

Conceptual Graph Theory Applied to Reasoning in Ontologies

Dan Corbett

Advanced Computing Research Centre
School of Computer and Information Science
University of South Australia
Adelaide, South Australia 5095
+61 (8) 8302 3102
corbett@cs.unisa.edu.au

Abstract. This paper discusses automated reasoning over ontologies represented as Conceptual Graphs. We discuss a tool which has been implemented using Conceptual Graphs as its underlying knowledge structure. The significance of this work is that we demonstrate that the power of logic as implemented in Conceptual Graphs, and the tools available in Conceptual Graph Theory can be used as powerful ontology reasoning tools in a real-world domain. We show that ontologies can be constrained and unified using efficient methods, and that these methods provide the basis for an automated reasoning system. The Conceptual Graph techniques of concept join, partial order and subsumption are all exploited to create these reasoning tools.

We discuss the implementation of these ideas, and demonstrate the software tool created in two domains: building architecture and defense. Examples show that the system can reason over these domains and assist the users in their tasks.

Keywords: Reasoning, visualization, ontologies, conceptual graphs

Conference topics: Reasoning models and Knowledge Representation

Conceptual Graph Theory Applied to Reasoning in Ontologies

Dan Corbett

Advanced Computing Research Centre
School of Computer and Information Science
University of South Australia
Adelaide, South Australia 5095

Abstract. This paper discusses automated reasoning over ontologies represented as Conceptual Graphs. We discuss a tool which has been implemented using Conceptual Graphs as its underlying knowledge structure. The significance of this work is that we demonstrate that the power of logic as implemented in Conceptual Graphs, and the tools available in Conceptual Graph Theory can be used as powerful ontology reasoning tools in a real-world domain. We show that ontologies can be constrained and unified using efficient methods, and that these methods provide the basis for an automated reasoning system. The Conceptual Graph techniques of concept join, partial order and subsumption are all exploited to create these reasoning tools.

We discuss the implementation of these ideas, and demonstrate the software tool created in two domains: building architecture and defense. Examples show that the system can reason over these domains and assist the users in their tasks.

1. A Brief Overview of Conceptual Graphs

Conceptual Structures (or Conceptual Graphs, or $\mathcal{CG}s$) are a knowledge representation scheme, inspired by the existential graphs of Charles Sanders Peirce and further extended and defined by John Sowa [1, 2]. Informally, CGs can be thought of as a formalization and extension of Semantic Networks, although the origins are different. They are labeled graphs with two types of nodes: concepts (which represent objects, entities or ideas) and relation nodes, which represent relations between the concepts. As an example, Figure 1 shows a Conceptual Graph which represents the knowledge that $\text{The cat Felix is sitting on the mat which is known as mat 47.}$

Every concept or relation has an associated type. A concept may also have a specific referent or individual. A concept in a CG may represent a specific instance of that type (e.g., *Felix* is a specific instance, or individual, of type *cat*) or we may

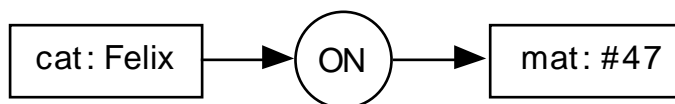


Figure 1. A Simple CG.

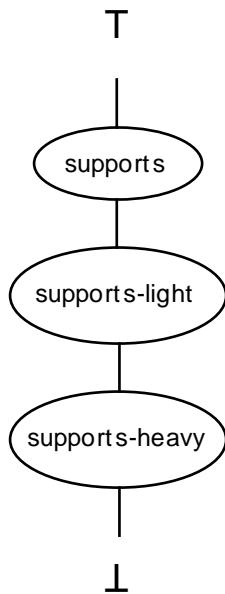


Figure 2. A relation type hierarchy.

