DOWNLOAD PDF CONCEPTUAL GRAPHS FOR KNOWLEDGE REPRESENTATION

Chapter 1 : Download [PDF] graph structures for knowledge representation and reasoning

Conceptual graphs (CGs) are a formalism for knowledge blog.quintoapp.com the first published paper on CGs, John F. Sowa used them to represent the conceptual schemas used in database systems.

Conceptual graph Save Conceptual graphs CGs are a formalism for knowledge representation. In the first published paper on CGs, John F. Sowa Sowa used them to represent the conceptual schemas used in database systems. The first book on CGs Sowa applied them to a wide range of topics in artificial intelligence , computer science , and cognitive science. Since , the model has been developed along three main directions: A graphical interface for first-order logic Elsie the cat is sitting on a mat In this approach, a formula in first-order logic predicate calculus is represented by a labeled graph. The diagram above is an example of the display form for a conceptual graph. Each box is called a concept node, and each oval is called a relation node. The letters x and y, which are called coreference labels, show how the concept and relation nodes are connected. In CLIF, those letters are mapped to variables, as in the following statement: Reasoning can be done by translating graphs into logical formulas, then applying a logical inference engine. Diagrammatic calculus of logics Another research branch continues the work on existential graphs of Charles Sanders Peirce , which were one of the origins of conceptual graphs as proposed by Sowa. In this approach, developed in particular by Dau Dau , conceptual graphs are conceptual diagrams rather than graphs in the sense of graph theory , and reasoning operations are performed by operations on these diagrams. All kinds of knowledge ontology, rules, constraints and facts are labeled graphs, which provide an intuitive and easily understandable means to represent knowledge. Reasoning mechanisms are based on graph notions, basically the classical notion of graph homomorphism ; this allows, in particular, to link basic reasoning problems to other fundamental problems in computer science problems concerning conjunctive queries in relational databases , constraint satisfaction problem , etc. The formalism is logically founded, i. From a computational viewpoint, the graph homomorphism notion was recognized in the s as a central notion, and complexity results and efficient algorithms have been obtained in several domains. Sentence generalization and generalization diagrams Sentence generalization and generalization diagrams can be defined as a special sort of conceptual graphs which can be constructed automatically from syntactic parse trees and support semantic classification task Galitsky et al Similarity measure between syntactic parse trees can be done as a generalization operation on the lists of sub-trees of these trees. The diagrams are representation of mapping between the syntax generalization level and semantics generalization level anti-unification of logic forms. Generalization diagrams are intended to be more accurate semantic representation than conventional conceptual graphs for individual sentences because only syntactic commonalities are represented at semantic level.

DOWNLOAD PDF CONCEPTUAL GRAPHS FOR KNOWLEDGE REPRESENTATION

Chapter 2 : Dynamic Conceptual Graphs

Conceptual Graphs John F. Sowa Abstract. A conceptual graph (CG) is a graph representation for logic based on the semantic networks of artificial intelligence and the existential graphs of Charles Sanders Peirce.

Logical, Philosophical, and Computational Foundations". Over the decades he has researched and developed emerging fields of computer science from compiler, programming languages, and system architecture [4] to artificial intelligence and knowledge representation. He is a fellow of the Association for the Advancement of Artificial Intelligence. With this company he was developing data-mining and database technology, more specific high-level " ontologies " for artificial intelligence and automated natural language understanding. Currently Sowa is working with Kyndi Inc. Conceptual graph Sowa invented conceptual graphs, a graphic notation for logic and natural language, based on the structures in semantic networks and on the existential graphs of Charles S. He published the concept in the article "Conceptual graphs for a data base interface" in the IBM Journal of Research and Development. In the s this theory had "been adopted by a number of research and development groups throughout the world. Principles of Semantic Networks. International Conference on Conceptual Structures 2nd: Conceptual structures, current practices: Applications, Implementation and Theory". Current Research and Practice. International journal of human-computer studies. Encyclopedia of Cognitive Science.. Conceptual graphs are devised as a language of knowledge representation by Sowa , based on philosophy, psychology and linguistics. Knowledge in conceptual graph form is highly structured by modelling specialised facts that can be subjected to generalised reasoning. Advances in Methodologies, Components, and Management. The original theory of conceptual graphs was introduced by Sowa Sowa, A conceptual graph is a finite, connected, bipartite graph. It includes notions of concepts, relations, and actors Jain Intelligent-Based Systems Engineering. Sowa and John Zachman

DOWNLOAD PDF CONCEPTUAL GRAPHS FOR KNOWLEDGE REPRESENTATION

Chapter 3 : Conceptual Graphs Home Page

This book studies a graph-based knowledge representation and reasoning formalism stemming from conceptual graphs, with a substantial focus on the computational properties.

