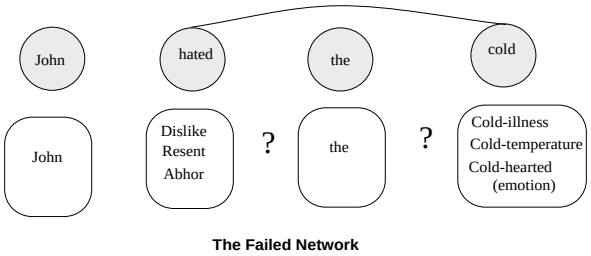


**The Failed Network**

The diagram below shows how the preceeding sentence can be used to disambiguate the word 'cold' in the second sentence.

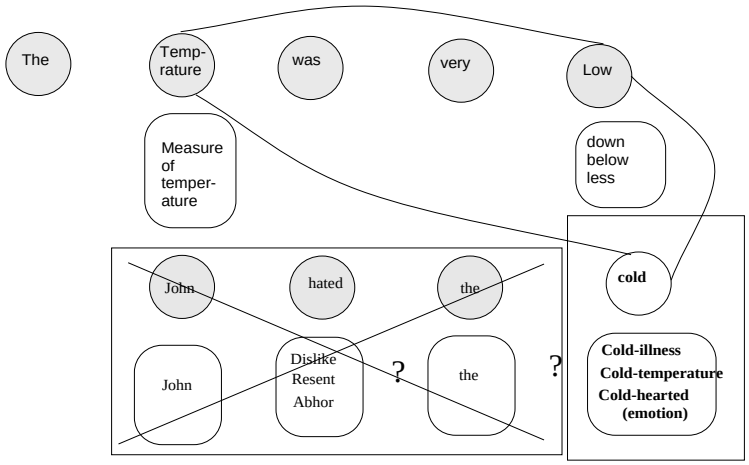


Figure 5: Ensured Disambiguation

[5] Rich, Elaine and Knight, Kevin, *Artificial Intelligence*, McGraw-Hill: New York, 1991.

[6] Small, Steven, "Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding," Dissertation, Technical report TR-954, Department of Computer Science, University of Maryland, College Park, Maryland, 1980.

[7] Small, Steven., Cottrell, Gary and Shastri, Lokendra, "*Toward Connectionist Parsing*, Department of Computer Science, The University of Rochester, Rochester, New York. AAA-82.

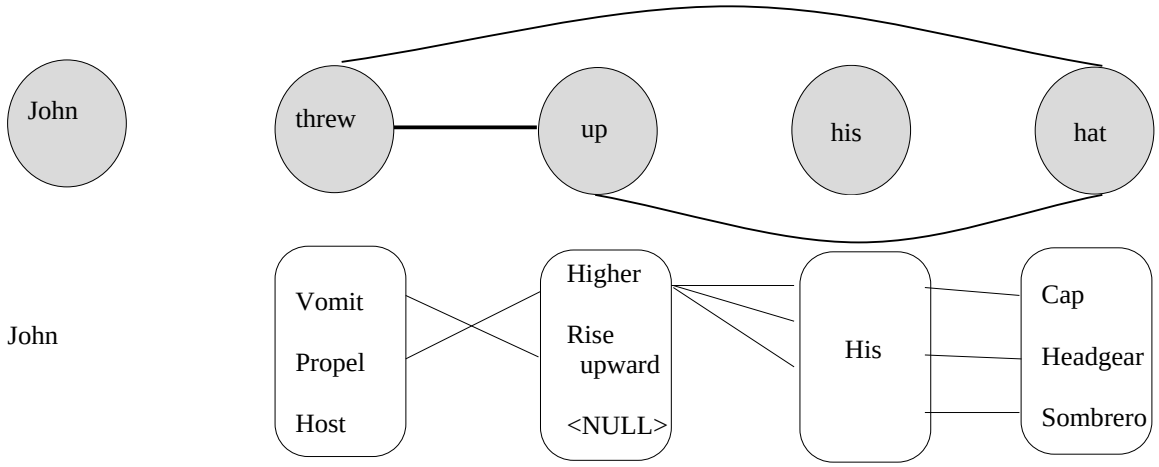


Figure 4: Network for "John threw up his hat""

ature was very low. John hated the cold". The first part of the figure shows the network for the second sentence "John hated the cold". This network cannot disambiguate the meaning of cold by itself. A revised network which includes the first sentence and discards portion of the failed network is shown in the second part of Figure 5.