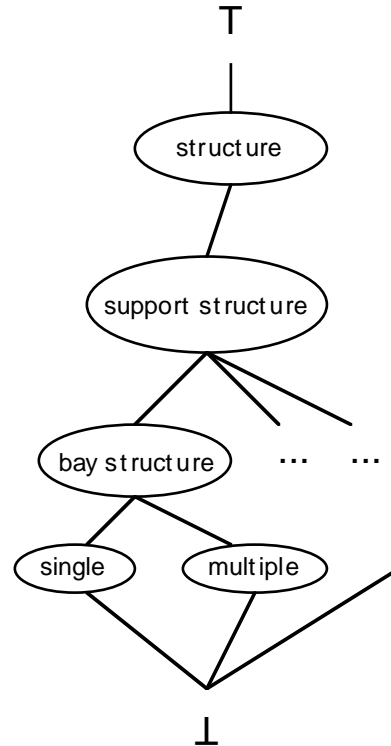


Figure 3. A concept type hierarchy.

choose only to specify the type of the concept. That is to say that a concept may simply represent a generic concept for a type, such as *mammal* or *room*, or a concept may represent a specific object or idea, such as *my cat* or *the kitchen at the Smith house*. In the former case, the concepts in Figure 1 would be shown as *cat*: * and *mat*: * indicating non-specified entities of types *cat* and *mat*. In the standard canonical formation rules for Conceptual Graphs, unbound concepts are existentially quantified.

A relation may have zero or one incoming arcs, and one or more outgoing arcs. The type of the relation determines the number of arcs allowed on the relation. The arcs always connect a concept to a relation. Arcs cannot exist between concepts, or between relations.

A canon in the sense discussed here is the set of all CGs which are well-formed, and meaningful in their domain. Canonical formation rules specify how CGs can be legally built and guarantee that the resulting CGs satisfy sensibility constraints. The sensibility constraints are rules in the domain which specify how a CG can be built, for example that the concept *eats* must have a theme which is *food*. Note that canonicity does not guarantee validity. A CG may be well-formed in the canonical formation rules for the domain, but still be false.

Sowa discusses his original definitions in [1] but our work follows the further formalized and refined versions of Sowa's original ideas presented by Willems [3], by Chein and Mugnier [4, 5] and by Corbett [6, 7] .

A type hierarchy is established for both the concepts and the relations within a canon. This hierarchy is expressed by a subsumption or generalization-order on types. The formal definitions of Conceptual Graphs are not repeated here, but can be found in [2, 8] and other references.

2. Types and Inheritance

The set of types discussed in the last section is arranged into a type hierarchy, ordered according to the specificity of each type. A type hierarchy is established for both the concepts and the relations within a canon. A type hierarchy is based on the intuition that some types subsume other types, for example, every instance of *cat* would also have all the properties of *mammal*. This hierarchy is expressed by a subsumption or generalization order on types. A type t is said to be more specific than a type s if t specializes some of the concepts from s .

An example of a relation type hierarchy is shown in Figure 2. In one domain that we have worked in, building architecture, we may wish to represent that one structure supports another structure. We may further want to represent that any type of support structure which supports a heavy load will also support a light load. This relationship is expressed in the hierarchy. In this manner, some constraints on the relations between concepts can be represented.

Similarly, an example type hierarchy for concepts is shown in Figure 3. The universal type is shown at the top of the hierarchy, and is represented by \top . The absurd type is shown at the bottom of the graph, and is represented by \perp . Here we see that a support structure is a specialization of a structure, and that a bay structure specializes support structure. Using these type hierarchies, it is possible to show, for example, that the multiple-bay structure will support a heavy load, by using concepts for multiple-bay structure, and a relation of the type supports-heavy.

The definitions of unification, consistency and type subsumption in this paper are based on formal concepts of projection and lower bounds. Carpenter [9] defines each of these operators as a morphism. We have modified Carpenter's definitions to work with the properties of Conceptual Graphs. A morphism is then a mapping from the set of nodes of one Conceptual Graph to the set of nodes of another that preserves the order of relation arguments and the values of those arguments. In a morphism, all of the connections and arguments are preserved.

This definition of projection then gives us a formal definition for subtype and supertype and for subsumption on the partial order of the type hierarchy. All of these operations are now simply applications of the projection operator. Finding types which are compatible (i.e. that can be unified) is now a matter of finding a common subtype (or *join*) between the two types. If the only common subtype is \perp then there can be no unification.

