

Program Synthesis for Multi-Agent Question Answering ^{*}

Richard Waldinger¹, Peter Jarvis¹, and Jennifer Dungan²

¹ Artificial Intelligence Center, SRI International
Menlo Park, California, US
{waldinger, jarvis}@ai.sri.com
<http://www.ai.sri.com>

² Ecosystem Science and Technology Branch, NASA Ames Research Center
Moffett Field, California, US
Jennifer.L.Dungan@nasa.gov
<http://geo.arc.nasa.gov/~jennifer/dungan.html>

Abstract. Techniques that were developed for program synthesis are being applied to allow multiple agents to communicate with each other and cooperate to solve a single problem. We illustrate the use of program synthesis techniques in a system that answers questions about geography.

1 Introduction

Deductive program synthesis ([13]; see also [18] and [7]) is the application of theorem-proving techniques to construct a program that meets a given specification. The totally automatic synthesis of useful programs, which requires a combination of resolution and mathematical-induction theorem proving, is probably still beyond the capabilities of current theorem proving technology (but see, e.g., [2]).

Techniques originally developed for program synthesis, however, have been applied effectively to less demanding problems, such as the composition of library software to meet specifications [12]. In this area, there are useful programs whose structure is a simple sequence of subroutine calls. Because repetitive constructs (iteration or recursion) can be hidden in the given subroutines, the problem of intermixing resolution and induction is avoided.

Another such problem is to choreograph cooperation between multiple disparate agents. We imagine that there are many programs that have not been designed to work together and may adopt different notations and representational conventions. These programs may be regarded as agents if they are invoked not by name, but by “advertisement.” We assume that there is a common logical theory, and that a specification of the capabilities of each of the agents is provided by one or more axioms in the theory. A new problem is phrased as a theorem in the theory, and agents are invoked when the axioms that advertise

^{*} To Zohar Manna—colleague, co-author, guide, and friend—for his sixty-fourth birthday—RW

them are involved in the search for a proof; they stand forward when they are appropriate.

We illustrate this approach with the task of answering questions in geography.

2 Geographical Question Answering

Many questions cannot be answered from information in a single geographical source; often the answer must be deduced from information provided by several sources. It may not be obvious which sources to consult. Disparate sources seldom agree on conventions of nomenclature or notation. It becomes a problem to determine what place corresponds to a particular description; the same name may apply to many places, and the same place may have many names.

In the system GeoLogica, the coordination between multiple information sources is carried out by a theorem prover that draws inferences in a formal geospatial theory. GeoLogica differs in purpose from a search engine in that, instead of merely finding a list of documents with vocabulary that matches the question, it attempts to understand the question and provide an answer.

In developing this system, we have been forced to develop systematic ways for naming places, and for identifying places corresponding to given descriptions. It turns out to be quite appropriate to describe the naming of places in the context of an axiomatic theory.

In this paper, we shall first describe the GeoLogica system. We'll present the representation of place names and discuss mechanisms for finding places corresponding to a given description. We'll discuss the solution of a sample problem, mention some related work, and describe proposed extensions.

3 Outline of GeoLogica

Questions are posed to GeoLogica in a subset of English and translated into logic by a natural language parser, the system Gemini [4]. The logical form of the question is rephrased as a theorem and presented to the theorem prover SNARK [17]. (A knowledgeable user of GeoLogica may prefer to bypass the parser and phrase the query directly in logical form.) The geospatial theory that SNARK uses for this application consists of a set of first-order axioms that provide definitions and describe properties of important spatial constants, functions, and relations, including those in the logical form of the query. When SNARK proves a theorem, it not only shows that the theorem follows logically from the axioms in the theory, but also extracts from the proof an answer to the query encoded in the theorem. This answer-extraction mechanism is developed directly from program synthesis techniques.

