Finding Tautology*

Lokman Koluksa

June 19, 2007

Contents

*Kullanlan terimler iin makalenin sonuna baknz

1 Summary

In this section we will summarize the study and try to explain key points.

1.1 Statement Formulas

Consider the following theorem.

Theorem 1 *$f(x_1,..,x_i,..,x_n)$ is a tautology iff $f(x_1,..,0,..,x_n)\cdot i(x_1,..,1,..,x_n)$ is a tautology.*

According to this theorem, one can eliminate all the occurrences of a variable and obtain another formula which has the same tautological value. This is called reduction and shown as below.

$$f(x_1..x_i..x_n) \overset{x_i}{\rightarrow} f(x1..0..x_n)\cdot f(x_1..1..x_n)$$

So all the variables of a formula can be eliminated as follows.

$$\begin{aligned} f_0(x_1,..,x_n) &\overset{x_1}{\rightarrow} f_1(x_2,..,x_n)\\ f_1(x_2,..,x_n) &\overset{x_2}{\rightarrow} f_2(x_3,..,x_n)\\ &\vdots\\ f_{n-1}(x_n) &\overset{x_n}{\rightarrow} f_n() \end{aligned}$$

After reducing all the variables and simplifying it, only 0 or 1 remains. Since the tautological properties of the formulas are preserved with this operation, if 1 remains the original formula is a tautology and if 0 remains the original formula is not a tautology.

In the 2.section of the previous paper, how to do this process by means of other ways are investigated and at last the two dimensional forms of the formulas are obtained.

1.2 Two dimensional Formulas

Here the conversion from the statement formulas to two dimensional ones and the properties of two dimensional formulas are investigated. Also there are some definitions. For conversion, negation must be applied to variables only in the statement formula. In that case, the conversion is easy and an example is shown in the figure 1.

Two dimensional formulas consist of arces, two type of nodes like • and ⌓ and in the form of a tree whose leaves are literals or variables. The first type of node is AND node and the second one is called OR node. The OR nodes have a main arc and some base arches. As seen, two dimensional formulas are actually AND-OR graphs with extra features.

An interpretation of the given formula and the value of each arc are shown in the figure 2. Naturally arches have directions. The value of an arc connected to a variable is equal to the value of the variable. If the value of one of the outgoing base arches of an OR node is 1 then the incoming main arc takes the value 1. If the values of all the outgoing base arches are all 0 then the incoming main arc becomes 0. The incoming base arches takes the value of outgoing main arc. If one of the outgoing arches of an AND node has the value 0 then the incoming arc takes the value 0. Otherwise

$x_1 \cdot x_2 \cdot \widetilde{x_3} \vee x_3 \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee x_3 \cdot \widetilde{x_2}) \vee x_2 \vee \widetilde{x_3} \cdot x_4$

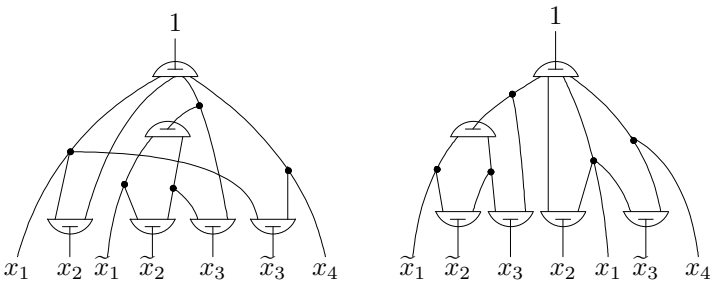


Figure 1: Two dimensional correspondence of a statement formula and the simplified one

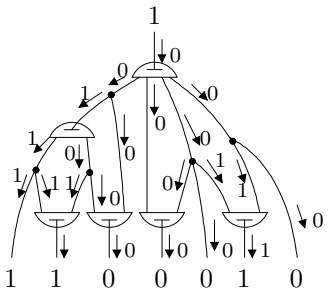


Figure 2: The values of the arches for the interpretation $x_1 = x_2 = x_3 = x_4 = 0$

it becomes 1. The root arc becomes 0 for this interpretation. This means that the value of the formula for this interpretation is 0.

Each arc represents a formula and this is the formula which affects the value of this arc. If a literal can affect the value of an arc, then there is a connection from the arc to that literal. I.e. the formula of an arc is the sub formula which is found by following the paths beginning from that arc.

The reduction of variables in two dimensional forms are done as in the figure 3. If both x_i and its negation exists then an arc connecting the symbols x_i and $\widetilde{x}_i$ are put and the symbols x_i and $\widetilde{x}_i$ are erased. If there is only x_i or $\widetilde{x}_i$ then a 0 is put instead.

Here the first formula is called the initial formula and the last formula is called totally reduced form.

After this operations what we get at hand is a pure graph and whether the original formula is a tautology or not should be determined by examining this form. The question here is "what properties of such a form distinguish a tautology from a nontatology?"

To get an idea about such properties, look at the formulas and their totally reduced forms given in the figure 4.

As seen in the first formula there is a closed path. The second formula is a tautology and there is no closed path. Thus closed paths should indicate nontautology.

In any formula the inconsistent e-products does not affect the tautological property of the formula. Then in any formula if the formula formed from the consistent e-products is a tautology the the original formula is a tautology. Otherwise the original formula is not a tautology.