3. Unification as Reasoning

In early pioneering work on the unification of first-order terms, Reynolds [10] used the natural lattice structure of first-order terms, which was a partial ordering based on subsumption of terms [11]. Many terms (or types in our case) are not in any subsumption relation, for example *cat* and *dog*, or *wood* and *mammal*. Unification corresponds to finding the greatest lower bound of two terms in the lattice [12]. The bottom of any lattice, which is represented with the symbol \perp , is the type to which all types can unify, and represents inconsistency. The top of the lattice, represented by \top , is the type to which all pairs of types can generalize, and is called the universal type. Every type is a subtype of \top . Inheritance hierarchies can be seen as lattices that admit unification and generalization [12]. The common specialization of two Conceptual Graphs, s and t , is known as a join. The common generalization of the two graphs is known as a meet.

The process of unifying Conceptual Graphs includes the process of finding the most general subtypes for pairs of types of concepts, which depends on the two types in question being consistent. We also allow constraints on the concepts in the graphs, which are processed during the unification and resolution process. Unification (by projection) is the mechanism we use to find the solution of the constraints. In our work, unification is a tool which performs the work of identifying two structures using subsumption, where the elements of the structure can be constrained.

4. Knowledge Conjunction

Unification is somewhat more complicated, and also more interesting and useful than merely an extension of the join operation. The unification of two graphs contains neither more nor less information than the two graphs being unified. Figure 6 shows that the unification of the two graphs in Figure 5 still retains all the information of the original two graphs. This is the idea behind knowledge conjunction.

The main thrust of the research described in this paper is the unification of Conceptual Graphs in terms of conjoining the knowledge contained in two different graphs. While this may involve term substitution (or the Conceptual Graphs equivalent - instantiation, subsumption, variable binding, etc.) and constraint solving, our research is more concerned with knowledge conjunction. Carpenter defines unification as a system in which two pieces of partial information can be combined into a single unified whole [9]. In our case, these pieces of partial information are represented by Conceptual Graphs. Carpenter refers to this idea as information conjunction, but in our work, it is *knowledge conjunction* that is more important to us. We want to be able to combine the expert knowledge of a system, or even combine knowledge from different sources, not merely gather additional information. Unification here is the combining of pieces of knowledge in a domain, represented as Conceptual Graphs. We define unification as an operation that simultaneously determines the consistency of two pieces of partial or incomplete knowledge, and if they are consistent, combines them into a single result.

When an ontology is represented by the use of Conceptual Graphs constructed in this way, subsumption can be used to combine, refine and reuse the knowledge contained in the graphs. This further allows us to perform reasoning over the knowledge in the graphs as concepts. Reasoning is not limited to objects, classes or libraries, but can also be applied to generic concepts in the knowledge. We demonstrate reasoning over generic concepts in the next section.

One major advantage that Conceptual Graphs have over other representation schemes is that Conceptual Graphs which contain existentially quantified concepts can still be unified. In Feature Structures theory [9] for example, it is important to know whether one is attempting to unify the *intensions* or the *extensions* of two Feature Structures (FS). Essentially, the intension of a Feature Structure is all of the attributes (or properties, or *features*) of a construct. The extension of a Feature Structure is the actual object being represented, with the attributes specified, even if only partially. In Feature Structures theory, one must decide whether the Feature Structures being unified are of the same *intensional* type, or the same *extensional* type, and then seek to identify the two FSs under that type. The unification of two FSs under their extensional type is simply the identification of all their values for their features (similar to type labels and individual markers for the concepts in CGs). There is no way to derive identities of intensional types of two Feature Structures, as there are no values to be compared.

Mineau uses Conceptual Graphs to represent the semantics behind web agents in [13]. Mineau shows that the main advantages of Conceptual Graphs in this regard is that they are highly expressive, formal, easy to use and easy to understand. He shows that the use of CG-based agents as Knowledge Servers increases the interoperability between objects in the ontology. Knowledge conjunction extends this capability by providing a formal, efficient model for reasoning over ontologies.

5. The Air Operations Officer

The results discussed in this section are those recorded from the application of the knowledge conjunction reasoning tool operating over the defense domain. The domain knowledge is represented as Conceptual Graphs with constraints on either the structure of the graph or on the values in the concepts [6]. Here, we discuss the idea behind the reasoning mechanism by employing order sorted unification and constraints within the domain of architectural design. The concepts discussed previously were implemented in Allegro Common Lisp on a Sun Workstation.

An Air Operations Officer (usually known as an OPSO) is the defense officer responsible for deciding the appropriate defensive response to an air threat. A study of the Operations Officer decision-making methods was recently conducted, using a cognitive modeling technique [14, 15]. The study was used to show the usefulness of cognitive modeling in deriving rules from expert knowledge. In this section, we only make use of the rules which resulted from the study; the cognitive modeling technique is not discussed here.

