The Ontology Lifecycle in RoboCup: Population from Text and Execution

Stephan Gspandl, Andreas Hechenblaickner, Michael Reip, Gerald Steinbauer, Máté Wolfram, and Christoph Zehentner*

Institute for Software Technology, Graz University of Technology, Graz, Austria
kickofftug@ist.tugraz.at

Abstract. In RoboCup it is important to build up domain knowledge for decision-making. Unfortunately, this is a time-consuming and laborious job. At championships easy adaptability of this domain knowledge can be especially crucial as teams need to be able to change tactics and adjust to opponent behavior as fast as possible. An intuitive interface to the agent is therefore necessary.

In this paper, we present a methodology to automatically populate a domain ontology from natural language text. The resulting populated ontology can then be deployed in a multi-agent system. This automatic transformation of text to knowledge for decision-making thus provides such an intuitive interface to the agents. It is embedded into the broader (up to now) theoretical context of an ontology lifecycle.

We have created a proof-of-concept implementation in the 2D RoboCup Simulation League on the base of tactics descriptions from soccer literature. Experiments show that 71% of tactics are perfectly transformed and 86% of the actions are executed correctly in terms of geometric relations.

1 Introduction

In every RoboCup league domain knowledge is necessary for the agents or robots to perform intelligent tasks. Many teams use expert literature (for example from books like [14] and [7], or from the HowTo-Wiki[1]) for the purpose of transforming information how to do things into specific models, plans or source code. Unfortunately, this is a very time-consuming and laborious job which yields many hybrid and incompatible descriptions of the same concepts. Even worse, as a rule or knowledge base for robots is generally rather large, its maintenance is very demanding, too. It commonly takes at least one person to put constant effort into its manual definition and update. From this point of view we strongly argue for the automation of the knowledge generation, its incorporation into the team as well as the maintenance of this knowledge base.

* The work has been partly funded by the Austrian Science Fund (FWF) by grant P22690.

[1] http://www.wikihow.com

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 389–401, 2012.

Ontologies store data in a structured way under a semantic context and thus provide a suitable basis for knowledge-based agents to operate on. In order to make the definition and adaption of behavior as simple, fast and intuitive as possible, an ontology-based agent system should be able to

- populate a domain ontology based on various sources of information,
- link the ontology classes to actions and
- constantly improve, enrich and verify the ontology during execution.

Thus, descriptions of soccer strategies from books, websites or other knowledge artefacts should be automatically transformed into an ontology the agents can execute. This way, new tactical instructions can be easily incorporated into the agents' knowledge base at championships to exploit specific weaknesses of other teams. Another example would be a logistics robot which has to be trained to deliver parts from the warehouse to a newly installed assembly line. In order to increase the factory's flexibility (and lower costs) a simple way of explaining the desired behavior to the robot in natural language is to be prefered over hardcoding it.

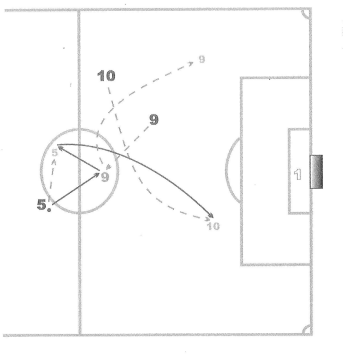

Fig. 1. A sketch accompanying the instructions from [7] to perform a penetration into the opponent field

Therefore, we are constantly working on a methodology for an easy translation of arbitrary, even dynamic (i.e. that also conveys temporal relations) information into behavioral descriptions. Take the following tactics description in natural language from [7] (which is accompanied by the sketch shown in Fig. 1); this will also serve as running example throughout this paper:

```
* 5 passes the ball to 9 (who has come towards him) and moves
  diagonally to get the return pass;
* 9 passes the ball back to 5;
* 10 sprints deep to the right, criss-crossing with 9 who does the
  same thing from the opposite side;
* 5 can choose to pass the ball to either 9 or 10.
```

It is easy for humans to understand these sentences and create a mental picture of the described situation. Looking at the according sketch enables them to capture the relations even faster and yields additional information how to interpret it correctly. This is mainly due to the fact, that people are acquainted with soccer and language, so the words can be easily grounded in the soccer domain and (commonly) immediately make sense in the way they have been put together.