Using the appropriate axioms, SNARK transforms and decomposes the query into simpler and simpler subqueries. If the right subqueries of the query are answered, SNARK can use the proof to compose an answer to the main query from the answers to the subqueries. Answers may be logical terms or, when

demanded, visualizations, such as maps or satellite images. There may be many proofs of a theorem, and each proof may yield a different answer; it is possible to induce SNARK to find more and more proofs of the same theorem, and hence more and more answers to the query.

SNARK has some geospatial knowledge that is built into its axioms, but it has access to a far larger body of knowledge through its procedural-attachment mechanism. Procedural attachment allows subqueries to be answered by external knowledge sources, which may be programs or data bases and may reside on any machine accessible through the Web.

The capabilities of agents are advertised by axioms in the theory, and the agents themselves are linked to symbols of the theory, so they may be invoked when they are appropriate. The procedural-attachment mechanism allows SNARK to behave as if the information possessed by the external agents were represented as axioms in the geospatial theory.

4 Agents

Among the external agents we have been accessing are

- the Alexandria Digital Library Gazetteer [8]: a repository of information about some six million place names, including a latitude/longitude or bounding box, a geographical type, a list of alternative names, a small map, and a list of regions that include the place in question.
- the CIA World Factbook [3]: an almanac of most of the world's countries, including geographic, economic, and political information about each, such as its principal subdivisions, bordering countries, capital cities, religions, and principal exports.
- the ASCS [14] search engine, which searches the Web for pages that are encoded in DAML (the DARPA Agent Markup Language) and extracts their content; much of the Factbook has been encoded in DAML and is actually accessed through ASCS.
- the TextPro [1] information-extraction engine, a system that can derive information from English text documents.
- a variety of procedures for performing numerical and geographical computations, such as those involving latitudes and longitudes and bounding boxes.
- a number of providers of maps and other geographical images, including TerraVision [16], NIMA's Geospatial Engine (<http://geoengine.nima.mil>), Online Map Creation based on Generic Mapping Tools (<http://www.aquarius-geomar.de/omc/>), the NASA Goddard Distributed Active Archive (<http://daac.gsfc.nasa.gov/WEBGIS/>), and the USGS Landsat Project (<http://glovis.usgs.gov/>). When we answer a question, we are constructing a program that may invoke TerraVision or another visualization system.

A new agent can be added to the system by providing one or more axioms that advertise its capabilities and introducing a link between a symbol in the theory and the agent.

5 Answer Extraction from Proofs

A query to GeoLogica is translated by Gemini into a logical expression

```
find ?z such that Q[?z].
```

(Our convention is to prefix variables with question marks.) This corresponds to finding a (sufficiently) constructive proof of the theorem

```
exists (?z) Q[?z].
```

During the proof, the variable ?z will be instantiated (i.e., replaced) by other terms, which may in turn contain other variables to be instantiated.

Let us give a simple example. Suppose our query is

```
find ?z such that borders(france, ?z).
```

In other words, our task is to find an entity ?z that borders France. Assume that our theory contains the axiom

```
borders(france, germany),
```

i.e., France borders Germany. Then the proof is immediately complete, and the variable ?z is instantiated to Germany. That is, Germany is an answer to our query.

Of course, in general, the answer will not be provided by a single axiom; a sequence of inferences will be necessary, and components of the answer will only be discovered gradually.

This is similar to, but simpler than, a program synthesis problem, in which the program is specified by

```
f(a) <=  
  find ?z such that Q[a, ?z].
```

This corresponds to proving a theorem

```
forall (a) (exists (?z) Q[a, ?z]).
```

In the question-answering case, when we are dealing with a particular question and a fixed input **a**, we do not have the surrounding universal quantifier.

6 The Geospatial Theory

The geospatial theory provides the meanings for the symbols in the query, describes the relationships between various geospatial concepts, advertises the capabilities of our agents, and serves as a repository of knowledge in its own right. One of the first problems we had to face in developing this theory was to provide a way of representing named geographical features. It is desirable for such a place to have a name that allows us to identify it uniquely. For instance, while

Springfield, United States does not uniquely specify a town, Springfield, Illinois, United States does.