In the domain of the Operations Officer, the magnitude of the response to an air threat is in proportion to the threat itself. So, if the opposing aircraft are very close, or

if the aircraft is of a type which can cause a great deal of damage (known as a *strike* aircraft), then the response is large. If the threat is smaller, then the response is smaller. For example, Figure 7 shows a rule in this domain. (We have borrowed the style of Cao [16] to express the rule, although we do not employ Cao's fuzzy reasoning here.) This graph expresses the rule that if a fighter aircraft (small threat) is between 400 and 500 nautical miles distant, then assert a threat level of alert 60 (the lowest level of alert, in which response fighters must be ready to take off within sixty minutes), and a single fighter is assigned to deal with this threat.

The assertion shown in Figure 6 unifies with the *if* portion of this rule. The *then* portion represents the response to the situation, and it is asserted into the current world knowledge. In this manner, we can represent the decision-making capabilities of the Operations Officer.

The rule shown in Figure 8 is used for a bigger and more impending threat. Any threat aircraft which is closer than 400 nautical miles is considered an immediate threat, and a response squadron must be ready very quickly. Further, a strike aircraft is one which can inflict a great deal of damage, and is therefore dealt with more severely than a fighter aircraft.

The assertion shown in Figure 8 states that a bomber is known to be between 380 and 390 nautical miles distant. Our type hierarchy indicates that a bomber is a type of strike aircraft. Because of the proximity of the threat, the response aircraft are put on alert 10 status. Because of the enormity of the threat, two fighters are assigned to deal with the target aircraft. Again, the assertion unifies with the *if* portion of the

Assertion:

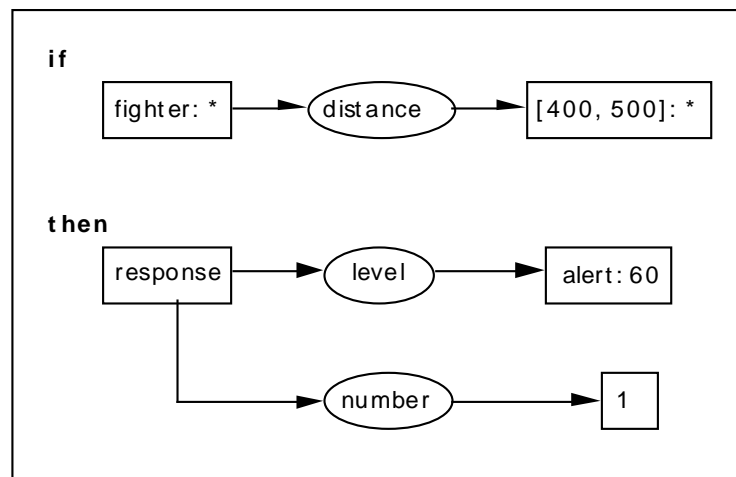
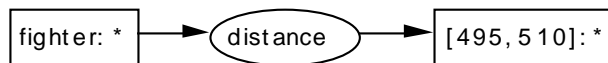


Figure 7. A rule in the defense domain.

Assertion:

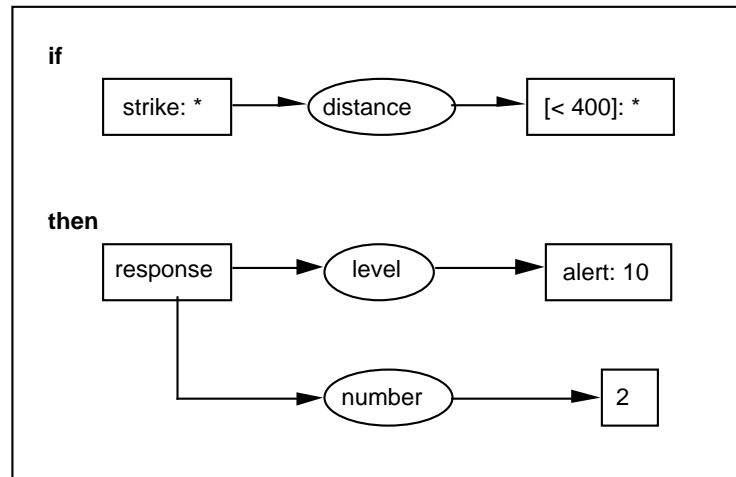
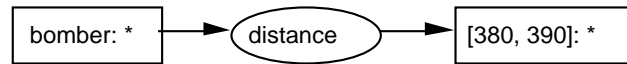


Figure 8. Another rule from the same domain.

rule, causing the **then** portion of the rule to be asserted.

