# Situations in P2P data integration system

**Jerzy Bartoszek**[1], **Grażyna Brzykcy**[1]

**Abstract:** Situations can be used to model context of agents' actions in heterogeneous P2P data integration system. They are the crucial concept in the situation theory and are suitable to cope with data, metadata, information partiality, open-world and non-monotonic reasoning in knowledgebase systems. In the SIXP2P system agents can ask and answer queries, which are propagated to their partners and are evaluated with respect to partners' contexts. These actions are done via Prolog-like goals. Operational semantics of the computations is presented as a set of context-dependent rules. Specification of sample actions is depicted.

**Keywords:** P2P system, agent's actions, situations, context-dependent computations, Prolog-like computations.

## 1. Motivation

In information systems even the same data may be represented in different ways. Data (knowledge) representation depends on many design requirements and decisions, like methods used to solve a problem, a set of concepts used to describe the problem etc. But the form of knowledge representation is mostly application-oriented. So, to build intelligent services enabling access to heterogeneous information sources we adopt the data-oriented approach and focus on the substance of information (its content, meaning) processed by agents. We assume that information content does not depend on data representation.

The content-oriented approach is also a very basic requirement of the Semantic Web (Berners-Lee 2001), a vision of a distributed network of data sources that can be automatically and effectively processed with respect to their semantics. To make the Semantic Web a useful reality, the proper software tools are built that allow data integration, effective searching and querying of information sources and coordination of agents' actions. One can find models and implementations of a middleware for the Semantic Web developed over known standards, such as XML, RDF and OWL (Ciancarini 2002, Fensel 2004, Tolksdorf 2004). The evolution of some proposals (for example Tolksdorf 2006) clearly demonstrates that on the one hand, we strongly need reasoning services, but on the other, that Semantic Web technologies are still immature. Discovered deficiencies, in most cases, are direct consequences of constraints of the Semantic Web standards. These standards have grown increasingly to cope with larger and larger area of problems. As an effect we are situated at the moment of time when armed with different specialized tools we

[1] Institute of Control and Information Engineering, Poznań University of Technology, pl. M. Skłodowskiej-Curie 5, 60-965 Poznań
e-mail: {jerzy.bartoszek,grazyna.brzykcy}@put.poznan.pl

are not able to effectively use them together. Worse, trying to solve other significant problems, like information partiality, open-world and non-monotonic reasoning, we probably need another new standards. To remedy the problem we propose to found semantic processing on the solid and strong enough formal basis and on the tested and known Prolog-like inference mechanisms (Barwise 1983).

As a theoretical background we choose situation theory (Barwise 1997, Devlin 1991), which is regarded as one of the best tools of meaning analyses. In the formalization actions and interactions are analyzed in terms of the creation, acquisition, storage and exchange of information. The situation theory supports partiality, context-dependence and extensive reification and offers a useful abstraction mechanisms. We use situations to model mental states of agents in heterogeneous P2P (Peer-to-Peer) system of semantic data integration.

Prolog-like systems, particularly those with types and extended records ($\psi$-terms, feature structures) (Carpenter 1990, Ait-Kaci 1994), provide not only a high level of abstraction for knowledge representation but also a pattern matching, unification and subsumption – techniques having practical use in the data integration tasks. Moreover, they come with metaprogramming facilities and divers extensions (e.g., *LogicPeer*, *LogicWeb* (Loke 2006), situations (Loke 2004)), a backtracking search and executable specifications. All the characteristics are suitable to tackle the querying and reasoning with semantics.

In this paper we consider software systems, which consist of autonomous components (agents, peers) (Wooldridge 2002) each of which manages some part of data (knowledge). An agent can independently decide how to structure its local data and can cooperate with other agents by asking them and answering queries. Peers decide when to join and leave the system, when to communicate and share their data with other partners. The semantic data integration is one of the main goals of such systems. Due to this goal and, generally, to the dynamism and variability of today's network systems, agents need to know, along with their object data, also different kinds of metadata. They are crucial to efficiently adapt agents' behavior in a distributed open environment. We assume that an agent performs actions with respect to all data and metadata it posses. They form agents' context or mental state (Zambonelli 2004). Rules which describe the cooperation of agents may be expressed formally, too. Following (Loke 2006) we propose a set of rules, which describe how Prolog-like questions are evaluated by agents. These rules describe operational evaluation of questions.

