Whatever Happened to Deductive Question Answering?

Richard Waldinger

Artificial Intelligence Center, SRI International

Deductive question answering, the extraction of answers to questions from machine-discovered proofs, is the poor cousin of program synthesis. It involves much of the same technology—theorem proving and answer extraction—but the bar is lower. Instead of constructing a general program to meet a given specification for any input—the program synthesis problem—we need only construct answers for specific inputs; question answering is a special case of program synthesis. Since the input is known, there is less emphasis on case analysis (to construct conditional programs) and mathematical induction (to construct looping constructs), those bugbears of theorem proving that are central to general program synthesis. Program synthesis as a byproduct of automatic theorem proving has been a largely dormant field in recent years, while those seeking to apply theorem proving have been scurrying to find smaller problems, including question answering.

Deductive question answering had its roots in intuitionistic and constructive logical inference systems, which were motivated by philosophical rather than computational goals. The idea obtained computational force in McCarthy's 1958 Advice Taker, which proposed developing systems that inferred conclusions from declarative assertions in formal logic. The Advice Taker, which was never implemented, anticipated deductive question answering, program synthesis, and planning.

Slagle's Deducom obtained answers from proofs using a machine-oriented inference rule, Robinson's resolution principle; knowledge was encoded in a knowledge base of logical axioms (the *subject domain theory*), the question was treated as a conjecture, a theorem prover attempted to prove that the conjecture followed from the axioms of the theory, and an answer to the question was extracted from the proof. The answer-extraction method was based on keeping track of how existentially quantified variables in the conjecture were instantiated in the course of the proof.

The QA3 program of Green, Yates, and Rafael integrated answer extraction with theorem proving via the *answer literal*. Chang, Lee, Manna, and Waldinger introduced improved methods for conditional and looping answer construction. Answer extraction became a standard feature of automated resolution theorem provers, such as McCune's Otter and Stickel's SNARK, and was also the basis for logic programming systems, in which a special-purpose theorem prover served as an interpreter for programs encoded as axioms.

Deductive databases allow question answering from large databases but use logic programming rather than a general theorem prover to perform inference.

N. Dershowitz and A. Voronkov (Eds.): LPAR 2007, LNAI 4790, pp. 15–16, 2007.

The Amphion system (of Lowry et al.), for answering questions posed by NASA planetary astronomers, computed an answer by extracting from a SNARK proof a straight-line program composed of procedures from a subroutine library; because the program contained no conditionals and no loops, it was possible for Amphion to construct programs that were dozens of instructions long, completely automatically. Software composed by Amphion has been used for the planning of photography in the Cassini mission to Saturn.

While traditional question-answering systems stored all their knowledge as axioms in a formal language, this proves impractical when answers depend on large, constantly changing external data sources; a procedural-attachment mechanism allows external data and software sources to be consulted by a theorem prover while the proof is underway. As a consequence, relatively little information needs to be encoded in the subject domain theory; it can be acquired if and when needed. While external sources may not adhere to any standard representational conventions, procedural attachment allows the theorem prover to invoke software sources that translate data in the form produced by one source into that required by another. Procedural attachment is particularly applicable to Semantic Web applications, in which some of the external sources are Web sites, whose capabilities can be advertised by axioms in the subject domain theory.

SRI employed a natural-language front end and a theorem-proving central nervous system (SNARK) equipped with procedural attachment to answer questions posed by an intelligence analyst (QUARK) or an Earth systems scientist (GeoLogica). While non-computer-scientists found the natural language input to be more congenial than logic, it turned out to be difficult to restrict questions to be within the system's domain of expertise.

In this talk we will describe recent efforts (BioDeducta) for deductive question answering in molecular biology, done in collaboration with computational biologist Jeff Shrager. Questions expressed in logical form are treated as conjectures and proved by SNARK from a biological subject-domain theory; access to multiple biological data and software resources is provided by procedural attachment. We illustrate this with the discovery of the gene responsible for light adaptation in cyanobacteria, which are water bacteria capable of photosynthesis.

A proposed query-elicitation mechanism allows a biological researcher to construct a logical query without realizing it, by choosing among larger and larger English-language alternatives for fragments of the question. A projected explanation mechanism constructs from the proof a coherent English explanation and justification for the answer.

While the annotation language OWL has achieved some currency as a representation vehicle for subject domain knowledge, we argue that it and proposed Semantic Web rule languages built around it are inadequate to express queries and reasoning for Semantic Web question answering. Lacking are such central features as full quantification and equality reasoning. It is also impossible to express a closed-world assumption, which states that a particular source provides exhaustive information on a given topic.