Representing Logical Inference Steps with Digital Circuits

Erika Matsak

Tallinn University, Department of Computer Science, 25 Narva Road, 10120 Tallinn, Estonia
and
Tallinn University of Technology, Department of Computer Engineering, Raja 15, 12618 Tallinn, Estonia
erika.matsak@tlu.ee

Abstract. The use of inference steps in natural language reasoning is observed. An algorithm is presented for representing logically correct inference steps with digital circuits. New foundations for creating decision making systems are studied.

Keywords: Logical inference steps, logic gates, digital circuits representing logical inference steps.

1 Introduction

Many decisions that are required for efficient results with modern systems need to be made without human intervention. For example, driving a Mars rover remotely from Earth is not practical because the sensor information from Mars takes tens of minutes to reach Earth and it takes equally long for the steering commands to reach the rover. Another example would be taking defensive action in case of cyber attacks: a human will not be able to understand the situation and make a (informed) decision in a fraction of a second. Therefore, computers must make these necessary decisions. At the same time we want that the computer-made decisions would be at least as reliable as the one that an intelligent person would make (if he/she would be able to do that).

This brings us to the point that the decision making computer must possess logical instruments: logic formulas (for formulating propositions) and logic inference rules (for constructing an argument). A problem in this case is that a number of different logic systems are in use. For example, classical logic is suitable for operating with legal arguments, while intuitionistic logic [16, 17] can be used for structural program synthesis. One of the ways to distinguish the various logic systems is to use inference rules and the corresponding inference steps. In information technology the inference rules (using inference steps to move between formulae) are implemented in software [1, 2]. However, this does not have to be the only viable option. It is not excluded, in principle, that some of the inference steps could be more efficiently implemented at the hardware level. For this we would first need to develop instruments that allow the separation of logical constructs, such as formulas and inference steps, from natural language [6, 10-14]. We could then proceed to implement these constructs with digital circuits using logic gates [5].

M.J. Smith and G. Salvendy (Eds.): Human Interface, Part II, HCII 2009, LNCS 5618, pp. 178–184, 2009.
© Springer-Verlag Berlin Heidelberg 2009

2 Inference Steps and Implications in Logic Gate Circuits

Let us agree that within this paper we rely on the classic bivalent logic and that we will stay in the confines of first-order predicate calculation. In this case we can use the fact that each correct inference step corresponds to a correct implication, which has the conjunction of the premises of the inference step as an antecedent and the formula of the conclusion of the inference step as a result [6]:

$$\frac{M\ P\ \ldots\ Q}{R} \quad \text{- inference step,} \qquad (1)$$

$$(M\&P\&\ \ldots\ \&Q)\supset R \quad \text{- corresponding implication.}$$

From this point on the implication representing an inference step will be matched with a two-part digital circuit, where the first part (above the dotted line on drawings) represents the conjunction of the premises of the inference step M&P& ... &Q, and the second part represents the formula R.

Note. In bivalent logic the implication can be replaced by the disjunction of the negation of the antecedent and the result (for example, the implication $X\supset Y$ can be replaced with the disjunction $\neg X\vee Y$). We **did not use** such replacements above! Therefore, for example, the *Modus Ponens* inference step is not represented with the formula $\neg(A\&(\neg A\vee B))\vee B$, but with a two part circuit, where the first part represents the formula $A\&(\neg A\vee B)$ and the second part represents the formula B.

The solution described above allows for representing inference steps with traditional digital circuits composed of three types of logic gate elements: negation, conjunction and disjunction. As explained previously, each circuit is divided into two parts, where the first part represents the list of premises and the second part represents the conclusion formula. The *use of the inference step* therefore corresponds to moving from the first part of the circuit to the second part. A separate problem in here is creating such circuits as well as suitable visualization software, which was not as simple as it first appeared.