Thus if all the e-products of a totally reduced formula are found to be inconsistent, then we can decide that the original formula is not a tautology. However, before applying this idea we must answer what is an e-product and how to find them in a two dimensional formula.

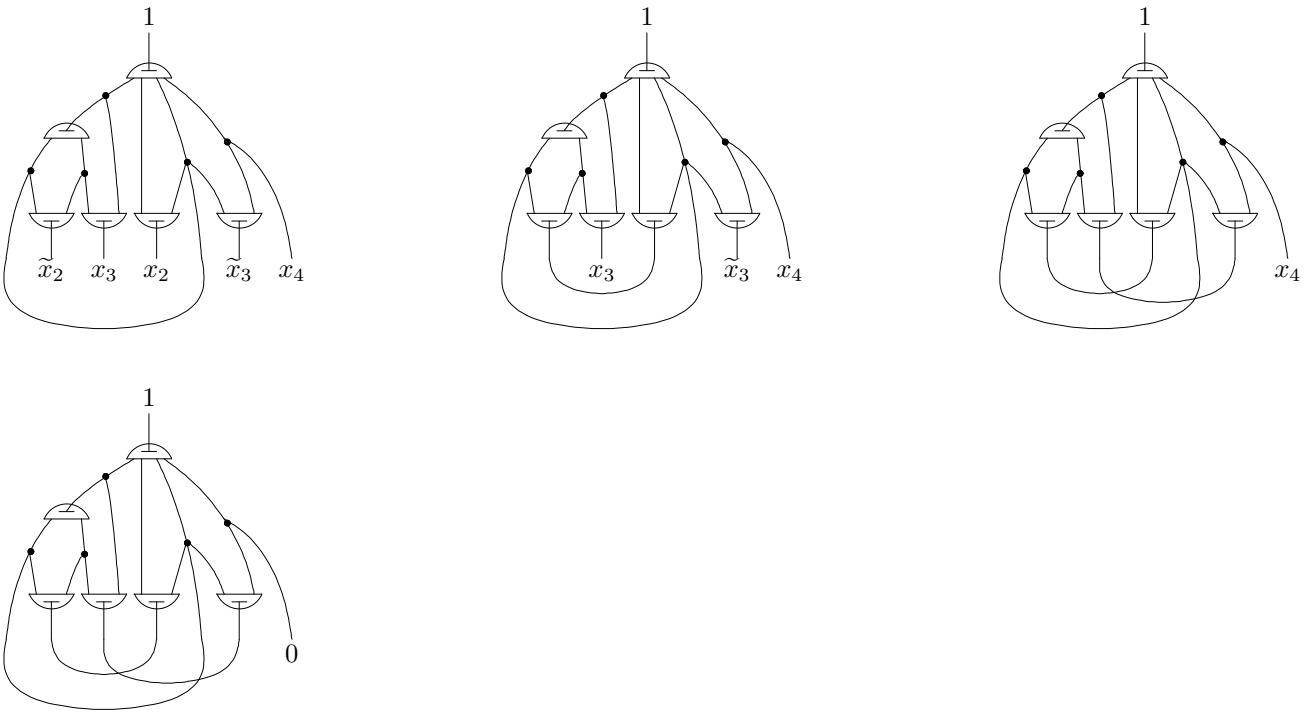
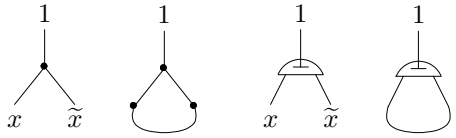


Figure 3: Reduction of the sample formula for x_1, x_2, x_3 and x_4



(a) A nontautology (b) A tautology

Figure 4: The initial and totally reduced forms of two formulas

1.2.1 E-products and Two Dimensional Formulas

Finding E-Products in Initial Formulas The following is an algorithm to find an e-product in an initial formula.

Algorithm 1.1 *This algorithm finds an e-product beginning from the arc p.*

1. *Push p to a stack. Repeat the followings until the stack is empty.*
   (a) *Pop an arc from the stack. Let this be p.*
   (b) *If p is connected to an OR node, i.e. if in the form* $p \dashv (r_1, .., r_n)$ *then push* r_i *for one and only one* $i = 1, .., n$.
   (c) *If p is in the form* $p \bullet (dizir)$ *then push* r_i *for every* $i = 1, .., n$.
   (d) *If p is in the form* $p \vdash r$ *then push r.*
   (e) *If p is connected to a literal or variable then don't do anything.*

2. The graph thus scanned is an elementary product.

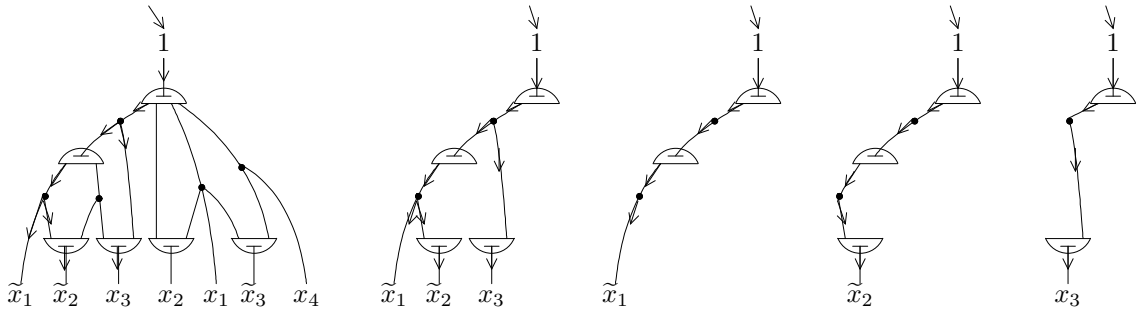


Figure 5: The scanning of an e-product, the e-product found, and the paths of it

To find the value of an e-product the following assumptions are made.