So, the contribution of this paper is as follows:

1. We adopt abstract situations as contexts of agents' actions. Such context can consist of Prolog-like facts and procedures. Facts are used to represent elementary pieces of data and metadata, and procedures represent actions undertaken by the agents.

2. We propose a set of rules, which describe evaluation of these actions with respect to explicitly shown context.

3. We outline a Prolog specification of some actions executed by agents in the

semantic data integration system SIXP2P currently under development in Poznań University of Technology (Brzykcy, Bartoszek, Pankowski 2007).

In the next section main concepts of situation theory are recalled. Section 3 is devoted to evaluations of data integration processes. Selected procedures executed by agents are shown in section 4.

## 2. Main concepts of situation theory

The situation theory is a useful mathematical theory of meaning and information content. In knowledge representation it may serve as a formal foundation of semantic interpretation and reasoning. Situation semantics may be also applied to communicative events, where communication (e.g., via XML documents) is considered to be the effective transfer of meaningful information. Distinctive features of situational approach are the partiality and relevance of information (due to the finite, situated nature of the agent), context-dependence and extensive reification. All these properties make situation theory particularly convenient for modeling of network and communication services.

In fact software developers may have two points of view of situation theory. The first one, called conceptual, uses the notions of situation theory to model knowledge. In the second one, called computational, different contextual inference mechanisms can be adapted to make reasoning.

The basic ontology of situation theory gives a small number of concepts with individuals, relations and situations, which are suitable for world modeling. The most elementary construct is infon – a discrete item of information. If R is an n-place relation and $a_1$, ..., $a_n$ are objects appropriate for the respective argument places of R, then a tuple $<<R, r_1 \rightarrow a_1, ..., r_n \rightarrow a_n, +>>$ denotes the informational item that $a_1, ..., a_n$ are standing in the relation R, and a tuple $<<R, r_1 \rightarrow a_1, ..., r_n \rightarrow a_n, +>>$ denotes the informational item that $a_1, ..., a_n$ are not standing in the relation R. Explicit representation of negative information is the first step towards an open-word reasoning. $r_1$, ..., $r_n$ describe roles of the objects $a_1, ..., a_n$ respectively and they are called the names of arguments of R, whereas the last element is called its polarity and is equal to – or $+$. Infons in themselves are not true or false.

To represent partial information one can omit some arguments of R. The minimality conditions for R specify groups of argument roles of R that need to be filled in order to produce a well-formed (well-defined) infon. If $\sigma 1$ and $\sigma 2$ are two infons with the same relation R, and $\sigma 2$ has at least the same arguments as $\sigma 1$, then $\sigma 1$ subsumes $\sigma 2$ (in $\sigma 1$ there is less information than in $\sigma 2$).

A situation is a limited structured part of the world individuated by a cognitive agent. Situations make certain infons factual. Taking into consideration a situation s and an infon $\sigma$, it is written $s \mid = \sigma$ when a situation s supports an infon $\sigma$ ($\sigma$ is true in s). If I is a finite set of infons and s is a situation we write $s \models I$, if $s \mid = \sigma$ for every infon $\sigma$ in I.

Given a real situation s, the set $\{\sigma \quad \mid s \mid = \sigma\}$ of infons that support the situation is taken to be the corresponding abstract situation (the model of the real situation).

A flexibility of the agent's scheme of individuation – a way of carving the world into uniformities (i.e. types) – is a particularly valuable aspect of the situation theory. The basic types include temporal locations, spatial locations, individuals, relations, situations, infons, parameters and polarity. It is written o : T to indicate that object o is of type T. New types can be defined over some situation s. If p is a parameter and I is a finite set of infons (involving p), then there is a type [p | s |= I] of those objects to which p may be anchored in s, so that all conditions in I are satisfied. Taking into account a situation parameter S and a set I of infons, there may be a corresponding type [S | S |= I] of situations in which the conditions in I obtain.