3 An Algorithm for Using Logic Gates to Design a Digital Circuit That Represents Inference Steps

While designing a digital circuit we assume that as we move from left to right in the formula all signals must have reached the corresponding gates. In order to guarantee this property, we will change the formula (and sub-formulas) as necessary:

- If the formula contains a conjunction that *is not in parentheses* and immediately before or after it are other operations then the conjunction must be surrounded by parentheses.
- For example we replace the formula $A \vee B \,\&\, C \supset D$ with the formula $A \vee (B \,\&\, C) \supset D$
- If the formula contains sub-formulas or their negations then we nest the components from left to right in successive parentheses.
- For example we replace the formula $\neg A \,\&\, B \,\&\, \neg C \,\&\, D$ with the formula $(((\neg A \,\&\, B) \,\&\, \neg C) \,\&\, D)$. We use an analogous process in a formula consisting of only disjunctions.
- If conjunctions (disjunctions) contain sub-formulas of various lengths (including negations or “quantifications” of formulas) then we arrange them from left to right by order of decreasing length (number of symbols).

- For example we replace the formula (Δ ∨ Γ) & (¬A & Γ) ∨ ((A & B) & X) with the formula ((A & B) & X) ∨ (¬A & Γ) & (Δ ∨ Γ).
- We replace implications with applicable formulas consisting of negations, conjunctions and disjunctions. For example we replace the formula X⊃Y with the formula ¬X∨Y.
- If following the rearrangements there is a negation at the right end of the formula then we surround it with parentheses. For example we replace Δ&¬B with Δ&(¬B). If there are two conjunctions or two disjunctions without parentheses at the right end of the formula, then we surround them with parentheses. For example we replace Δ & A & B with Δ & (A & B). Similarly, we replace Δ ∨ A ∨ B with Δ ∨ (A ∨ B).
- The final change is perhaps the most unusual. We write the negation symbol after the formula in question, not before. For example, we replace ¬C with C¬.

The described changes enable the use of the algorithm in Figure 1.

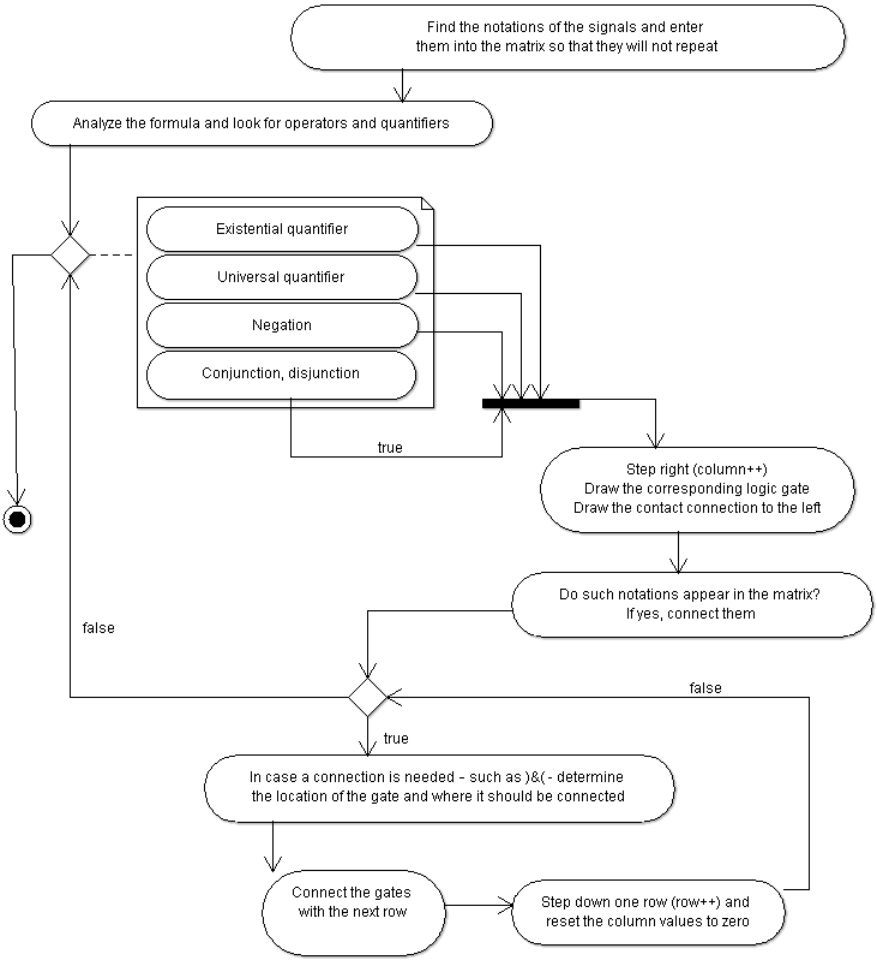

Fig. 1. The algorithm for designing an inference step

Using the algorithm in figure 1 we get the following circuit for the *Modus* Ponens inference step:

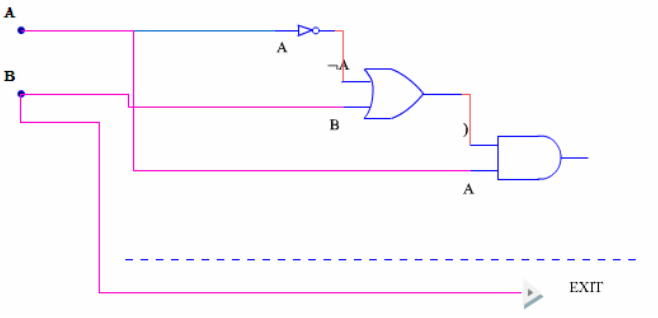

Fig. 2. "Digital" Modus Ponens

By introducing the universal quantifier to the rule [3]

$$\frac{(\ A(\beta)\ \&\ \Gamma\ ) \supset \Delta}{(\ \forall\xi\ A(\xi)\ \&\ \Gamma\ ) \supset \Delta} \tag{2}$$

we get the following digital circuit:

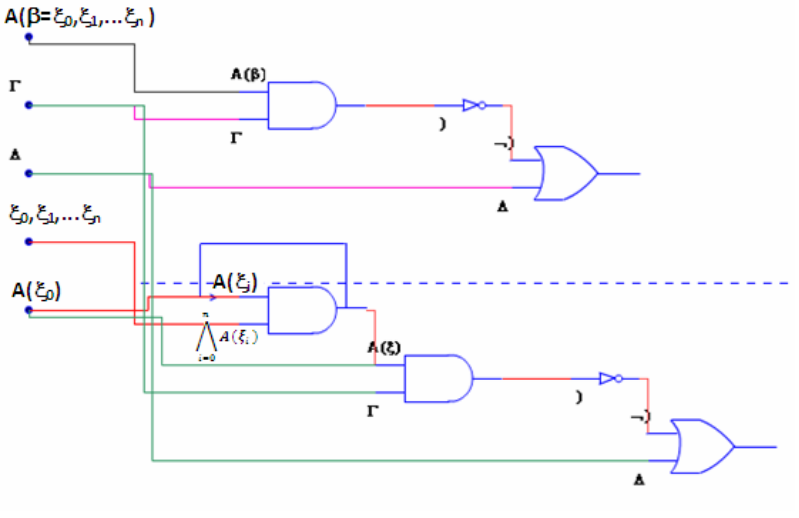

Fig. 3. Digital circuit of the univeral quantifier rule ($\forall^{+}\rightarrow$)

4 Circuits in Practice

The logic module of decision system could consist of the following components:

- a binary matrix of notation-denotation (symbol-meaning) relations, where rows represent notation (symbols, signs) and columns represent denotation (meaning). The intersections contain markers (for example 1 or 0) that indicate that the corresponding notation applies to the denotation (or not). It is important to remember, that according to Lorents [7-9] some notations may have multiple denotations, and some denotations may have multiple notations;
- the set of correct formulas;
- the set of inference rules.

Before implementing the inference steps by using digital circuits, the data in the role of predicates must be inserted. In order to achieve this, the relation between formulas and digital logic gates must be established. Since classically there are two possible truth values 1 and 0 (or true and false) and each logic gate also has two values 1 and 0 (or High Voltage (+5V) and Low Voltage (0V)), then it is natural to connect the gates in a way that correct atomal formulas are represented by the signal "1". Non-atomal formulas should be treated in the following way:

- Identify the part of the circuit that corresponds to the non-atomal formula in question;
- Identify the input points (corresponding to the atomal formulas) for that specific circuit part;
- Identify an input signal combination for the above input gates that produces "1" as an output for that circuit part.

This way we can provide the necessary input signals to the (upper) part of digital circuits, which corresponds to the predicates of the inference step.

The construction of decision may take the simple form of "fitting" puzzle pieces, where the "suitable" premise set of an inference rule allows the rule to be matched to a combination of existing formulas that normally do not involve more than a few formulas.