1. The value of an e-product is the conjunction of the values of all the paths of it.

2. The value of a path is equal the value of its terminal.

For example the value of the e-product shown in the figure is found to be $x_1 \cdot \widetilde{x}_2 \cdot \widetilde{x}_3$. Later it is shown that the value of an initial formula is equal to the disjunction of the values of its e-products.

The given e-product scanning algorithm can be made recursive as follows.

Algorithm 1.2 *This algoritm scans an e-product beginning from the arc p. $EPS(p)$ represents the e-product thus scanned.*

1. Do the followings beginning from p.

(a) If p is in the form px then $EPS(p) = px$ where x is either a variable or literal.

(b) If p is in the form $p\bullet(r_1, .., r_n)$ then apply this algorithm for each $r_1, .., r_n$. Then $EPS(p) = p \bullet \{EPS(r_1), .., EPS(r_n)\}$.

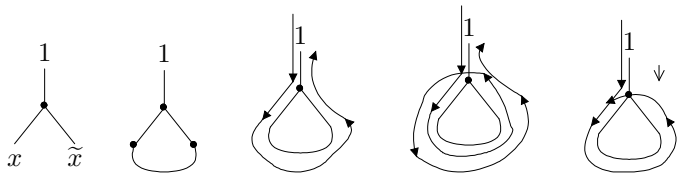
(c) If p is in the form $p \dashv (r_1, .., r_n)$ apply this algorithm for one and only one r_i. Then $EPS(p) = p \dashv EPS(r_i)$.

By using these relations, it is shown that the disjunction of the values of the e-products of an initial formula is equal to the value of the formula. Thus validity of the given algorithm is verified.

Later how to find e-products in reduced forms are investigated and it is decided to apply the given algorithm without any modification.

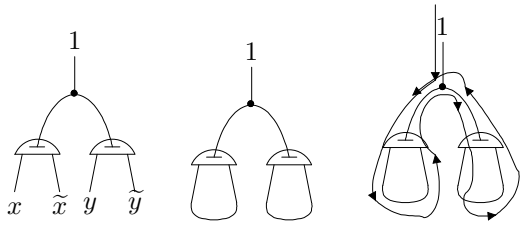
However, by reduction, some infinite paths are generated and troubles begin. We have defined that the value of a path is equal to the value of its terminal, but the infinite paths have no terminals. So this definition does not work for them. The infinite paths can occur in any formula and e-product. That is the value of an infinite path can not be determined by the literals or the variables of the formula or the e-product it takes place. So their values must be either 0 or 1.

To determine the values of them lets investigate some examples. The first example is the statement formula $x \cdot \widetilde{x}$. Obviously it is not a tautology. Thus the value of totally reduced form must be 0. The initial and the last reduced form is shown in the figure 1.2.1.



(a) (b) Some paths terminating with 1 (c) An infinite path

Figure 6: Reducing the formula and some paths thus generated



(a) (b) (c)

Figure 7: In all the e-products there is an infinite path

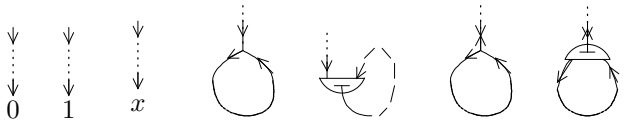
After reduction, only one e-product is generated. All the paths of this e-product are either terminates with 1 or infinite. Obviously, the value of these infinite paths must be 0 otherwise the reduction gives the wrong result.

On the other hand let look at the example in the figure 7.

The statement correspondence of the initial formula is $x \vee \widetilde{x}) \cdot (y \vee \widetilde{y})$ and obviously a tautology.Thus the value of last(totally) reduced form must be 1. However, in the last reduced form, every e-product has an infinite path. So to make the result correct the value of these infinite paths must be taken as 1.

At first look, these are confusing results. However, it is not difficult to observe that there is a difference between the infinite paths in the first example and the infinite paths in the second example. The paths in the second example is in the form $..r..r'..$ where r is an arc, however, the paths in the second example are not of this type.

So we can classify the paths as follows.



(a) Finite paths (b) Cyclic paths (c) Returning paths

Figure 8: Type of paths

Definition 1 *1. If a path is in the form ..p..p′.. then it is a returning or bidirectional path. If a path is not returning then it is a unidirectional path.*

2. If a path is in the form ..p..p.. then it is a cyclic path. If a path is not cyclic then it is acyclic.

A path can be both returning and cyclic. For example $..p..r..r'..p..$ is both returning and cyclic. Then we classify the infinite paths as follows.

1. unidirectional infinite paths
2. bidirectional infinite paths

If $\underline{A}$ is a unidirectional infinite path then indicate this as $\underset{\rightarrow}{A}^{\infty}$. However, the returning infinite paths can be separated into two class: can be written as $\underset{\leftrightarrow}{A}\underset{\rightarrow}{B}^{\infty}$ where A is bidirectional and those which can not be written so. Call the letters as true bidirectional infinite paths and show with $\underset{\leftrightarrow}{A}^{\infty}$. For example $(\underset{\leftrightarrow}{A})^{\infty} = \underset{\leftrightarrow\leftrightarrow}{AA}..$ is a true bidirectional infinite path where $\underset{\leftrightarrow}{A}$ is a bidirectional path.