In the situation theory, the "flow of information" (inferences), is realized via constraints on situation types. A constraint [S1 => S2], where S1 and S2 are situation types, describes dependency between a situation of the type S1 and some situation of the type S2. With the infon <<S1=>S2,+>>, where => is a relation written in the infix style, an agent knows that if it is in a mental state of the type S1 than it is also in a state of the type S2. Generally, any constraint may depend on a set (e.g., the set B) of background conditions under which it will convey information. This is written as [S1 => S2] / B. Constraints with background conditions allow to capture in a satisfactory way phenomenon like non-monotonicity in commonsense reasoning that arises from shift in the context under which reasoning takes place.

Situations together with constraints constitute a context in situation theory (Akman). Using the same concept of situation one can homogenously express data, metadata, inference rules and various aspects of context (ontologies, accessible resources, knowledge shared by the agents, agents' mental states etc.). Context may be easily modified as real situations are modeled by means of sets of infons.

## 3. Context-dependent evaluation of agents' actions

As information is always about some situation it is modeled by abstract situation – a set of infons. In (Erkan 1995) it has been shown that situation can be adopted in Prolog-like systems. We assume that in such system:

- some standard types are predefined,
- new types can be derived from more than one super type, so hierarchies of objects may be quite complex,
- an inference engine is built upon the hierarchy of these types,
- arguments of terms have names and types (compare Ait-Kaci 1994 and Carpenter 1990),
- terms with missing arguments are accepted, so partial information about individuals may be expressed.

Agents (peers) have unique names denoted by p, q, r and so on. Each peer has its own knowledge modeled as a set of infons with Prolog facts and rules (Prolog-like rules are represented as infons, for example, the rule H:-B can be treated as infon <<H:- B, +>>). This set of infons constitutes an abstract situation, which represents the mental state of the peer (or the context) in which the peer executes actions. The mental state of the peer p is denoted by $\underline{p}$. The peer sees a set of another agents, its partners, and may request them to perform actions. Queries issued by the peer initialize these actions. A query has the form of Prolog-like goal passed to some peer (peers). For example, if the peer p wants the goal G to be evaluated by the agent q, it executes q*G i.e. sends the goal G to the agent q. If the evaluation of G succeeds, then some result $\theta$ (a Prolog-like substitution) is returned to p. Following (Loke 2006) we denote this evaluation process by a relation p $|-_\theta$ q*G.

Some goals can be evaluated locally by an agent p itself. If the mental state of p is $\underline{p}$ and the evaluation of the goal G succeeds with the result $\theta$, then we denote it as p/$\underline{p}$ $|-_\theta$ G.

Now, it is necessary to point out that an agent p can "know" some constraints. For example, if its mental state $\underline{p}$ is the situation of type Q and the constraint [Q =>R] is known to an agent, then it can perform its action with respect to some additional situation $\underline{r}$ of type R, i.e. an agent evaluates this action in the new context $\underline{p}\cup\underline{r}$. Note, that we consider contexts as abstract situations (i.e. sets of infons).

In many processes actions have to be executed simultaneously by several agents. Therefore, in our system we assume simultaneous evaluation of Prolog-like goals. Let us once more recall (Loke 2006). We find there two forms of parallel goals: $q_1,q_2,...,q_n$ [] G and $q_1,q_2,...,q_n$ <> G. The goal G is sent to all the agents $q_1,q_2,...,q_n$. In the first case the goal G must be executed with success by all the peers from the list $q_1,q_2,...,q_n$. In the second case, the goal G must be executed with success by at least one peer in the list $q_1,q_2,...,q_n$. If more then one result is returned then all the answers need to be composed. If the results are inconsistent, the overall goal fails.