## 7 Conclusion

We have constructed constraint networks for various sentences and subjected them the various disambiguation processes. It must be noted here that in this approach, the way the data is structured is a partial solution to the problem. To clarify further, the construction of constraints itself played a very crucial role in the search for the correct meaning to a sentence. Constraint satisfaction does provide a reasonable framework in which to view the whole collection of steps that together create a meaning for a sentence [5]. We expect that this approach, if expanded further to include other *constraint types* rather just the *meaning pair constraints*, we would move closer to better semantic text processing. The fact that the implementations worked as predicted is enough reason that the constraint satisfaction method is a feasible approach to word sense disambiguation. Future research will investigate the problem of automatic constraint formulation and ways to include more than one type of constraint into a constraint network. This would also mean finding out what are the other types of constraints which could disambiguate words more effectively. A long term goal in this research would be for fault free machine translation by which we understand text written in one language and generate it in another language.

## References

[1] Berleant, D. and Kaduchak R., "Learn a Language in (almost) Zero Minutes a Day: A Computer Assisted Instruction System," in *Proceedings of the Arkansas Computer Conference*, Little Rock, AR, Oct. 29-30, 1992.

[2] Conrad, James M., Dharma P. Agrawal, and Dennis R. Bahler, "Scalable Static Parallel Arc Consistency Algorithms for Shared Memory Computers," *Proceedings of the Sixth International Parallel Processing Symposium*, Beverly Hills, CA, pp. 242-249, March 1992. Electrical and Computer Engineering, North Carolina State University, Raleigh, 1992.

[3] Mackworth, Alan K., "Consistency in Networks of relations", Artificial Intelligence, Vol. 8, No. 1, pp. 99-118, February 1977.

[4] Mohr, Roger and Thomas C. Henderson, "Arc and Path Consistency Revisited", *Artificial Intelligence*, Vol. 28, No. 2, pp. 225-233, March 1986.

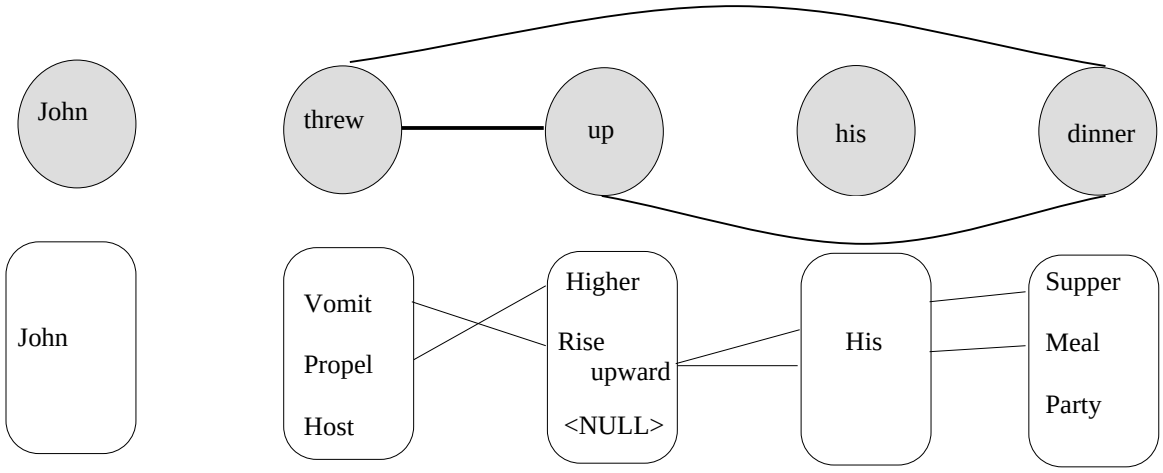


Figure 3: Network for "John threw up his dinner"

## 5 Is the CSP approach workable?

Rich and Knight [5] remark that natural language processing 'could' be viewed as a constraint satisfaction problem. They go further to state that a wide range of constraint types (rather than just formulating constraints between synonyms of words) must be considered, and even then may not work out well. However, they stress that constraint satisfaction does provide a reasonable framework in which to view the whole collection of steps that together create a meaning for a sentence.

This system can prove to be something on the same lines as Small's word expert approach [6] with a different computation method. This method may be more feasible than Steven Small's procedure oriented method. The CSP model forces the meanings of all the words in a sentence to be deduced at the same time. The solution at a node will be the meaning of the word in the context provided by the sentence.

The crux of the problem is on the formulation of constraints, i.e, who will formulate the constraints? Compare this situation with Steven Small's Word Expert Parser which, he says with a sense of defeatism, will require every Word Expert to be hand crafted. The situation is not very different with constraints. At present every constraint has to be hand crafted. A solution to this might be to provide a learning system along with the constraint network processor. The system should be made capable of learning the constraints. This probably will take a long time and will also depend on the amount of text examined. An Automatic Constraint Formulator may be the answer to this dilemma. Automatic construction might be achieved by processing a significant body of "parallel text," in which the same material exists in two languages. This is definitely an open area for research and should be investigated further.