We have developed a hierarchical naming scheme that allows us to mimic the mechanism used in addressing mail to provide a logical term that designates a named place.

We distinguish between

- regions, which stand for areas on Earth, not necessarily contiguous.
- geographical feature types, such as country, city, lake, or school.
- subregion indicators, such as Illinois or Springfield, which name subregions of a given region.

Then our named regions are built up as follows:

- earth is the entire Earth.
- `feature(?geographical-feature-type, ?subregion-indicator, ?region)`

is the subregion of `?region` whose name is `?subregion-indicator` and whose type is `?geographical-feature-type`.

Our convention is that `?region` is a variable that stands for a region, and so forth.

For instance,

```
feature(country, canada, earth)
```

stands for the country Canada, and

```
feature(city,
        paris,
        feature(country, france, earth))
```

stands for the city Paris, France. (Actually, GeoLogica abbreviates countries by their names; thus France is simply `france`.) Properties of these terms are specified by axioms in the geospatial theory.

It is not intended that notations of the above form are to be used by people; the average user of GeoLogica will never see them. They are used for the internal representation of places as terms in the logical theory.

Nothing guarantees that terms of the above form stand for places that exist or are unique:

```
feature(city,
        springfield,
        united-states)
```

is not unique, and

```
feature(city, new-york, japan)
```

does not exist.

7 Strings, Names, and Places

In formulating the geospatial theory we have found it necessary to distinguish between strings, names, and the entities they stand for. For instance, `france` is a symbol that denotes the actual country France, a region on Earth. The string “France”, on the other hand, is merely a string, that could be the name of a person rather than a country. Between them we have the name

```
regionq("France")
```

which denotes the name of the country France, not the country itself. Here `regionq` maps a string into the corresponding name. We do not assume that a country has a unique name, nor is it impossible for the same name to stand for two countries, although this is more common for cities than for countries. The relation between the name for a region and the region itself is called `region-val`. In other words,

```
region-val(regionq(?region-string),  
           ?country)
```

is true if `?region-string` is a name for `?country`. Thus, for example,

```
region-val(regionq("France"), france)
```

is true because “France” is a name for the country France.

8 Geographical Feature Types

It is necessary to deal explicitly with geographical feature types within a geospatial theory. We may be asked to find a feature of a certain type. The Alexandria Digital Library Gazetteer requires us to specify the type of a region before it will attempt to find it. We have incorporated the ADL’s feature type classification scheme, which is hierarchical. For example, `capital` is a subtype of `city`, which is a subtype of `populated place`.

We also distinguish between types and their name strings, because different agents may have different conventions for specifying types. The ADL, for instance, uses plural strings to stand for types. Thus the string “countries” corresponds to the geographical feature type `country`.

9 Latitude/Longitude Pairs and Bounding Boxes

Another way of describing places on Earth is by coordinate systems, such as latitude and longitude. There are many representations for latitudes and longitudes, in terms of numbers or strings. For example, the 37th south latitude can be represented by the signed string “-37” or the compass string “37S”. We can also use decimal notation, or the notation based on degrees, minutes, and seconds. Different knowledge agents will produce different representation of latitude

and longitude as outputs, and expect different representations as inputs. For instance, the Alexandria Digital Library Gazetteer agent accepts and produces latitudes and longitudes in signed string notation. The agent that computes the distance between latitude/longitude pairs requires latitudes and longitudes in compass notation. The axiom that advertises an agent must specify the notations expected and produced. The geospatial theory, therefore, must discriminate between these notations. Also, some agents will be able to convert from one representation to another.

For latitudes and longitudes represented in compass notation, there is a function

```
lat-long-compass(?lat-compass, ?long-compass)
```

that yields the corresponding latitude/longitude pair. Similarly, for the signed notation, there is the function

```
lat-long-sign-string(?lat-sign-string,
                    ?long-sign-string).
```

The bounding box of a region is the smallest rectangle that encloses the region, where the sides of the bounding box are made of latitude and longitude lines. Our notations for bounding boxes resembles those for latitude/longitude pairs. Thus

```
bounding-box-sign-string(?lat-sign-string1,
                        ?lat-sign-string2,
                        ?long-sign-string1,
                        ?long-sign-string2)
```

is the bounding box determined by the four numbers given in signed string notation. The numbers correspond to the north, south, east, and west extremes, respectively, of the region in question.