The ideas described in this paper follow the same intuition when applied to RoboCup: tactics descriptions should be processed and grounded in the RoboCup soccer domain using a domain ontology and Natural Language Processing (NLP) techniques. The resulting populated ontology, containing the grounded instructions in semantic relation, should then be employed as knowledge base for the agents in the simulations, just like a human remembers during the game what her trainer taught her. Resembling human intuition as well, this ontology should then be verified and improved during gameplay. If a human player notices, for example, that an opposing team adjusts to her tactics, she will try to counter by improving her gameplay or asking her trainer for advice. Similarly an agent could try to vary its behaviour or learn new tactics or variations of the original to win.

The efforts in this paper cover the population of a soccer ontology from text and executing the ontology tactics in the 2D RoboCup Simulation League in the more general scope of an ontology lifecycle. We define this lifecycle as the creation, population and management of an ontology, where the latter consists of the continous evaluation, enrichment and improvement of the populated ontology in the environment. The remainder of this paper is therefore organized as follows: related research concerning the ontology lifecycle is discussed in Section 2. The methodology for automatic population and execution of a domain ontology from text is described in Section 3 in the context of this ontology lifecycle. The experiments and results are covered in Section 4 followed by conclusion and outlook in Section 5.

2 Related Research

Ontologies in software engineering are employed to structure data and store it in a semantic context (accessible to reasoning), or just to conceptualize a certain domain. They define a database of concepts, relationsships and other representational entities for the purpose of performing reasoning, presenting it to and preparing it for users, or to provide a knowledge base for software (knowledge agents).[2]

Naturally, there has been effort to employ ontologies in agent and also multi-agent systems (for example [16], and [19]). For our purposes, an agent is defined as an entity, receiving input from its surroundings and trying to find the best-possible action to perform in the environment on base of a performance measure. In a multi-agent system two or more of such agents have to work together to pursue a mutual goal. It is based on problem-decomposition, trying to reduce the

[2] see `http://tomgruber.org/writing/ontology-definition-2007.htm`

complexity of the given problem by employing many agents, often specialized to certain tasks [24]. The benefits of ontologies (to structure information, to make it accessible to reasoning and to allow for restructuring and presenting the knowledge base in an intuitive format) make them very suitable to control such knowledge-based agents.

Commonly, in most knowledge engineering applications, the users (or knowledge engineers) have to encode the required knowledge or employ large and bulky ontologies not specifically designed for the special domain of application. Thus, one has to accept many drawbacks. In order to keep the knowledge engineering as intuitive, simple, thus minimising time consumption, multiple concepts have to be combined into an integrated process.

A system as proposed which is able to support the complete knowledge engineering process, has to deal with various fields of ontology research: (1) automatic or semi-automatic (i.e. unsupervised or supervised) creation and management of a domain ontology, (2) population of ontologies from various (information or knowledge) sources, (3) grounding the ontology knowledge in a specific agent language, (4) adopting the populated ontology for deployment in the specific (multi-)agent system, as well as (5) refinement, improvement, enrichment and verification of the ontology (further called management of the deployed ontology) in the environment.

A domain ontology can be learned or induced from various sources, most commonly text and websites, but there are also many prepared ontologies for arbitrary topics to start off with. Often called the holy grail of NLP is the extraction of knowledge from completely unstructured text without supervision [23]. The result would be a domain ontology specific to the given problem and certainly well-suited for solving it. Unfortunately, this approach cannot yet provide adequate accuracy. Therefore, in most cases the user has to rely on at least semi-structured or dedicated input or multiple data sets, which have to be maintained separately. An example of a system for automatic learning of domain ontologies is OntoLearn which extracts them from websites and shared documents [20]. Common, though, are systems that supervise and support knowledge engineers in the (cooperative) ontology construction or learning. For example, multiple concepts, tools, data sources and algorithms are incorporated into an integrated system [15].