6. Results and Discussion: The Air Operations Officer

Conceptual Graphs and knowledge conjunction can be used to efficiently represent a set of rules in the domain of the Air Operations Officer. The use of Conceptual Graphs is an efficient method for representing the complete ontology of the OPSO, not only in the rules, but also in the exploration and use of the knowledge of types of aircraft and responses. General rules can be represented as Conceptual Graphs, and then specialized dynamically to match the current situation and describe an appropriate response.

7. Architectural Design Tool

The results discussed in this section are those recorded from the application of the knowledge conjunction reasoning tool operating over the domain of architectural design. The point of automated search for the designer is to use computer media that engage designers in exploring design modifications. The design user may want to create new designs, or index, compare or adapt existing designs. This type of user requires efficient representations for the designs and states (of designs) in a symbol system [17]. The designer needs to be able to represent spaces of possibilities which are both relevant to the language and knowledge of design and lend themselves to tractable computations.

Consider a design for the kitchen of a custom-made house. In this design, the architect has specified some of the lighting design and that the floor area must be greater than 20 square meters. The architect has also retrieved an old design, which specifies the remainder of the lighting design. The knowledge conjunction software discussed above combines these two graphs into a single result which represents neither more nor less knowledge than the original graphs. In this graph, all the original knowledge of the first two graphs has been preserved, and the values in the concepts have been joined as specified.

8. Results and Discussion: Architectural Design Tool

Conceptual Graphs can be used to efficiently represent a building design ontology. The use of Conceptual Graphs is an efficient method for representing not only the designs, but also constraints on the designs and knowledge conjunction of designs. The system described in this paper allows general designs to be represented as concepts, and also allows values to be constrained by specifying real-valued constraints as intervals.

The three main areas where the architects want the contribution of Knowledge Conjunction are in type subsumption, knowledge-level reasoning, and pattern matching. First, architects want to be able to use type subsumption to make statements such as, "An office (or kitchen, or corridor) is a kind of room. All the properties which apply to one should apply to its specializations." This is distinct from the object-oriented objective of objects inheriting all the properties of a class of objects. The essential difference is in treating a kitchen as you would any generic room. A generic room can be placed, occupy space, and have attributes like color and number of doors. A *class* of rooms will have attributes, but cannot be said to occupy a space or have specific dimensions, or have a specific count or placement of doors.

The knowledge conjunction model that we developed give this ability to the architects. The algorithm allows the user to specialize designs by matching (unifying) previous designs with the current design problem. Since all characteristics, attributes and constraints are carried along in the unification, the specialization represents all of the design concepts included in the more generic design. Further, and more importantly, there is no real separation between generic and specific, since all points in between can be represented. Conceptual Graphs combined with the ability to specialize using unification are the ideal tool for the knowledge conjunction approach and the constructive nature of architectural design.

The second major concern of architectural designers was the ability to have knowledge-level reasoning. That is, they want to be able to speak in the language of the architect, not the language of the computer (or CAD system). The user wants to be able to refer to the "North Wall" or "Door" without resorting to discussing geometric coordinates in space. The user wants to depart from previous CAD-based data-level processing, and work at the knowledge level in the architecture domain.

This is certainly another area where Conceptual Graphs and unification combine to bring a solution to this domain. While spatial coordinates (and their constraints) can be stored in a graphical representation of a room, there is no need for the user to

bother with using them. The graph can be manipulated as a whole, and treated as a room, rather than a square in a diagram. The completed system will not deal with lines and boxes, but rather with specializing entire designs for rooms (or houses, or office buildings). This approach frees the architect from dealing with data-level concerns of numbers and coordinates, and allows the architect instead to deal with the architectural design.

Finally, the users want to be able to start with a high-level, generic description of a building, and then make queries such as, "Can this bay structure be used in the support structure?" Or, "Do the constraints match up adequately for a particular technology to be used? If yes, tell me the constraints under which it is usable."