In the language of the geospatial theory, `region-to-lat-long(?region)` is the function that maps a region into its bounding box. Note that the bounding box of a region may contain a lot of terrain outside the region in question. The bounding box of the United States includes much of Canada and Mexico.

The geospatial theory can be extended to deal with representations of the boundaries of regions, such as vectors of latitude/longitude pairs.

10 The Procedural-Attachment Mechanism

The procedural-attachment mechanism allows an agent that is attached to a symbol in the theory to be executed while the proof is in progress. Let us consider a simple example. Suppose that our proof-in-progress contains somewhere the term

```
plus(?real, 2),
```

where ?real stands for a real number. Assume that the symbol plus is attached to an agent, an ordinary program, that performs numerical addition. Because ?real is a variable that has not yet been instantiated, the agent cannot operate. But now assume that, at some step in the proof, the variable ?real is instantiated by constant 3. Then the resulting term

```
plus(3, 2)
```

is sent to the external addition program, which returns the constant 5.

The Alexandria Digital Library Gazetteer is used for several purposes, and the Gazetteer is procedurally attached to more than one symbol in the theory. For instance, the symbol place-to-lat-long invokes the Gazetteer simply to find the bounding box corresponding to some place whose name is ?region-string and whose type is ?geo-feature-type-string:

```
place-to-lat-long(?region-string,  
                  ?geo-feature-type-string,  
                  ?lat-sign-string1,  
                  ?lat-sign-string2,  
                  ?long-sign-string1,  
                  ?long-sign-string2).
```

The procedural attachment of the above relation is used mainly to find the bounding boxes of countries, which are usually uniquely specified by their names.

For example, suppose our proof-in-progress contains the formula

```
place-to-lat-long("Zimbabwe", "countries",  
                  ?lat-sign-string1,  
                  ?lat-sign-string2,  
                  ?long-sign-string1,  
                  ?long-sign-string2).
```

Then the ADL Gazetteer will be invoked to find the bounding box for Zimbabwe. The variables in the above formula will be instantiated appropriately. SNARK will behave exactly as if the axiom

```
place-to-lat-long("Zimbabwe", "countries",  
                  "-15.22", "-22.93", "33.65", "25.11").
```

were included in the geospatial theory. For this reason, we call the above sentence a “virtual axiom.”

Other symbols are procedurally attached to more complex invocations of the ADL that find a region that is a subregion of a given named region, bounding box, or both. These invocations are necessary when there are many places with the same name and we need to tell the Gazetteer which one we mean.

11 Axioms that Advertise the Capabilities of an Agent

By advertising an agent with one or more axioms, we allow it to be invoked when it is appropriate. This makes it easy to add new agents without reprogramming the system. When a new query is presented, the agents that are appropriate for its subqueries stand forward, invoked not by name but as a byproduct of the theorem-proving process.

Some of the simplest of these axioms are those that advertise agents that translate from one notation to another. For example, the agent that translates from the signed string notation to the compass notation for latitude and longitude is advertised by the axiom

```
lat-long-compass(?lat-compass, ?long-compass) =
lat-long-sign-string(lat-compass-to-sign-string(?lat-compass),
long-compass-to-sign-string(?long-compass)).
```

Here `lat-compass-to-sign-string` and `long-compass-to-sign-string` are function symbols with procedural attachments to programs that perform the conversion from compass notation to signed string notation, for latitudes and longitudes, respectively.

Applying this axiom to a lat-long pair in compass notation, such as

```
lat-long-compass("37N", "122E")
```

will yield the term

```
lat-long-sign-string(lat-compass-to-sign-string("37N"),
long-compass-to-sign-string("122E")).
```

Because of the procedural attachments to the function symbols, the compass notation will be converted to signed string notation, yielding the term `lat-long-sign-string("37", "-122")`.

