh/postscript/IEEE-IC00.pdf`

8. Scowen R.. Extended BNF — A generic base standard / Software Engineering Standards Symposium, 1993. [Electron. res.]: `http://www.cl.cam.ac.uk/~mgk25/iso-14977-paper.pdf`

9. Kolonin A. High-performance automatic categorization and attribution of inventory catalogs / Proceedings of All-Russia conference „Knowledge — Ontology — Theories“ (KONT-2013), October 2013, Novosibirsk, Russia. [Electron. res.]: `http://webstructor.net/papers/Kolonin-HP-ACA-IC-text.pdf`

Kolonin Anton Germanovich — Ph.D.,
Aigents Group, e-mail: akolonin@gmail.com

Дата поступления — 28.04.2015