Now we can make the following assumptions.

1. The value of any formula is equal to the disjunction of the values of its e-products.
2. The value of any e-product is equal to the conjunction of the values of its paths.
3. The value of any path is equal to the value of its tail.
4. The value of any unidirectional infinite path or a path whose tail is a unidirectional infinite path is 0.
5. The value of any true bidirectional infinite path is 1.

Then we need to investigate the general properties of e-products and formulas. So we need to abbreviate and symbolize them. After achieving this, finding the value of e-products in reduced formulas and their relation to the e-products before reduction are investigated. At last a general algorithm for scanning the e-products is introduced to solve this challenging subject.

The difference between the normal algorithm and the general one is that in general algorithms a new type of paths in the form $..pp'..$ is introduced. The value of such infinite paths is taken as 1, otherwise the interpretation of the e-products would be wrong.

Actually, the general algorithm is not a necessary part of the study. Without it, proving the properties of two dimensional forms is still possible, however it greatly simplifies this job.

In this subsection, the following proposition is proved.

1. If the value of any bidirectional path (finite or infinite) is taken as 1, the value of the formula does not change.

However, this subsection is far from complete yet.

1.2.2 General Form and Reduction

This section is not complete and there are many missing parts.

Here how to find the value of any two dimensional formula is investigated and the following proposition is proved.

Proposition 1 *Let $TD(x_i)$ represents a two dimensional formula and $S(x_i)$ represents its statement equivalence. I.e. $TD(x_i) \cong S(x_i)$. Also let $TD(x_i) \overset{x_i}{\rightarrow} TD()$ and $S(x_i) \overset{x_i}{\rightarrow} S()$. Then $TD() \cong S()$.*

In a totally reduced form there is no variable and all terminals are either 0 or 1. In that case the followings will be true as a result of the assumptions.

1. If all the paths of an e-product are either terminates with 1 or bidirectional then this is a valid e-product. If a path of an e-product is either terminates with 0 or unidirectional infinite then this is an invalid e-product.

2. If at least one of the e-products of a formula is valid then the initial formula is a tautology otherwise it is not a tautology.

On the other hand the value of a formula is equal to the conjunction of the values of its e-sums. The following a scanning algorithm to find an e-sum in an initial two dimensional formula.

Algorithm 1.3 *1. This algorithm find an e-sum from the arc p.*

2. Push p to the stack. Repeat the followings until the stack is empty.

   (a) Pop an arc from the stack. Let this be p.

   (b) If p is connected to an OR node, i.e. if in the form $p \dashv (r_1, .., r_n)$ then push r_i for every $i = 1, .., n$.

   (c) If p is in the form $p \bullet (dizir)$ then push r_i for one and only one $i = 1, .., n$.

   (d) If p is in the form $p \vdash r$ or pr then push r.

   (e) If p is connected to a literal or variables then don't do anything.

3. The graph thus scanned is an elementary sum.

This algorithm can be made recursive as follows.

Algorithm 1.4 *The algorithm ESS(p):*
This algorithm scans an e-sum from the arc p. ESS(p) represents the e-product thus scanned.

1. Do the followings beginning from p.

   (a) If p is in the form px then $ESS(p) = px$ where x is either a variable or literal.

   (b) If p is in the form $p \bullet (r_1, .., r_n)$ then apply this algorithm for one and only one r_i. Then $ESS(p) = p \bullet ESS(r_i)$.

   (c) If p is in the form $p \dashv (r_1, .., r_n)$ apply this algorithm for every $r_1, .., r_n$. Then $ESS(p) = p \dashv \{ESS(r_1), .., ESS(r_n)\}$.

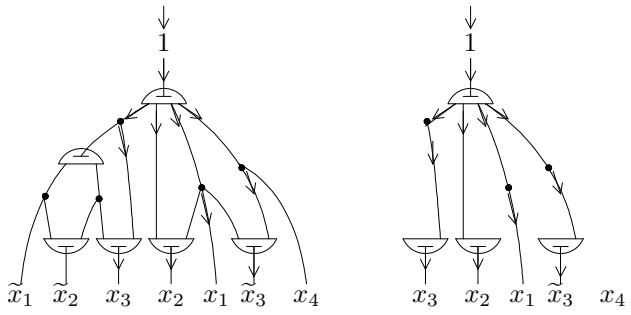


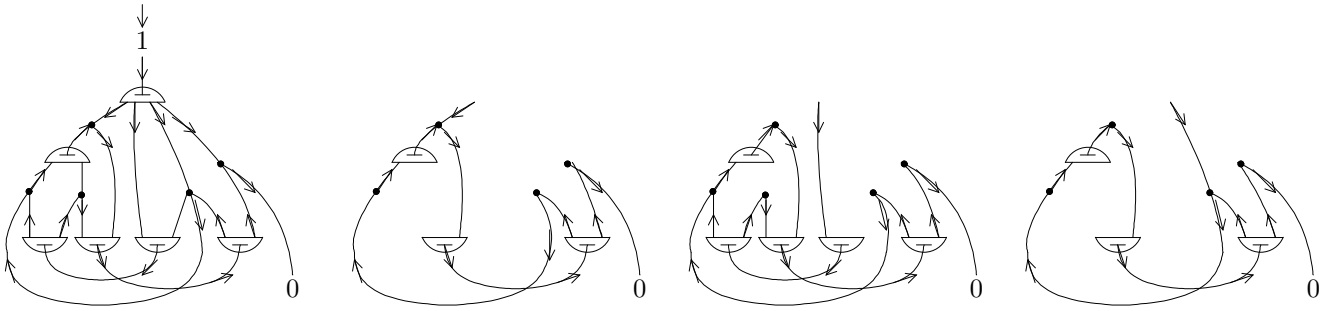
Figure 9: The scanning of an e-sum and the e-sum found

The figure 9 illustrates the scanning of an e-sum.