## 6 Failsafe Disambiguation

An extension to the above ideas would be to reinforce constraints by considering the neighboring sentences in the event the constraint database did not possess enough knowledge to disambiguate a word. If an attempt to disambiguate failed, it means that the network yielded more than one solution. That is, though some of the wrong meanings may have been eliminated, more than one meaning still remained. This new extension would require the system to generate a constraint network made up of nodes which will have the 'central ideas' of the neighboring words as domain information for each node. The multiple meanings yielded by the disambiguation attempt will also be included in the new constraint network in one node. The number of nodes will depend on the number of sentences to be taken into consideration plus one, the one being the node for the word which could not be disambiguated but now has a reduced domain probably. This new constraint network will be in effect something like a network made up of networks. An illustration of this idea is given in Figure 5. The word in question is "cold" in the text "The temper-

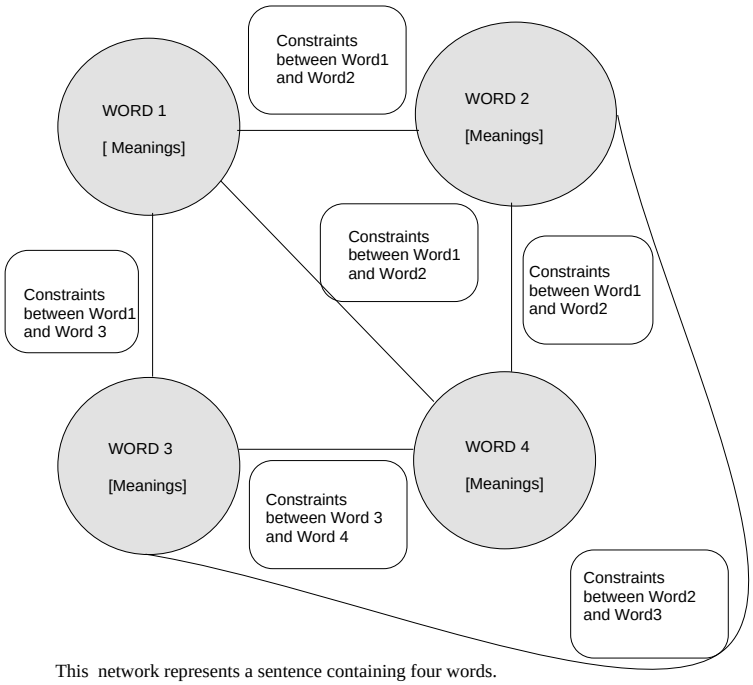


Figure 2: Disambiguation as a Constraint Network

sentence and "propel" for the first and second sentence. This was done by developing a constraint network for each sentence. The constraints were formulated manually by reasoning how the different meanings could be associated with each other.

Networks included the proper noun "John" and the pronoun "his." A NULL constraint was set up between "John"/"his" and all other words. A NULL constraint between two nodes is represented by disconnecting the arc between the two nodes. Another way of setting up null constraints is by allowing the node ("John" or "his") to be assigned the same word it represents as its meaning irrespective of what is assigned to at the neighboring (connected) nodes. However this again gives the problem of having an inconsistent network without solutions. The previous approach to NULL constraints was more workable. The various versions of the constraint networks are illustrated in Figures 3 and 4. The shaded circles represent nodes of the constraints and the curves represent arcs. In the second portion of the figures, the rectangles represent the domain of each node and the values (meanings) each node takes are written inside. The lines between the rectangles show the relations (constraints showns as sets of meaning pairs) between the nodes. For example, if "Propel" is assigned in the "threw" node, then the "up" node has to take the value "Higher".

The program works in three stages. The first stage is where the constraint network information is read in and represented internally as binary matrices. The second is the arc consistency part (using algorithm AC1) and the third is the backtracking tree search. The combination of AC1 along with backtracking produces an effect called dependency-directed backtracking. This is in contrast to the conventional chronological backtracking [5]. In chronological bactracking, in the event of an assignment failing at a node, the procedure dictates that it should back up to the most recent assignment and try an alternative. However with dependency-directed bactracking we avoid instances in the problem space which will cause a failure and thereby also avoid the work that explicitly depended on it. The AC1 process prunes the search space, thus eliminating wrong meanings. This makes the backtracking faster as some of the possible wrong meanings have already been eliminated.