The main focus of this work comprises the automatic population of ontologies from text. There has been a lot of research into this topic, from semi-automatic to unsupervised concepts. For example, LexOnto provides a way of mapping information to a domain ontology with the help of a lexical ontology [5]. Other researchers use linguistic rules as well as machine learning techniques to populate ontologies and extract information [17]. A semi-automatic system is described in [1] which pre-processes the input texts to generate a conceptual tree and further depends on rules to capture knowledge from it. Focussing on processing the input text for mapping between text and a structured world state, the authors of [13] propose a generative model segmenting text into utterances and mapping these semantically.

The task of deploying the populated ontology in an agent environment is closely related to that of population. This is primarily because an important task in both concepts is grounding. In [12] action ontologies provide natural language interfaces to agents. This paper deals with the problem of representing special action concepts and mapping natural language to suitable actions. Such specific grounding of text onto action concepts is also the topic of [3] and [2], both employing learning techniques to semantically connect commentaries to soccer games and instructions to actions respectively. In the context of grounding and interpreting, it is necessary to identify action structures and intents in the given input texts. [26], and [25] deal with identifying intentions in web search and natural language.

As the generated ontologies will always contains errors and thus have to be continously maintained, management of ontologies is an important issue. Although the same concepts and ideas as shown above can be employed to iteratively improve and refine the ontologies, the feedback of the environment the agents operate in is valuable for this task. Therefore, this should also be captured in a complete knowledge engineering process for (multi-)agent systems. [22] and [6] give an overview over the state of the art in ontology population and enrichment, whereas [28] mainly employs learning techniques for this purpose. Concerning verification [11] computes justifications to explain inconsistencies in ontologies.

The experiments described in this paper are based on the 2D RoboCup Simulation League and a team which has participated in several RoboCup world championships. Detailed information on this league can be found in [4], the according changelogs, on the official website[3] and the championship website[4].

3 The Ontology Lifecycle

Related research yielded an integrated process (shown in Fig. 2) dealing with all necessary steps to automatically generate a self-optimizing RoboCup Simulation League team from soccer knowledge. We call this the Ontology Lifecycle. Before applying the process, relevant knowledge sources have to be identified and evaluated. The books on soccer tactics mentioned in the introduction have provided a good starting point for our purposes. The ideas proposed, though, go even beyond textual input. Proper transformations could also be defined to make use of sketches [9], inspired by research of the MIT Artificial Intelligence Laboratory [10] or the Qualitative Reasoning Group at Northwestern University [8].

In the first stage of the ontology lifecycle, these knowledge artifacts are preprocessed to provide more structured data for the remaining steps. This data will then be interpreted and grounded onto a soccer domain ontology, i.e. the ontology is populated with concrete concepts derived from the data. The domain

[3] see `http://sourceforge.net/apps/mediawiki/sserver/index.php?title=Main_Page`
[4] see `http://julia.ist.tugraz.at/robocup2010`

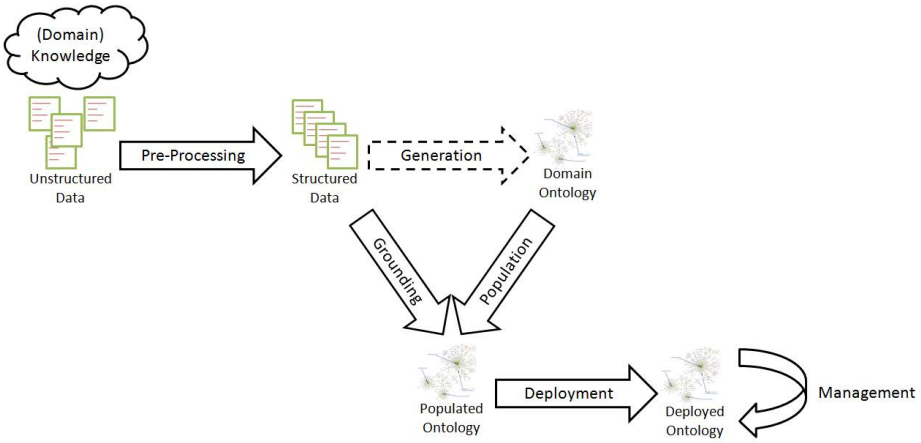

Fig. 2. An integrated process for automatic transformation of knowledge and its deployment in multi-agent systems

ontology may be generated out of suitable data, but is created manually for our tests.