A graphical representation for logic. Full first-order logic plus metalanguage capabilities. A cat is on a mat. Display form for CGs: Conceptual graph interchange form CGIF: Formally equivalent to CGIF. The default quantifier for each concept is the existential, which says that something of the specified type exists. Each conceptual relation has one or more arcs. In KIF, the quantifier "exists" or "forall" and its associated "sort", such as Cat or Mat, corresponds to a declaration in a programming langage. In CGIF, an asterisk in front of a coreference label declares the concept in which the quantifier and type are associated with the correference label. In the display form, the arcs are numbered from 1 to some upper limit n, which depends on the relation type. Alternatively, the first arc may be shown as an arrow pointing toward the circle, and the n-th arc may be shown as an arrow pointing away from the circle. Every cat is on a mat. By default, universal quantifiers have precedence over the existential quantifiers. If necessary, special concept boxes called contexts may be used to delimit the scope of quantifiers. A person is between a rock and a hard place. The arrow on the third arc may be replaced by the number 3. John is going to Boston by bus. For some purposes, the simplest choice is to represent a verb by a relation with one arc or argument for each participant. Unfortunately, this solution cannot be easily generalized to handle all the options that may occur. For example, the sentence John is going would require a monadic relation, John is going to Boston would require a dyadic relation, and John is going to Boston by bus at noon would require a tetradic relation. The relations named Agnt, Dest, and Inst represent the thematic roles of agent, destination, and instrument, which are commonly used in linguistics. With such relations, arbitrarily many participants and qualifiers can be linked to the same concept. Defining Relations by Lambda Expressions A definition is a metalevel statement that associates a name with a definition of the entity it names. In this example, the Def relation associates the relation name GoR3 with a triadic lambda expression, which defines the corresponding relation. The concept of type LambdaExpression contains a nested conceptual graph, in which the Greek letter lambda is used to mark three of the concepts as formal parameters. The rule of lambda expansion allows any graph that contains a relation of type GoR3 to be expanded to a graph with a concept of type Go linked to relations of types Agnt, Dest, and Inst. The inverse rule of lambda contraction allows any graph with a concept of type Go linked to three relations of type Agnt, Dest, and Inst to be contracted to a graph with a relation of type GoR3. Tom believes Mary wants to marry a sailor. The context boxes delimit the scope of quantifiers and other logical operators. The dotted line from the concept [Person: Mary] to the concept [T], which is called a coreference link, shows that the inner concept [T] has the same referent as concept to which it is linked. In CGIF, the coreference link corresponds to a pair of coreference labels that associate the defining node [Person: The type T, which represents the most general type at the top if the type hierarchy, is true of everything; therefore, a node such as [T: There is a sailor that Tom believes Mary wants to marry. In the corresponding logic, the concept [Sailor] or the existential quantifier for? Therefore, a particular sailor is presumed to exist in the actual world. Graphs reduce the number of variables by showing connections directly. Some algorithms are more efficient on graphs. Some algorithms are more efficient on linear forms. Ability to mix and match tools that take advantage of different structural properties. More natural for certain applications: Map cities to nodes and highways to arcs. Show flow through programs, electrical wiring, and plumbing. Direct mapping to computer networks. ICCS will be held in Bulgaria. Bibliography Chein, Michel, ed. Ellis, Gerard, Robert A. Tepfenhart, William, and Walling Cyre, eds. Dick, and John F. This is a revised version of a presentation by John F.

Chapter 4 : Tutorial on CGs

Conceptual graphs are a knowledge representation language designed as a synthesis of several different traditions. First are the semantic networks, which have been used in machine translation and computational linguistics for over thirty years.