5 Advantages of the Proposed Circuits

While creating decision making systems (that are based on, for example, binary decision diagrams (BDD), negation normal form (NNF), propositional directed asyclic graph (PDAG), etc.) data structures related to Boolean functions are often used. The logical operations used to form decisions are simple: AND, OR, NOT. In recent years, several problems have surfaced in solutions relying on neural networks or graphs. This does not mean that these methods should be cast aside (for example, neural networks have advantages in modeling non-linear characteristics of sample data – [4]). However, systems based on implementing inference steps with digital circuits also have advantages. One source of these advantages is the ability to include "regular" operations (AND, OR, NOT), as well as other operations (implication, etc.) and quantifiers. Second and more important advantage is the possibility to notably

"shorten" the decision making process (it is well known from logic studies that manipulating with the rules may sometimes allow an exponential (!) decrease in the number of inference steps).

6 Conclusion

The described digital circuits consisting of logic gates are not the only way to represent inference steps. In principle, using special transformations one could implement them in neural networks [15] or other circuits. The important part here is how to implement logical inference steps in hardware based on the logical constructs extracted from natural language. It is possible that a similar implementation is present in the human brain, which allows us to use logical constructs, including the ability to formulate propositions and to come up with the correct conclusion.

Acknowledgements. The author of the given work expresses profound gratitude to professor Peeter Lorents for assistance in a writing of given clause and to Rain Ottis for English version edition.

References

1. Chang, C., Lee, R.: Symbolic logic and mechanical theorem proving. Academic Press, New York (1973)
2. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Springer, Heidelberg (1996)
3. Gentzen, G.: Die Widerspruchsfreiheit der reinen Zahlentheorie. Mathematische Annalen 112, 493–565 (1936)
4. Kim, D., Lee, J.: Rule Reduction over Numerical Attributes in Decision Trees Using Multilayer Perceptron. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS, vol. 2035, p. 538. Springer, Heidelberg (2001)
5. Kunz, W., Stoffel, D.: Reasoning in Boolean Networks: Logic Synthesis and Verification Using Testin Techniques. Kluwer Academic Publishers, Dordrecht (1997)
6. Lorents, P.: Language and logic. EBS Print, Tallinn (2000)
7. Lorents, P.: Formalization of data and knowledge based on the fundamental notation-denotation relation. In: Proceedings of the International Conference on Artificial Intelligence, ICAI 2001, vol. III, pp. 1297–1301 (2001)
8. Lorents, P.: Knowledge and understanding. In: Proceedings of the International Conference on Artificial Intelligence, ICAI 2004, vol. I, pp. 333–337 (2004)
9. Lorents, P.: Taxonomy of intellect. In: Proceedings of the International Conference on Artificial Intelligence, ICAI 2008, vol. II, pp. 537–544 (2008)
10. Matsak, E.: Dialogue system for extracting Logic constructions in natural language texts. In: Proceedings of the International Conference on Artificial Intelligence, ICAI 2005, vol. II, pp. 791–797 (2005)
11. Matsak, E.: Using Natural Language Dialog System DST for Discovery of Logical Constructions of Children's Speech. In: The 2006 International Conference on Artificial Intelligence, ICAI 2006, Las Vegas, Nevada, USA (2006)

12. Matsak, E.: System DST for Transforming Natural Language Texts, Representing Estimates and Higher Order Predicates and Functionals. In: The 3rd International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2006, Orlando, Florida, USA (2006)
13. Matsak, E.: The prototype of system for discovering of inference rules. In: Proceedings of the International Conference on Artificial Intelligence. International Conference on Artificial Intelligence, ICAI 2007, vol. II, pp. 489–492 (2007)
14. Matsak, E.: Improved version of the Natural Language Dialog System DST and its application for discovery of logical constructions in children's speech. In: International Conference on Artificial Intelligence, ICAI 2008, vol. I, pp. 332–338 (2008)
15. Minsky, M.: Finite and Infinite machines. Prentice-Hall, Inc., Englewood Cliffs (1967)
16. Mints, G., Tyugu, E.: Justification of the structural synthesis of programs. Science of computer programming 2(3), 215–240 (1982)
17. Mints, G., Tyugu, E.: The programming system PRIZ. Journal of Symbolic Computation (4) (1987)