The resulting populated ontology is then deployed in the RoboCup Simulation League to build a virtual soccer team. The process is implemented up to this point and will be concluded by concepts to verify and improve this basic ontology in simulations in future research. In the following sections the ontology population and deployment of a RoboCup team based on the descriptions of [7] (see the running example given above in Section 1) are described in detail.

3.1 Populating a RoboCup Ontology from Text

The automatic population proposed is based on natural language descriptions as found in [7] and a domain ontology. The domain ontology contains general soccer concepts relevant for games in the RoboCup Soccer Simulation League as well as more specific entities for the language processing: this includes players (identified by their uniform number) and player types (e.g. goalkeeper), components of the field (e.g. wing, goal etc.), positions (e.g. center) and the tactic model. For a team to perform a tactic in a game, this model comprises a pre-condition (under which the tactic should be executed), a set of actors (participating in the tactic) and a compound of actions each player has to perform, where actions can be executed in sequence or in parallel.

Pre-processing. Before a description can be actually populated on the base of and into this ontology, it has to be pre-processed to reduce the complexity of correctly grounding the text in the semantics of the ontology. This pre-processing comprises several subtasks, combining state-of-the-art NLP techniques: First, the text is split into atomic actions by a set of stop-words (like 'and') and stop-chars

(like ';'). In order to allow for correct identification of terms, the Stanford POS (part-of-speech) tagger [27] is used to assign the lexical category to each word. Now, irrelevant words or irrelevant categories can be removed. On the same basis, pronouns are resolved and the atomic sentence stubs are transformed into a tuple $\{S, P, O\}$ (simply referred to as SPO in the remainder of the paper), with S being the subject, P the predicate and O the object of the sentence. This is done by identifying the verb by POS tag, checking for an actor in the first part and retrieving the object from the rest. After subject and object have been identified, pronouns can then be exchanged by their corresponding entity. For example, after the transformation of the first atomic sentence stub in the running example ('5 passes the ball to 9') into the SPO $\{5, pass, 9\}$, the pronouns 'who' and 'him' in the second ('who has come towards him') are substituted by '9' and '5', respectively.

As the descriptions of the tactics are not linear in time and involve parallel activities as well, temporal adjustments are necessary. By evaluating the tenses and identifying jumps as well as using keywords (for example 'while') in the original sentence stubs, a timestamp can be assigned to each SPO. If an SPO is associated with a timestamp m, it is to be executed before an SPO with timestamp n when $m<n$. The first action(s) have timestamp *0*. The second part in the example above would thus be assigned timestamp *0* and the first timestamp *1*. Naturally, same timestamps convey that actions have to be performed at the same time.

In the last step of the pre-processing, synonyms are resolved so that different words or phrases with the same meaning can be grounded onto the same semantic concept in the domain ontology. This is done using WordNet [18]: Each word of each SPO is looked-up in the WordNet dictionary (the POS type is transformed into the corresponding SynsetType), reduced to a base form and stored in a synonym dictionary. As in most cases multiple definitions exist for one word, there are two options: either all possible synonyms are saved or the correct is identified by the user (supervised resolution). With a small vocabulary, the first variant is simple and accurate enough, but with increasing size the latter should be preferred as the range of synonym terms might become too large.

Actual Population. Having the SPOs in correct temporal sequence and mapped to unique base forms, the actual population of the ontology can begin. For this purpose, test sequences are generated on the base of the ontology's taxonomy (this concept is easily extended to follow the restrictions as well; please note that formal reasoning will be a topic of future research). That is, the root classes corresponding to each element in the SPO tupel are retrieved from the ontology and specialized test strings are concatenated from simple identifiers (names or specializing qualifiers) associated with each class (as part of the domain modeling) while visiting the subclass relations. For example, the soccer domain ontology contains the following path through the taxonomy tree: *Object - GameEntity - Wing - Left Wing* (with A - B describing that B is a subclass of A). For this simple path, the test strings 'wing' and 'left wing' are created.