Naturally the value of an e-sum is the disjunction of the values of its paths. Then the followings would be true.

1. If all the paths of an e-sum are either terminates with 0 or unidirectional infinite then this is an invalid e-sum.
2. If any path of an e-sum is either terminates with 1 or bidirectional then this is a valid e-sum.
3. If at least one of the e-sums of a formula is invalid then the initial formula is not a tautology otherwise it is a tautology.

For example in the figure 10, an invalid e-sum is shown.



(a) An invalid e-sum

(b) All the paths of this e-sum is either terminates with 0 or has an unidirectional infinite path as a tail

Figure 10: There is an invalid e-sum in this formula

Another example is shown in the figure 11 and 12.

The tautological value of arches Every arc represents a formula beginning from it. So each arc has a tautological value.

Definition 2 *An arc is a tautology iff there is at least one valid e-product beginning from it or all the e-sums are valid.*

$x \cdot y \vee x \cdot \widetilde{y} \vee \widetilde{x} \cdot y \vee \widetilde{x} \cdot \widetilde{y}$

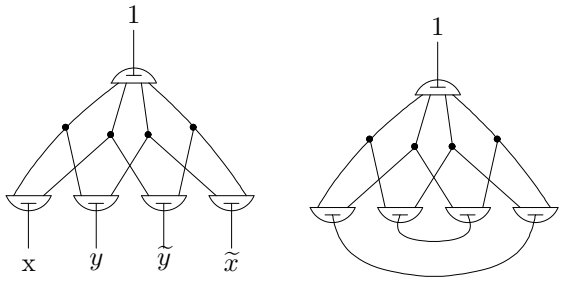
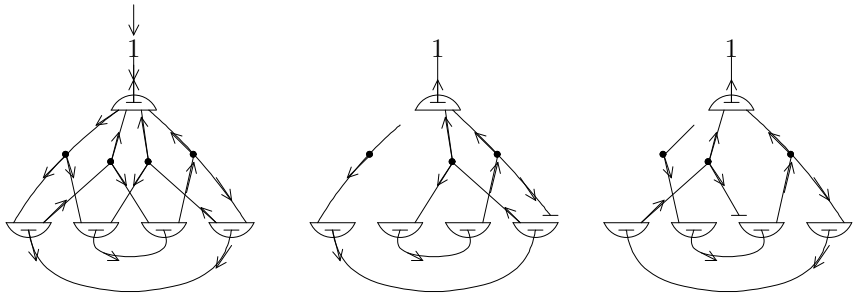


Figure 11: A tautology and its last reduced form



(a) A valid e-product

(b) All the paths of this e-product are either terminates with 0 or bidirectional

Figure 12: There is a valid e-product in this formula

The tautological values of arches are not independent from each other. The following is the rules of these dependencies.

Proposition 2 *Let $T(p)$ indicate that the formula beginning from p is a tautology.*

1. *For $p \vdash r$ or pr, $T(p) \leftrightarrow T(r)$.*
2. *For $p \bullet (r_1..r_n)$ $T(p) \leftrightarrow T(r_1) \cdot .. \cdot T(r_n)$ olur.*
3. *For $p \dashv (r_1, .., r_n)$, $T(r_1) \vee .. \vee T(r_n) \rightarrow T(p)$ and $T(p) \cdot T(p') \rightarrow T(r_1) \vee .. \vee T(r_n)$.*

However to make proofs(and life) easier we will prove the followings only.

Proposition 3 1. *For $p \vdash r$ or pr, $T(r) \rightarrow T(p)$.*
2. *For $p \bullet (r_1..r_n)$, $T(r_1) \cdot .. \cdot T(r_n) \rightarrow T(p)$.*
3. *For $p \dashv (r_1, .., r_n)$, $T(r_1) \vee .. \vee T(r_n) \rightarrow T(p)$.*

Proof 1 *1. For $p \vdash r$ let the e-products of r be $r\{\lambda r_1, .., \lambda r_n\}$. Then the e-products of p would be $p \vdash r\{\lambda r_1, .., \lambda r_n\} = \{p \vdash r\lambda r_1, .., p \vdash r\lambda r_n\}$. Assume r is a tautology. This means that there is an e-product of r whose all paths are either returning or terminates with 1. Let this e-product be $r\lambda r_i = \{r\underline{A}_1, .., r\underline{A}_m\}$ where $r\underline{A}_1, .., r\underline{A}_m$ are the paths of it. This means that for any j,*

$r\underline{A}_j$ is either in the form $r..s..s'..$ or in the form $r..1$. However, then, there is an e-product $p \vdash r\lambda r_i = \{p \vdash r\underline{A}_1, .., p \vdash r\underline{A}_m\}$ whose all the paths are either in the form $p \vdash r..s..s'..$ or in the form $p \vdash r..1$. So this e-product is valid and then p is a tautology.