The evaluation of goals can be described in operational manner by rules of the form $\frac{premises}{conclusion}$, where conclusions hold when all premises hold. In rules depicted below we do not show the obvious mental states of agents.

$$\frac{}{p/\underline{p}| -_{\in} true} \tag{1}$$

The above rule expresses the fact that any agent p with mental state $\underline{p}$ always can execute the empty goal denoted by *true*.

$$\frac{\underline{p}| =<< R(t_1,\ldots,t_n)\theta, + >>}{p/\underline{p}| -_\theta R(t_1,\ldots,t_n)} \tag{2}$$

If the agent p "knows" the fact <<R($t_1$,...,$t_n$)$\theta$, +>>, than it can execute (with a success) a goal R($t_1$,...,$t_n$).

$$\frac{\underline{p}| =<< R(t_1,\ldots,t_n)\theta, - >>}{p/\underline{p}| -_\theta \neg R(t_1,\ldots,t_n)} \tag{3}$$

If the agent p “knows” the fact <<R($t_1$,...,$t_n$)$\theta$, −>>, then it can execute (successfully) a goal ¬R($t_1$,...,$t_n$). Due to rules (2) and (3) the agent p can correctly act in the “open world”. Note, that at this point we eliminate assumption about the “closed world” made in Prolog reasoning engine.

$$\frac{\gamma = mgu\,(A, H) \wedge << H : -G, + >> \in \underline{p} \wedge p/\underline{p}| -_{\delta}\, G\gamma}{p/\underline{p}| -_{\gamma\delta}\, A} \tag{4}$$

The above rule shows how the agent p executes the goal A when it has the Prolog rule H:-G in his mental state. The assignment $\gamma$ is the most general unifier of A and H. If it does not exist, execution of A fails.

$$\frac{q/\underline{q}| -_{\theta}\, G}{p/\underline{p}| -_{\theta}\, q * G} \tag{5}$$

Rules (1) – (4) define execution of local goals i.e. goal executed without any help of agent’s partners. The rule (5) shows what will happen when the agent p sends the goal G to the agent q. The goal is executed by the agent q with respect to its mental state.

$$\frac{p/\underline{p}| -_{\theta}\, G_1 \wedge p/\underline{p}| -_{\gamma}\, G_2\theta}{p/\underline{p}| -_{\theta\gamma}\, G_1, G_2} \tag{6}$$

In the rule the composition of two local goals $G_1$ and $G_2$ is evaluated. After the execution of the goal $G_1$the new substitution $\theta$ appears. It is used then in the execution of the goal $G_2$.

$$\frac{p/\underline{p} : Q \wedge [p/\underline{p} : Q => p/\underline{r} : R] \wedge p/\underline{p} \cup \underline{r}| -_{\theta}\, G}{p/\underline{p}| -_{\theta}\, G} \tag{7}$$

The rule (7) shows the execution of the goal G by the agent p with respect to the constraint [Q =>R]. The goal is evaluated in the new mental state $\underline{p}\cup\underline{r}$.

$$\frac{q1/\underline{q1}| -_{\theta 1}\, G \wedge \ldots \wedge qn/\underline{qn}| -_{\theta n}\, G}{p/\underline{p}| -_{\theta 1 \ldots \theta n}\, q1, \ldots, qn[]G} \tag{8}$$

$$\frac{q1/\underline{q1}| -_{\theta 1}\, G \vee \ldots \vee qn/\underline{qn}| -_{\theta n}\, G}{p/\underline{p}| -_{\theta 1 \ldots \theta n}\, q1, \ldots, qn <> G} \tag{9}$$

Rules (8) and (9) are used in parallel evaluation of the goal G. In the rule (8) the goal G must be executed with success by all the agents $q_1$,$q_2$,...,$q_n$. In the rule (9) the goal G must be executed successfully by at least one agent $q_1$,$q_2$,...,$q_n$. In both cases if more then one result is returned then all the answers are composed. If the results are inconsistent, the overall goal fails.