This way, each test string is automatically associated with the correct ontology class. The test sequences are then generated as all possible combinations of the subject, predicate and object test strings, yielding for example '5 run left wing'.

Each SPO from the pre-processing can now be compared to each test sequence. As equality measure serves an adapted Levenshtein distance on word level. It derives a similarity factor based on the sequence of words on the one hand as well as the number of equal and the number of words not in both strings. As the synonym resolution is also applied to each word of the test strings, there is no need to compare individual words by characters. The test sequence with the highest equality is saved as target concept.

The SPOs are then traversed to create an action compound for the new tactic in the ontology: for each action a new subclass is derived and the restrictions to subject and object class are set. These classes are put in temporal context by either creating parallel actions or an action sequence depending on the timestamp. The result is the action compound which is associated with the tactic. In order to complete the tactic, it is connected to the set of participating actors and a pre-condition. The latter has to be defined by the user as it is not conveyed in the text.

Post-processing. As the descriptions may contain implicit semantics, the populated ontology can not yet be employed for execution. A set of simple rules is provided to post-process it and fill certain semantic gaps. These rules contain instructions on how to add or adapt ontology actions. After application the ontology is ready for execution.

An example for the need to post-process the populated ontology would be the criss-crossing action in the running example. As actions are generally directed, the criss-crossing of two players is an exception. The populated ontology, thus, contains only one action that player 10 criss-crosses with player 9. In order to resolve this issue, a post-processing rule states that criss-crossing is bi-directed and a criss-crossing action in the opposite direction has to be inserted parallel to the existing.

3.2 Executing the Populated Ontology

After successfully transforming the tactics descriptions into a populated ontology, execution is rather straightforward. Before it can be employed, though, it has to be embedded properly into this execution environment. In order to do so, semantics for execution have to be defined.

Preparing the Populated Ontology. The task of preparing the populated ontology is to provide semantics for the execution. It is thus necessary to define a mapping from ontology action to the method to execute as well as semantics to synchronize the activities correctly. Both have to be defined for the concrete execution environment, in our example a RoboCup simulation league team.

Consider the first two sentences of the running example again: it tells the reader that player 9 runs towards player 5 and then 5 passes the ball to 9. This sounds easy, but two questions arise when it comes to execution. These are (1) what method the agent should call when executing the ontology action associated with 'come towards' (from now on this relation will be represented by angle brackets), and (2) when exactly player 5 should pass the ball. In order to answer these questions we have to (1) define that the ontology action ⟨come towards⟩ maps onto the method *runTo(Position targetPos)* and (2) provide a rule concretizing the behavior of this action, for example that *player1* ⟨come towards⟩ *player2* makes a *player1* move to the point halfways between her and another *player2*.

On the base of these semantics, a condition tupel $\{Pre, Inv, Post\}$ (following the notion of STRIPS [21]) has to be defined for each action of the compound, with Pre being the pre-condition, Inv the invariant and $Post$ the post-condition. The pre-condition adopts the semantics of the action defined above and in a concrete situation thus grounds how the action has to be executed. In the example above, the pre-condition for ⟨come towards⟩ would therefore define the position player 9 runs to (the point halfways between player 9 and player 5) so that the action ⟨runTo⟩ is executed with this position. The invariant is a general model of an actions correct behavior, i.e. the invariant of ⟨come towards⟩ is true as long as player 9 is running towards the described position. Last, the post-condition describes when the action is successfully executed, i.e. that player 9 arrived at this position.

This way, synchronization can be created very naturally: actions committed in sequence or in parallel with other activities have to respect these semantics. This means that actions cannot be executed (stated by pre-condition) as long as actions which have to be performed earlier in the sequence, have been finished (stated by post-condition). Although this is true for most cases, yet experiments have shown that this constraint is too hard in some situations. Again consider our example: if actions would have to be performed strictly consequently, player 5 may pass only when player 9 has already arrived at her destination. Valuable time would be lost with players waiting for other players to finish running or for the ball arriving. Thus, running may be performed semi-consequently: An additional rule is added stating that actions may start after a player has already started running.