2. *For $p \bullet (r_1..r_n)$ let $T(r_i)$ be true for every $i = 1, .., n$. That is let there is at least one valid e-product of r_i for every $i = 1, .., n$ r_i. Let a valid e-product for r_i be $r_i\lambda r_{ij}$. Then one of the e-products of p be $p \bullet (r_1\lambda r_{1j}, .., r_n\lambda r_{nj}$ and all the paths of it would be either returning or terminate with 1. So p is a tautology.*

3. *For $p \dashv (r_1, .., r_n)$, assume $T(r_i)$ is true for one $i = 1, .., n$. Thus there is an e-product starting from r_i. Let this be $r_i\lambda r_{ij}$. Then one of the e-products of p would be $p \dashv r_i\lambda r_{ij}$ and all the paths of it would be either returning or terminate with 1. So p is a tautology.*

■

Now let examine a more complex example by using these relations. The initial formula and its last reduced form is shown in the figure 13

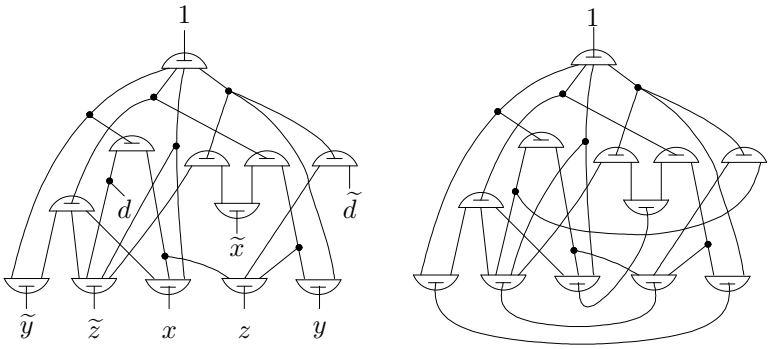


Figure 13: A tautology and its last reduced form

Let reduce the statement formula.

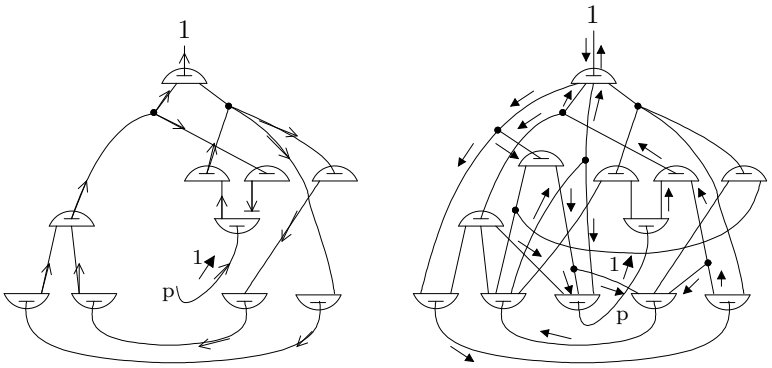
$$
\begin{aligned}
&= \widetilde{y} \cdot (\widetilde{z} \cdot d \vee x \cdot z) \vee (\widetilde{x} \vee y \cdot z) \cdot (\widetilde{y} \vee \widetilde{z} \vee x) \vee x \cdot \widetilde{z} \vee y \cdot (z \vee \widetilde{d}) \cdot (\widetilde{x} \vee \widetilde{z}) \\
&\xrightarrow{y} ((\widetilde{z} \cdot d \vee x \cdot z) \vee \widetilde{x} \vee x \cdot \widetilde{z}) \cdot ((\widetilde{x} \vee z) \cdot (\widetilde{z} \vee x) \vee x \cdot \widetilde{z} \vee (z \vee \widetilde{d}) \cdot (\widetilde{x} \vee \widetilde{z})) \\
&= (\widetilde{x} \vee z) \cdot (\widetilde{z} \vee x) \vee x \cdot \widetilde{z} \vee (z \vee \widetilde{d}) \cdot (\widetilde{x} \vee \widetilde{z}) \\
&\xrightarrow{z} (\widetilde{x} \vee x \vee \widetilde{d}) \cdot (x \vee \widetilde{x}) \\
&= 1
\end{aligned}
$$

The last reduced form of the statement formula is 1. I.e. the initial formula is a tautology. Then there must be a valid e-product in the totally reduced two dimensional form. Finding this in the graph is not easy, however the tautological relationship among the arches can help us as shown in the figure 14.

2 Finding Tautology

2.1 A class of Algorithms

In this section, it is assumed that all the e-product and e-sum scans are done with normal algorithm.



(a) There is a valid e-product starting from p. All the paths are either terminates with the top 1 or includes p'. Since p is a tautology, it is marked with a 1

(b) the arches which are tautology because p and the reverse root arc are tautology are marked according to the given rules

Figure 14: The root arc is marked. This means that there is an e-product whose all paths are either bidirectional or terminate with a 1

Here we will introduce a class of algorithms which gives order to the arches. We will begin with the general properties of the algorithms.

Let the arches take order according to the following rules where $O(p)$ shows the order of arc p.