Now let us look at one of the axioms that advertises the Alexandria Digital Library Gazetteer. The following axiom will invoke the Gazetteer to find the bounding box for a country:

```
(region-to-lat-long(?country) =
bounding-box-sign-string(?lat-sign-string1,
?lat-sign-string2,
?long-sign-string1,
?long-sign-string2))
<=
(region-val(regionq(?region-string), ?country) &
place-to-lat-long(?region-string, "countries",
?lat-sign-string1,
?lat-sign-string2,
?long-sign-string1,
?long-sign-string2)).
```

In other words, to find the bounding box (in signed string notation) for a country, find a string that can serve as a name for the country and submit that string, with geographical feature type string “countries”, to the Gazetteer. The resulting four number strings will correspond to the desired bounding box.

12 A Sample Problem

Let us consider a problem solved by GeoLogica to illustrate the discovery of a place characterized by a logical combination of properties. The query is

Show a petrified forest in Zimbabwe within 750 miles of the capital of South Africa.

One way to show a region is to display a three-dimensional visualization of its satellite image using the TerraVision terrain viewer.

The logical form of the query is

```
find ?x such that
  show(?x) &
  patient(?x, ?y) &
  petrified-forest(?y) &
  in(?y, zimbabwe) &
  within-dist-of(?y, within-dist-of(?z, ?u)) &
  mile-unit(?z) &
  count(?z, 750) &
  capital(?u) &
  capital-of(?u, south-africa).
```

In other words, we want to find ?x which is a showing of ?y, where ?y is a petrified forest that is in Zimbabwe and within a distance ?z of ?u, where ?z is in units of miles and has a magnitude of 750, and ?u is the capital of South Africa.

We will not follow the proof in detail; it will be described informally, with indication of where the principal agents were invoked, and what virtual axioms were introduced.

To show a region in TerraVision, it is necessary to find the center of its bounding box, because that is the point we have TerraVision focus on. To find a petrified forest in Zimbabwe, we first find the bounding box of Zimbabwe; this is given by the virtual axiom

```
place-to-lat-long("Zimbabwe", "countries",
  "-15.22", "-22.93", "33.65", "25.11"),
```

as we have seen.

Then we search for petrified forests within that bounding box, checking to see that they are indeed in Zimbabwe. The subquery is

```

place-to-lat-long-part-of-type-bounds(
  ?region-string, "petrified forests",
  "Zimbabwe", "countries",
  "-15.22","-22.93","33.65","25.11"
  ?lat-sign-string1, ?lat-sign-string2,
  ?long-sign-string1, ?long-sign-string2).

```

This causes an invocation of the Alexandria Digital Library Gazetteer in a more complicated way than before. We only specify the type, petrified forest, of the region we are looking for, not its name; we search only within the bounding box of Zimbabwe; and we insist that the found region be a subregion of the country Zimbabwe.

The location of the single petrified forest in Zimbabwe is given by the virtual axiom

```

place-to-lat-long-part-of-type-bounds(
  "Makuku Fossil Forest", "petrified forests",
  "Zimbabwe", "countries",
  "-15.22","-22.93","33.65","25.11"
  "-15.65","-15.65","29.95","29.95").

```

Note that, though we have requested a bounding box, the ADL Gazetteer has actually given us a latitude/longitude pair -15.65, 29.95; the north and south latitudes, and the east and west longitudes, are respectively the same.

It is still necessary to ensure that the petrified forest we have found satisfies the additional constraint, that it be within 750 miles of the capital of South Africa. The name of the capital of South Africa, Pretoria, is discovered by invoking the ASCS search engine, which contains the DAML encoding of the CIA World Factbook, including the capitals of countries.