The positions the actions arrive at, serve as synchronization points that are expressed in the $\{Pre, Inv, Post\}$ tupels. These positions are progressed on base of the action compound hierarchy using the action semantics. The initial positions a tactic is grounded on, suffice to calculate all further positions. In the example, player 5 knows where player 9 is running towards and can direct her pass to this position. As her next action (please revisit the example) is to move diagonally to intercept, this position can be derived for player 9 (and of course would be for all other players).

Defining these semantics might sound complicated but is not: First, the semantics necessary are rather simple and straight-forward. Second, they can be

defined generally for each ontology action (and don't have to be for each appearance in text) and in most cases the definitions are even reusable. Last, in order to perform actions it is necessary for a team to have these ready. It can thus fall back on implementations already available.

Execution. After proper preparation the actual execution is rather simple. Whenever a tactic should be performed (checked by its pre-condition), each participating player (grounded by the tactic's pre-condition) retrieves a linear plan containing only what she has to do. She then executes this plan step-by-step: She starts an action when the pre-condition allows it. The success of the tactic is monitored in each timestep by invariant. That is, each player checks whether all other participating players are still committed to the mutual plan. The invariant is thus a simple conjunction of the individual invariants of all parallel activities. The action the player is performing is executed as long as the post-condition is not fulfilled. If so, this process is repeated with the next action in the linear plan.

4 Experiments

In order to evaluate the concepts developed in this work, we have conducted two series of experiments. On the one hand the quality of the transformation process is measured. 16 tactics descriptions (containing a total of 129 actions) have been chosen randomly and populated into the prepared domain ontology. The ontology tactics are then compared to the original descriptions. The results are that

1. 71% of all actions were identified perfectly
2. an additional 20% of all actions were identified well enough (that is with few flaws which could be corrected by simple rules at execution)
3. the temporal sequence was adjusted correctly in all cases

The resulting populated ontology was therefore well-suited for execution. Most errors left can be easily explained by two factors: on the one hand flaws in POS tagging prevent the correct identification of all SPOs. On the other hand, complex reflexive concepts cannot be interpreted as SPO and require a more complex post-processing. An example would be the phrase 'who does the same thing from the opposite side' from the running example.

Post-processing and execution of the populated ontology were evaluated in simulation. Mapping and semantics as described in the previous setting were created. The quantification in the environment was turned off, so the agent's perception could not influence the simulation results (thus, a single run was sufficient to evaluate each tactic; please note that this was the only aspect that differed from the real championship settings). The remaining errors of the transformation were corrected manually. Then, each tactic was tested in the following setup: a trainer agent positions the number of agents on the field as depicted

in the sketch accompanying the description. The agents were provided with the ontology tactic to execute and the simulation was started and logged until all agents had finished their execution.

Each simulation was evaluated whether the geometric relations were as defined by the original tactic description. The results were satisfying as well, yielding that 86% of the tactics were executed correctly.

5 Conclusion and Outlook

We presented a methodology for an automatic population of a domain ontology from text and its deployment as knowledge base for decision-making in multi-agent systems. In order to demonstrate the benefits we have created a proof-of-concept implementation. The results of our experiments in the 2D RoboCup Simulation League show that (1) 71% of actions are perfectly populated into the domain ontology and (2) 86% of actions are executed correctly.

In the future we will prepare the ontology for deployment in tournaments by creating suitable pre-conditions for each tactic. This way, the tactics will be used to enhance the actual gameplay of a team. Extending the methodology by additional sources we will build a team which is solely based on a populated ontology for decision-making.

Furthermore, we will complete the ontology lifecycle in RoboCup: verification and improvement will be based on simulations as well. The defined semantics will be used as model to monitor the correct execution of tactics in the environment. If an interruption is detected, the reason is inferred from this model and the tactic adjusted.