Executable Conceptual Structures based on Conceptual Graphs As described by Sowa, , which is also evident today, although the graphs for doing computation have been developed by different people for different purpose, they share the following common themes: The system is a network of active elements, which may be called actors, functions, or transitions. Processes consist of signal passing from one actor to another, where the signals may be called message, data, tokens, or control marks. The internal structure of any actor is irrelevant to the behavior of the system, Replacing an actor with another one that has a different internal structure but the same message-passing behavior has no effect on the overall system. The actors are either unanalyzed primitives implemented directly in hardware or in a lower-level programming language, or modules which are themselves defined as networks of actors. In this paper, we review in detail three of the above eight executable systems based on Conceptual Graphs. The comparative study is carried out in terms of modeling the factorial function in each of these three approaches. The outline of this paper is as follows. In Sections 2, 3 and 4, detailed implementation and execution of each of the three executable models will be outlined, respectively. In Section 5, we conduct a comparative study on these three models and provide a brief conclusion to this paper. Actor and Dataflow Diagram 2. Definition Sowa carried out a methodical development of a formalism for graphs of actors that are bounded to conceptual graphs. As a conceptual graph is constructed by joining smaller graphs, actor nodes attached to the conceptual graphs are also joined to form a dataflow graph. Then, control maps on the graph are used to trigger the actors and compute referents for generic concepts. Control marks are drawn in the referent field after the colon in the concept node. Control marks direct the flow of a computation. There are three types of control marks: The neutral mark "o" may be drawn explicitly, but it is typically omitted. A dataflow graph is a finite, connected, directed, bipartite graph with one set of nodes called concepts, and the other set of nodes called actors. If an arc is directed from a concept c to an actor t, then the arc is called an input arc of t, and c is called an input concept of t. On the other hand, if an arc of a dataflow graph is directed from an actor t to a concept c, then the arc is called an output arc of t, and c is called an output concept of t. Every actor must be attached to at least one arc either input or output. The arcs of a dataflow graph are drawn as dotted lines to distinguish them from the arcs of conceptual graphs. The concepts are still drawn as boxes, but the actors are drawn as diamonds. Consider the dataflow graph in Figure 1. There are three actors: In conventional programming language, the following statements would generate the same results as this dataflow graph: A Dataflow Graph A concept c in a dataflow graph v is called: If all concepts in a dataflow graph u are also concepts of a conceptual graph v, then u is said to be bounded to v. If no concepts of the dataflow graph u are concepts of any conceptual graph, then u is said to be free. For example, consider the schema below: On the other hand, Figure 1 is a free dataflow graph. Bounded Dataflow Graph 2. Implementation of the Factorial function The "factorial" is a recursive function that takes as its input a positive number and computes its factorial value. Figure 3 lists the pseudo code for this function. The IDENT function takes as input one argument, and produces an output which is the same as its input value. The lower one is enabled. This will enable the output of "c" to be: This will effectively produce the output of "b" to be 1, a follows: This will effectively produce the output of "a" to be 2, as follows: A Recursively Defined Actor for Factorial 3. Definition Mineau, proposes a representation for dynamic processes which has five objectives: This representation proposes an extension to the definition of actors as originally proposed in Sowa, , and is also based on earlier work by Delugach, Basically, the main idea is to view an actor as a state transition mechanism: The pre and postconditions being a conjunction of conceptual graphs, this definition, though similar to the one found in Sowa, , is more general, and provides a framework which improves tremendously