To ensure that the forest is within 750 miles of Pretoria, we first find the latitude and longitude of Pretoria, using the Gazetteer again. We then use a geographical computational agent to ensure that the distance between the latitude/longitudes for the forest and for Pretoria is sufficiently small. The virtual axiom provided by the agent is

```

lat-long-dist("25.75S","28.16667E", "15.65S","29.95E",
              "708.0386").

```

Note that this agent requires compass notation—it will not accept signed strings; that conversion is performed by another agent. The distance agent determines that the distance between Pretoria and the forest is 708 miles, within the 750-mile limit that was specified in the query.

13 SNARK

Although many theorem provers can be used for question answering, SNARK is particularly well suited, for a number of reasons.

- It has strategic controls that allow it to exhibit high performance in a particular subject domain, finding proofs of hundreds of steps in theories with hundreds or thousands of axioms.
- It has a mechanism for extracting answers from proofs.
- It has a procedural-attachment mechanism.
- It has special built-in procedures for reasoning efficiently about space and time.

SNARK is a first-order logic theorem prover with resolution and paramodulation, implemented in Common Lisp. It has been used in NASA's Amphion system, for automatic software composition, and the Specware formal software development system of the Kestrel Institute, as well as several SRI systems. The current geospatial theory has about a thousand axioms. Proofs commonly contain about a hundred inference steps.

14 Other Query Forms

Here are some other forms of queries that can be answered using the current implementation of GeoLogica:

Is Zimbabwe north of South Africa?
 What is the capital of Zimbabwe?
 What is the distance from Arcturus, Zimbabwe to the capital of Cuba?
 Display the Generic Mapping Tools map for a beach in Thailand.
 Show a cave in Afghanistan within 100 miles of Kandahar, Afghanistan.
 Show another.

A question such as "Show another" is treated by allowing SNARK to continue where it left off, finding another proof to the previous theorem, and hence another answer to the previous question. This can be done repeatedly until the set of answers found is depleted.

15 Relation to Other Work

There is a large body of work, not surveyed here, in using theorem proving to answer questions or to synthesize programs from specifications. The approach of using procedural attachment to a logical theory to coordinate multiple agents is relatively new, but see Infomaster [6] and Ariadne [10]. The system CYC [11] includes a large theory of spatial reasoning and now also incorporates procedural attachment.

Fonseca et al. [5] are developing a geographical ontology, but it seems not to include axioms, only vocabulary. Hobbs [9] is leading a team in building a more spatial ontology and theory in DAML. The Sweet Ontology (Raskin, [15]) is specifically for the earth sciences. These should be valuable resources for us.

16 Plans for Future Work

Rather than merely finding and displaying individual places, we hope to introduce capabilities for dealing with and manipulating data, obtained from online agents, for collections of places. We could ask for differences, averages, and maximums, or compare figures from different years. Results could be displayed in charts and tables.

Although GeoLogica may answer a question appropriately, it is usually poor at explaining or defending its answer. Yet its user may not wish to accept the answer without question, particularly if the veracity of any of the sources is in doubt. The proof of the theorem contains a full explanation, but most people do not find logical proofs easy to understand. We plan to develop a more comprehensible explanation facility, in which an understandable explanation will be extracted from the proof, just as the answer itself is now.

SNARK has facilities for temporal reasoning, including date and time arithmetic and the relationships between temporal intervals. However, we have not yet introduced time into the geospatial theory. Taking this step will enable us to deal with changes, such as historical changes in the names and boundaries of countries, changes in the environment, weather, and the movement of populations and individuals.

So far we have been dealing with isolated questions, but it may be more realistic to imagine an information-seeking dialogue, in which the user of GeoLogica poses a series of questions, each refining or elaborating on the one before, and some providing new information to be used in answering future questions.

We have added facilities for extracting information from English text (Text-Pro), but we have not yet used this in the geographical domain. Care must be observed because information obtained through information extraction is not so reliable as that obtained from other sources.

We are currently applying similar techniques to provide tools of interest to intelligence analysts.

17 Conclusion

Program synthesis techniques have been found useful for solving simpler problems, such as software composition and multi-agent interoperability. As theorem-proving techniques become more powerful, it may be time to apply them once again to program synthesis.