References

1. Amardeilh, F., Laublet, P., Minel, J.L.: Document annotation and ontology population from linguistic extractions. In: Proceedings of the 3rd International Conference on Knowledge Capture (2005)
2. Branavan, S., Chen, H., Zettlemoyer, L., Barzilay, R.: Reinforcement learning for mapping instructions to actions. In: Proce. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th International Joint Conf. on Natural Language Processing of the AFNLP, Singapore, pp. 82–90 (2009)
3. Chen, D.L., Mooney, R.J.: Learning to sportscast: A test of grounded language acquisition. In: Proc. of the 25th International Conf. on Machine Learning, Finnland (2008)
4. Chen, M., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummenje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., Yin, X.: RoboCup Soccer Server. In: The RoboCup Federation (Juli 2002)
5. Cimiano, P., Haase, P., Herold, M., Mantel, M., Buitelaar, P.: Lexonto: A model for ontology lexicons for ontology-based nlp (2007)
6. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications, 1st edn. Springer (October 2006)

7. Fascetti, E., Scaia, R.: Soccer Attacking Schemes and Training Execrcises. Reedswain Publishing (1998)
8. Forbus, K.D., Usher, J.: Sketching for knowledge capture: a progress report. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, pp. 71–77. ACM, New York (2002)
9. Gspandl, S., Reip, M., Steinbauer, G., Wotawa, F.: From sketch to plan. In: 24th International Workshop on Qualitative Reasoning (2010)
10. Hammond, T., Davis, R.: LADDER: A language to describe drawing, display, and editing in sketch recognition. In: Proceedings of the 2003 Internaltional Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 416–467 (2003)
11. Horridge, M., Parsia, B., Sattler, U.: Explaining Inconsistencies in OWL Ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 124–137. Springer, Heidelberg (2009)
12. Kemke, C.: An action ontology framework for natural language interfaces to agent systems (2007)
13. Liang, P., Jordan, M., Klein, D.: Learning semantic correspondences with less supervision. In: Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th International Joint Conf. on Natural Language Processing of the AFNLP, pp. 91–99. Association for Computational Linguistics, Singapore (2009)
14. Lucchesi, M.: Coaching the 3-4-1-2 and 4-2-3-1. Reedswain Publishing (2002)
15. Maedche, A., Staab, S.: Learning ontologies for the semantic web. In: Semantic Web (2001)
16. Mathieu, P., Routier, J.C., Secq, Y.: Towards a Pragmatic Use of Ontologies in Multi-Agent Platforms. In: Palade, V., Howlett, R.J., Jain, L. (eds.) KES 2003. LNCS (LNAI), vol. 2773, pp. 1395–1402. Springer, Heidelberg (2003)
17. Maynard, D., Li, Y., Peters, W.: Nlp techniques for term extraction and ontology population. In: Proceeding of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge (2008)
18. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An on-line lexical database (1993)
19. Mousavi, A., Nordin, M.J., Othman, Z.A.: An ontology driven, procedural reasoning system-like agent model, for multi-agent based mobile workforce brokering systems. Journal of Computer Science 6, 557–565 (2010)
20. Navigli, R., Velardi, P.: Learning domain ontologies from document warehouses and dedicated web sites. Computational Linguistics 30 (2004)
21. Nilsson, N.J., Fikes, R.E.: Strips: A new approach to the application of theorem proving to problem solving. Artifical Intelligence 2(3-4), 189–208 (1971)
22. Petasis, G., Karkaletsis, V., Paliouras, G.: Ontology population and enrichment: State of the art. Tech. rep. (2007)
23. Poon, H., Domingos, P.: Unsupervised ontology induction from text. In: Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (2010)
24. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall International (2003)
25. Strohmaier, M., Kröll, M.: Analyzing human intentions in natural language text. In: Proceedings of The Fifth International Conference on Knowledge Capture (K-CAP 2009) (September 2009)
26. Strohmaier, M., Lux, M., Granitzer, M., Scheir, P., Liaskos, S., Yu, E.: How do users express goals on the web? - an exploration of intentional structures in web search. In: We. Know 2007 (2007)

27. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the EMNLP/VLC 2000, Hong Kong, pp. 63–70 (2000)
28. Valarakos, A.G., Paliouras, G., Karkaletsis, V., Vouros, G.A.: Enhancing Ontological Knowledge Through Ontology Population and Enrichment. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 144–156. Springer, Heidelberg (2004)