1. For $p \bullet (r_1, .., r_n)$, $O(p) \geq \min\{O(r_1), .., O(r_n)\}$.
2. For $p \dashv (r_1, .., r_n)$, let $O(p) \geq \max\{O(r_1), .., O(r_n)\}$.
3. For $p \vdash r$, $O(p) \geq O(r)$.
4. For $p0$, $O(p) \geq O(0)$.
5. For $p1$, $O(p) \geq O(1)$.

By this way, starting from any arc, there will be an e-sum whose order is monotonically decreases or remains same through the paths. The following is an algorithm to find such an e-sum.

Algorithm 2.1 *1. Initially $BUF = \{p\}$. Repeat the followings until BUF is empty.*

(a) Get an arc from BUF. Let this be p.

(b) For $p \dashv (r_1, .., r_n)$, $O(p) \geq O(r_i)$ for every $i = 1, .., n$ by the rules. So put every r_i to the buffer.

(c) For $p \bullet (r_1, .., r_n)$, $O(p) \geq min\{O(r_1), .., O(r_n)\}$. So for at least one $i = 1, .., n$, $O(p) \geq O(r_i)$. Put this r_i to BUF.

(d) For $p \vdash r$, $O(p) \geq O(r)$. Put r to BUF.

(e) For $p0$ and $p1$, $O(p) \geq O(0)$ and $O(p) \geq O(1)$ respectively. Do nothing

2. *The graph scanned is an e-sum whose order is monotonically decreases or remains same through the paths.*

Let call this as monotonically decreasing e-sum or MD e-sum. The monotonically decreasing paths are also called MD paths.

Let $O(\underline{A}) = L$ describes that every arc on $\underline{A}$ has the same order and equal to L.

In this section, instead of $\Rightarrow\underline{A}$, often $\underline{A}$ is used.

Theorem 2 *If a MD path is cyclic, i.e. in the form* $p..r\underline{A}r..$ *then* $O(r\underline{A}r) = O(r)$.

Proof 2 *Let* s *be any arc in* $\underline{A}$. *Since* $p..r..s..r..$ *is a monotonically decreasing path* $O(r) \geq O(s)$ *and* $O(s) \geq O(r)$. *This is possible only if* $O(s) = O(r)$. ■

Definition 3 1. *If for an arc* p, $O(p) \leq O(p')$ *then call this arc REG arc.*

2. *If every arc on a path is REG, then this is a REG path.*

3. *If every arc on a monotonic decreasing path is REG then this is a MDREG path.*

Minimal E-sum The minimal e-sum scan from an arc is an e-sum scan in which every selected arc on any AND-node has minimum order.

The following algorithm finds a minimal e-sum from an arc.

Algorithm 2.2 1. *Start from* p. *Let* $BUF = \{p\}$ *initially.*

2. *Do the followings until BUF is empty*

   (a) *Get an element from BUF. Let this be* p.

   (b) *For* $p \dashv (r_1, .., r_n)$, *put every* r_i *to BUF.*

   (c) *For* $p \vdash r$, *choose* r *and put to BUF.*

   (d) *For* $p \bullet (r_1, .., r_n)$, *choose a* r_i *such that* $O(r_i) = min\{O(r_1), .., O(r_n)\}$ *and put it to BUF.*

   (e) *For* $p1$ *or* $p0$ *do nothing.*

3. *The e-sum scanned is a MES(Minimal E-Sum).*

Theorem 3 *Let* $O(p') \geq O(p)$. *Then all the paths of a MES scan starting from* p *are MDREG.*

Proof 3 1. *For* $p \dashv (r_1, .., r_n)$, *assume* $O(p') \geq O(p)$. *Then since* $O(p) \geq max\{O(r_1), .., O(r_n)\}$, *for any* r_i, $O(p) \geq O(r_i)$. *On the other hand* $O(r_i') \geq O(p')$ *because this node can be written as* $(r_1', .., r_n') \vdash p'$. *Then the relations becomes* $O(r_i') \geq O(p') \geq O(p) \geq O(r_i)$. *Therefore for every* r_i, $O(r_i') \geq O(r_i)$.

2. *For* $p \vdash r$, *we know that* $O(p) \geq O(r)$. *Also this node can be written as* $r' \dashv (p', ..)$. *I.e.* $O(r') \geq O(p')$. *Then the relations becomes* $O(r') \geq O(p') \geq O(p) \geq O(r)$. *Therefore* $O(r') \geq O(r)$.

3. *For* $p\bullet(r_1, r_2)$, $O(p) \geq min\{O(r_1), O(r_2)\}$. *Let choose a* r_i *such that* $O(r_i) = min\{O(r_1), O(r_2)\}$. *Let this be* r_1 *without loss of generality. Since* $O(r_1) = min\{O(r_1), O(r_2)\}$, $O(r_1) \leq O(r_2)$ *and* $O(p) \geq O(r_1)$. *Also this node can be written as* $r_1' \bullet (r_2, p')$ *so* $O(r_1') \geq min\{O(r_2), O(p')\}$.

(a) *Assume* $O(r_2) = min\{O(r_2), O(p')\}$. *Then from the relations* $O(r_1') \geq O(r_2) \geq O(r_1)$, *we conclude* $O(r_1') \geq O(r_1)$

(b) *Assume* $O(p') = min\{O(r_2), O(p')\}$. *Then from the relations* $O(r_1') \geq O(p') \geq O(p) \geq O(r_1)$, *we conclude* $O(r_1') \geq O(r_1)$

So the only possibility is that $O(r_1') \geq O(r_1)$

■

Theorem 4 *If a path is MDREG and in the form ..r..r'.. then* $O(r) = O(r')$.