Acknowledgments This research has been supported by the NASA Intelligent Systems Program, the ARDA Acquaint Program, and the DARPA DAML Program.

We would like to thank Doug Appelt, Chris Culy, John Fry, Jerry Hobbs, David Israel, David Martin, Martin Reddy, Susanne Riehemann, Mark Stickel and Mabry Tyson for contributions to the conceptualization, implementation, and presentation of this work.

References

1. Appelt, D. E., Martin, D. L. Named Entity Recognition in Speech: Approach and Results Using the TextPro System <http://www.nist.gov/speech/publications/darpa99/html/ie30/ie30.htm> (1999).
2. Bundy, A., Smaill, A. D., Wiggins, G. The Synthesis of Logic Programs from Inductive Proofs. in Lloyd, J. *Computational Logic* Springer-Verlag (1990) 135–149.
3. Central Intelligence Agency The World Factbook 2002. <http://www.cia.gov/cia/publications/factbook/> (2002).
4. Dowding, J., Gawron, J. M., Appelt, D., Bear, J., Cherny, L., Moore, R., Moran, D. GEMINI: A Natural Language System for Spoken-Language Understanding. *Proceedings of the 31st Annual Meeting of the Ass. for Computational Linguistics*, (1993) 54–61.
5. Fonseca, F., Egenhofer, M., Davis, C., Camara, G. Semantic granularity in ontology-driven geographic information systems *AMAI Annals of Mathematics and Artificial Intelligence—Special Issue on Spatial and Temporal Granularity*, 36(1–2): (2002) 121–151.
6. Genesereth, M., Keller, A. M. Duschka, O. M. Infomaster: an Information Integration System. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(1997)539–542.
7. Green, C. C.: Application of Theorem Proving to Problem Solving. *Proc. Int. Joint Conf. on Artificial Intelligence*(1969) 219–239.
8. Hill, L. J., Frew, J., Zheng, Q. Geographic Names: The Implementation of a Gazetteer in a Georeferenced Digital Library. *D-Lib* (1999)
9. Hobbs, J. et al. A DAML Spatial Ontology. <http://www.daml.org/listarchive/damls-spatial/0002.htm> (2003).
10. Knoblock, C. A., Minton, S.. The Ariadne Approach to Web-Based Information Integration. *IEEE Intelligent Systems*, 13 5 (1998) 17–20.
11. Lenat, D. B., Guha, R. V. Enabling Agents to Work Together. *Communications of the ACM*, 37 7 (1994)
12. Lowry, M., Philpot, A., Pressburger, T., Underwood, I., Waldinger, R., Stickel, M., Amphion: Automatic Programming for the NAIF Toolkit, in *NASA Science Information Systems Newsletter, Issue 31, Feb. 1994, pp. 22-25.*
13. Manna, Z., Waldinger, R.: A Deductive Approach to Program Synthesis. *ACM Trans. on Programming Languages and Systems* 2 (1980) 90–121.
14. Pease, A., Li, J., Barbee, C. DAML Agent Semantic Communications Service (ASCS) <http://oak.teknowledge.com:8080/daml/damlquery.jsp> (2002)
15. Raskin, R. Semantic Web for Earth and Environmental Terminology (SWEET). *Earth Science Technology Conference* Adelphi, MD (2003)
16. Reddy, M., LeClerc, Y.-G., Iverson, L., Bletter, N. TerraVision II: Visualizing Massive Terrain Databases in VRML. *IEEE Computer Graphics and Applications (Special Issue on VRML)*. 19 2 (1999) 30–38.
17. Stickel, M. E., Waldinger, R. J., Chaudhri, V. K. A Guide to SNARK. SRI International, Menlo Park, CA (2000).
18. Waldinger, R., Lee, R. C. T.: PROW: A Step Toward Automatic Program Writing. *Proc. Int. Joint Conf. on Artificial Intelligence*(1969) 241–252.