## 4. Specification of sample actions

We assume that the system SIXP2P for semantic data integration, currently under development in Poznań University of Technology, consists of autonomous agents, each of which can independently decide how to structure its local data. Local schemas that reflect possibly heterogeneous semantic models that are developed by different peers describe the local data.

An agent in the SIXP2P system sees a set of another agents, its partners, and may ask queries only to them. However, a query may be propagated to partners of each peer inducing a significant extension of the set of possible "knowledge sources". So, agents indirectly connected to the inquirer cooperatively evaluate the query.

There are two types of the agents in the system: peers and brokers. A context of each peer consists of its partners (with their schemas and data) and a broker, whereas the broker sees all the peers (their identifiers). Each peer can perform three actions directed to a broker, namely introduction (introduce/3), exit (log out/2) and schema modification (modify schema/1).

Introduction of the Agent to the Broker consists of registration, which is an action executed by the Broker (a broker identifier prefixes the goal register(Agent, Parts)), and results in replying a list of Agent's partners. The Agent stores the list as a part of its context.

```
introduce(Agent, Broker, Parts) :-
  Broker * register(Agent, Parts),  % registration is done by a broker
  assert(partners(Parts)).          % list of partners is stored
```

Note that metapredicate assert/1 is used to change the current mental state (context) of the agent. Similarly, the action of leaving the system by the Agent contains a message to the Broker. The broker conveys than this information (logged_out/1) to all the agents, which previously have cooperated with the Agent.

```
log_out(Agent):-
  agents(As),
  a_remove(Agent, As, As1),         % the logged out agent is removed
  retract(agents(_)), assert(agents(As1)),
  set_of(A, (partners(A, Parts),    % agents, which cooperate with
    member(Agent, Parts)), As2),    % the removed agent are selected
  l_inform_all(As2, Agent),         % and informed
  retract(partners(Agent, _).

  l_inform_all([ ], _).
  l_inform_all([A | As], Agent):-
  A * logged_out(Agent),
  l_inform_all(As, Agent).
```

If the agent's schema has been changed, the broker sends an appropriate message (modified_schema/1) to all the agents that may use this schema.

```
modify_schema(Agent):-
  set_of(A, partners(A, Parts),     % agents which may use changed
  member(Agent, Parts)), As),       % schema are selected and informed
  s_inform(As, Agent).
  s_inform_all([], _).

  s_inform_all([A | As], Agent):-
  A * modified_schema(Agent),
  s_inform_all(As, Agent).
```

To gain information from any partner an agent has to prepare suitable mappings (create_map/3) between schemas. A mapping from the Partner to the Agent is denoted as Mpa and from the Agent to the Partner as Map (the special value null is chosen to depict the situation when such mappings are not constructed).

```
create_map(Part, Mpa, Map) :-
  schema(self, Scha),              % agent's schema
  Part * schema(self, Schp),       % schema is taken from the partner
  map(Scha, Schp, Mpa, Map),       % mappings from and to the partner
  assert(mappings(Part, Mpa, Map)). % mappings are stored in context
```

A complex process of querying and answering is specified as an action (ask/3). It consists of local query processing (query/2), asking of the partners (ask_partners/2) and merging of answers (merge/3).

```
ask(Agent, Query, Answer) :-
  query(Query, Ansl),              % local query is answered
  ask_partners(Query, Ansr),       % partners' queries are answered
  merge([Ansl], Ansr, Answer).     % answers are merged
```

Only qualified partners, for whom both mappings are constructed, are inquired.

```
ask_partners(Query, Ans):-
  choose(QParts),                  % qualified partners are chosen
  ask_qparts(Query, QParts, Ansr),
  merge([ ], Ansr, Ans).           % answers from partners are merged

  choose(QParts):-
  set_of(Part, (mappings(Part, Mpa, Map),
  Mpa <> null, Map <> null), Qparts).
```