the representational power of the CG theory. Figure 5 shows a simple example of an actor, where g1, g2, g3 and g4 are conceptual graphs. In this case, two assertions will be made: Of course, there may be problems with this state transition: Also, this may trigger thorough knowledge revision activities: In that case, the actor is blocked and the user is notified. It may also be the case that an actor does not change the current state of the system if g3 is already true and g4 already false for instance. It is clear from our model, that the input to an actor is composed of complete conceptual graphs, called input graphs, and not just input concepts as proposed in Sowa, , and that the output from an actor is also composed of conceptual graphs, called output graphs, either asserted or negated, and not just instantiations of generic concepts, as proposed in Sowa, Furthermore, let us mention that the concepts in any graph, either input or output graph, can be coreferenced to any other concept in the knowledge base: So a concept in an input graph can be represented in an output graph as well. Actors that trigger change on concepts need to refer to these concepts in both their input and output graphs. The other main difference between this representation and what is proposed in Sowa, concerns the sequencing mechanism. With simple actors as those of Section 2 above, this is not a problem. However, in the general case, it may be problematic to find all the actors which need to be initially triggered in order to accomplish some task correctly without deadlocks for instance. Generally, we think that the user should not have to interact with the actors in order to produce the right execution sequence; the user should only start the process and the initial tokens should automatically be sent to the relevant actors. Also, in the construction of a CPE, we do not wish to complexify the execution control mechanism. The only mechanism we wish to implement is a precondition checking mechanim. Keeping that in mind, we propose to represent the execution sequence using predicates that will be incorporated into the preconditions of the relevant actors. Doing so, the execution control mechanism we need in order to implement our CPE is the precondition checking mechanism which needed to be developed anyhow. Making sure that each execution sequence is identified by its own unique predicate, the user does not have to know about the sequencing structure of the process in order to engage it. Consequently, each actor will include preconditions needed by the sequencing mechanism of the process it takes part in. Implementation of the Factorial function As mentioned above, a process will be represented as a set of actors linked by an execution sequence explicitly described by predicates in their pre and postconditions. When the preconditions of an actor become conjunctively true, its postconditions will be asserted, updating the current state of the system. For instance, mathematical operators on variables will be implemented using basic actors as presented in Section 2 above ; they represent external procedures which compute some output value from their operands. All sequencing instructions such as if-else and loops will be implemented using the synchronization predicates which will be added to the pre and postconditions of sequence-related actors. In order to produce the synchronization predicates, the algorithm is analyzed and a synchronization graph is produced. Figure 6 below shows our version of the factorial problem using C; Figure 7 shows the corresponding synchronization graph. The Corresponding Synchronization Graph All algorithms will be required to have unique entry and exit points, as shown in the synchronization graph of Figure 7, with the S start and E end nodes. This hides the synchronization details from the user who will need to trigger only the entry node of the synchronization graph in order to engage the process. All synchronization predicates tokens will be instantiated in the proper execution order from that point on. The arc labeled x is used to denote an exclusive-or between two pathways leading from l4 to l11, since their selection is based on the evaluation of a boolean condition in line l4. Normally, different edges reuniting at a node imply a conjunction of preconditions. In this case, it represents a disjunction of preconditions. All other links are set based on the dependencies between the instructions which use the variables of the algorithm, and those which update them. In this example, such dependencies are fairly simple to identify; we will not explain them any further at this time. This means that any cg could be used as an input parameter to a function or process. We need a way to identify how cgs can be exchanged between functions. For that purpose, we define that the input parameters of a function, the cgs expected by a function, will appear in the preconditions of the first transition rule the one associated with line l0. In our example, rule 0 is the first rule of our process 4. Consequently, its precondition

will include a cg describing the information that is expected from the calling function. For now, let this cg be the first graph encountered in the precondition of rule 0 pre0. Coreference links from this graph to any other graph in the process description allows the passing of the appropriate input values to other graphs. Hence, the value of the variable n will be called y1 throughout the whole description of the process 5.

Chapter 5 : Conceptual graph - Wikipedia

John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations, is not limited to conceptual graphs, but provides broad coverage of the entire field. It is available from Barnes and Noble books and blog.quintoapp.com

Chapter 6 : Conceptual Graphs - Introduction to ontologies and semantic web - tutorial

This book studies a graph-based knowledge representation and reasoning formalism stemming from conceptual graphs, with a substantial focus on the computational properties. Knowledge can be symbolically represented in many ways, and the authors have chosen labeled graphs for their modeling and computational qualities.

Chapter 7 : John F. Sowa - Wikipedia

In the early s, J.F. Sowa introduced the conceptual graph (CG) theory which provides a knowledge representation framework consisting of a form of logic with a graph notationand integrating several features from semantic net and frame representations.

Chapter 8 : Conceptual graph | Revolvy

Conceptual Graphs (CG, blog.quintoapp.com) provide a powerful knowledge representation and inference environment, whilst exhibiting the familiar object- oriented and database features of contemporary enterprise and web applications.