Once again, the work presented in this paper meets the requirements of the architects. A query is represented as a Conceptual Graph. The user can specify a type of structure for support, and make the query by attempting to unify the structure with the more generic design. If the unification fails, then the user knows that the proposed structure does not meet the constraints of the design problem. If the graphs unify, then the resulting graph will contain the constraints which must be met in order to make the design work.

Overall, the system of unification over constraints on Conceptual Graphs presented in this paper gives a set of tools to the designer. The ability to use knowledge conjunction with constraints to handle objects at the knowledge level greatly leverages the ability of the designer to work efficiently.

9. Conclusions

We have demonstrated a method for automated reasoning on ontologies, using Conceptual Graphs to represent the underlying ontology. Type hierarchies and the canonical formation rules efficiently specialize graphs into concrete instances. A simple unification operation, using join and type subsumption, is used to perform knowledge conjunction of the concepts represented as graphs. The significance of our work is that the previously static knowledge representation of ontology is now a dynamic, functional reasoning system.

References

1. Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine*. 1984, Reading, Mass: Addison-Wesley.
2. Sowa, J.F., *Conceptual Graphs Summary*, in *Conceptual Structures: Current Research and Practice*. 1992, Ellis Horwood: Chichester, UK.
3. Willems, M. *Projection and Unification for Conceptual Graphs*. in *Proc. Third International Conference on Conceptual Structures*. 1995. Santa Cruz, California, USA: Springer-Verlag.
4. Chein, M. and M.-L. Mugnier, *Conceptual Graphs: Fundamental Notions*. *Revue d'Intelligence Artificielle*, 1992. **6**(4): p. 365-406.

5. Mugnier, M.-L. and M. Chein, *Représenter des Connaissances et Reasonner avec des Graphes*. Revue d'Intelligence Artificielle, 1996. **10**(6): p. 7-56.
6. Corbett, D.R., *Conceptual Graphs with Constrained Reasoning*. Revue d'Intelligence Artificielle, 2001. **15**(1): p. 87-116.
7. Corbett, D.R. and R.F. Woodbury. *Unification over Constraints in Conceptual Graphs*. in *Proc. Seventh International Conference on Conceptual Structures*. 1999. Blacksburg, Virginia, USA: Springer-Verlag.
8. Corbett, D.R. *Reasoning with Conceptual Graphs*. in *Proc. Fourteenth Australian Joint Conference on Artificial Intelligence*. 2001. Adelaide, South Australia: Springer.
9. Carpenter, B., *The Logic of Typed Feature Structures*. 1992, Cambridge: Cambridge University Press.
10. Reynolds, J.C., *Transformational Systems and the Algebraic Structure of Atomic Formulas*. Machine Intelligence, 1970. **5**.
11. Davey, B.A. and H.A. Priestley, *Introduction to Lattices and Order*. 1990, Cambridge: Cambridge University Press.
12. Knight, K., *Unification: A Multidisciplinary Survey*. ACM Computing Surveys, 1989. **21**(1): p. 93-124.
13. Mineau, G. *A First Step Toward the Knowledge Web: Interoperability Issues Among Conceptual Graph Based Software Agents, Part I*. in *Proc. International Conference on Conceptual Structures*. 2002. Borovets, Bulgaria: Springer.
14. Mitchard, H. *Cognitive Model of an Operations Officer*. Honours Thesis, Computer and Information Science, University of South Australia. Adelaide, South Australia, 1998.
15. Mitchard, H., J. Winkles, and D.R. Corbett. *Development and Evaluation of a Cognitive Model of an Air Defence Operations Officer*. in *Proc. Fifth Biennial Conference of the Australasian Cognitive Science Society*. 2000. Adelaide, South Australia.
16. Cao, T.H., P.N. Creasy, and V. Wuwongse. *Fuzzy Unification and Resolution Proof Procedure for Fuzzy Conceptual Graph Programs*. in *Proc. Fifth International Conference on Conceptual Structures*. 1997. Seattle, Washington, USA: Springer-Verlag.
17. Woodbury, R., S. Datta, and A.L. Burrow. *Erasure in Design Space Exploration*. in *Proc. Artificial Intelligence in Design*. 2000. Worcester, Massachusetts, USA.