To ask a partner an agent has to convert (q_reformulate/3) the original Query to the appropriate form Qp, directed to the suitable partner P. Similarly, answers are reformulated (a_reformulate/3) by the inverse mapping.

```
ask_qparts(_, [ ], _).             % all partners are asked
  ask_qparts(Query, [P | Ps], [A |As]) :-
  mappings(P, Mpa, Map),
  q_reformulate(Qp, Map, Qp1),     % query is transformed
  P * ask(Qp1, Ap),                % query is answered by a partner
  a_reformulate(Ap, Mpa, A),       % answer is transformed
  ask_qparts(Query, Ps, As).
```

## 5. Final remarks

In this paper we consider some problems related to context-aware reasoning in P2P data integration systems. In our approach we do not use standards of Sematic Web, but try to show how to take advantage of a strong formal basis, the situation theory, and known reasoning mechanisms to remedy deficiencies of today's technologies. It is worth noticing that XML data can be expressed in extended Prolog systems (Brzykcy, Bartoszek, Pankowski 2007). Equipped with reasoning tools over partial data and metaprogramming facilities these systems are also particularly suitable to tackle the semantic-oriented processing.

## References

Ait-Kaci, H. et all. (1994) *The Wild LIFE Handbook*, Paris Research Laboratory.

Akman, V., Surav,M. The Use of Situation Theory in Context Modeling. *Computational Intelligence* 12 (4).

Brzykcy, G., Bartoszek, J. (2007) : Context in Rules Used in P2P Semantic Data Integration System, In: *Proceedings of RR'2007*, LNAI 4424,pp.377-380.

Brzykcy, G., Bartoszek, J., Pankowski, T. (2007) Semantic data integration in P2P environment using schema mappings and agent technology. In: *Proceedings of AMSTA 2007*, LNAI 4496, pp. 385-394.

Barwise, J., Perry, J. (1983) *Situations and attitudes.* MIT.

Barwise, J., Seligman, J. (1997) Information Flow. The Logic of Distributed Systems. Cambridge Universtity Press.

Berners-Lee, T., Hendler, J., Lassila, O. (2001) The Semantic Web. *Scientific American* 284:34-43.

Carpenter, B. (1990) The logic of typed feature structures: inheritance, (in)equations and extensionality. Second European Summer School in Language, *Logic and Information.* Leuven.

Ciancarini, P., Tolksdorf, R., Zambonelli, F. (2003) A Survey on Coordination Middleware for XML-Centric Applications, In: *Proceedings of ACM SAC 2002*, p. 335-343.

Devlin, K. (1991) *Logic and information.* Cambridge University Press.

Erkan, T., Akman, V. (1995) Situations and Computation: An Overview of Recent Reseach. In: Griffith J., Hinrichs E.W., Nakazawa T. (eds) *Proceedings of the Topics in Constarint Grammar Formalism for Computational Linguistics*, Copenhagen.

Fensel, D. (2004) Triple Space Computing. *Technical Report.* Digital Enterprise Research Institute (DERI).

Loke, S. W. (2006) Declarative programming of integrated peer-to-peer and Web based systems: the case of Prolog., *Journal of Systems and Software*, 79(4), p. 523-536.

Loke, S. W. (2004) Representing and Reasoning with Situations for Context-Aware Pervasive Computing: a Logic Programming Perspective. *The Knowledge Engineering Review.*, Vol. 19, No. 3, pp. 213-233.

Manzalini, A., Zambonelli, F. (2006) Towards Autonomic and Situation-Aware Communication Services: the CASCADAS Vision. *IEEE Workshop on Distrubuted Intelligent Systems*, Prague.

Tolksdorf, R., Nixon, L., Liebsch, F., Nguyen, D., Paslaru Bontas, E. (2004) Semantic Web Spaces. *Technical report B-04-11*, Freie Universitat Berlin.

Tolksdorf, R., Paslaru Bontas, E., Nixon, L.: (2006) A coordination model for the Semantic Web. *SAC'06*, Dijon.

Wooldridge, M.: (2002) *An Introduction to Multiagent Systems.* John Wiley & Sons.

Zambonelli, F., Van Dyke Parunak, H. (2004) Towards a Paradigm Change in Computer Science and Software Engineering: A Synthesis. *The Knowledge Engineering Review.*