Proof 4 *Since the path is MD,* $O(r) \geq O(r')$. *On the other hand, since it is REG,* $O(r') \geq O(r)$. *So the only possibility is that* $O(r) = O(r')$. ■

Maximal E-product The maximal e-product scan from an arc is an e-product scan in which every selected arc on any OR-node has maximum order.

The following algorithm finds a maximal e-product from an arc.

Algorithm 2.3 *1. Start from p. Let* $BUF = \{p\}$ *initially.*

2. Do the followings until BUF is empty

(a) *Get an element from BUF. Let this be p.*

(b) *For* $p \dashv (r_1, .., r_n)$, *choose a* r_i *such that* $O(r_i) = max\{O(r_1), .., O(r_n)\}$ *and put it to BUF.*

(c) *For* $p \vdash r$, *choose r and put to BUF.*

(d) *For* $p \bullet (r_1, .., r_n)$, *put every* r_i *to BUF.*

(e) *For* $p1$ *or* $p0$ *do nothing.*

3. The e-product scanned is a MEP(Maximal E-Product).

Min-max Paths A path common to a MES and a MEP is a min-max path. In a min-max paths, the outgoing arches on the OR-nodes has maximum order and the outgoing arches on the AND-nodes has minimum order.

Theorem 5 *Let p is a REG arc. Then any min-max path starting from p is a MDREG path.*

Proof 5 *Any min-max path is a part of a MES scan. Since all the paths of a MES scan starting from a REG arc is MDREG, any min-max path starting from a REG arc is MDREG.* ■

2.2 An algorithm

Let the arches take order according to the following rules where $O(p)$ shows the order of arc p.

1. For $p \bullet (r_1, .., r_n)$, $O(p) \geq \min\{O(r_1), .., O(r_n), O(p')\}$.
2. For $p \dashv (r_1, .., r_n)$, let $A = \max\{O(r_1), .., O(r_n)\}$. Then $O(p) \geq \min\{A, O(p')\}$
3. For $p \vdash r$, $O(p) \geq O(r)$.

4. For $p0$, $O(p) \geq 0$.

5. For $p1$, $O(p) \geq 1$.

Now let introduce an algorithm which gives orders to the arches.
$STAGE_1$ algorithm

Algorithm 2.4 *1. Initially, there are three orders $\{0, \epsilon, 1\}$ and all the arches of the formula is accepted to be marked with ϵ where $0 < \epsilon < 1$.*

2. Apply EXPAND algorithm.

3. Repeat the following until there is no arc p where $O(p) = O(p') < 1$.

(a) Find an arc p such that $O(p) = O(p') < 1$. Create a new order M such that $O(p) < M$ and $M < L$ if $O(p) < L$. I.e. M is the next order of $O(p)$. Then make $O(p) = M$. Now p is said to be the source arc of M and denoted as $p = S(M)$.

(b) Apply EXPAND algorithm.

EXPAND algorithm

Algorithm 2.5 *1. Repeat the following for every arc of the graph until there is no change in the orders of them.*

(a) Find an arc p such that $O(p) < O(p)_i$, $O(p) \leq O(p')$ and do the followings.

i. If $O(p)_c = \{\}$ or $O(p)_c = 1$ then make $O(p) = O(p)_i$.

ii. Otherwise make $O(p) = O(p)_c$.

(b) Then check whether $O(p) = O(p') = L$ and $O(p') = S(L)$ where L can be any order except 1. If the condition is satisfied then make $O(p) = 1$. Thus we ensure that if the order of a source arc is less than 1, then it and its reverse arc does not have the same order. Here p is called a pseudo source of 1 since $O(p)_i < 1$.

$O(p)_c$ and $O(p)_i$ is calculated as follows.

1. For $p \bullet (r_1, .., r_n)$, let $A = \{O(r_1), .., O(r_n)\}$. Then $O(p)_c = \min\{A - \{O(p')\}\}$ and $O(p)_i = \min\{A\}$.

2. For $p \dashv (r_1, .., r_n)$, let $O(p)_c = O(p)_i = \max\{O(r_1), .., O(r_n)\}$.

3. For $p \vdash r$, $O(p)_c = O(p)_i = O(r)$.

4. For $p0$, $O(p)_c = O(p)_i = 0$.

5. For $p1$, $O(p)_c = O(p)_i = 1$.

After $STAGE_1$, if $O(k) < 1$ where k is the root arc then the formula is invalid and thus the original formula is not a tautology. This algorithm is designed such that every arc whose order is less than 1 will have a reverse arc whose order is different. Thus by theorem 3 and 4 is is proved as follows.

Theorem 6 *Assume in a formula for every arc p, if $O(p) < 1$ then $O(p) \neq O(p'$. Then if $O(r) < 1$ and $O(r) < O(r')$ then r is an invalid arc.*

Proof 6 *It is known that (theorem 3) starting from r there is an e-sum whose all paths are MDREG. Now we will show that any such path must be unidirectional. Assume such a path is not unidirectional. Then it must be in the form r..s..s'... Since it is monotonically decreasing $O(s') \leq O(s) \leq O(r) < 1$. Since it is REG(reverse is equal or greater) then $O(s) \leq O(s')$. So the only possible condition is $O(s) = O(s')$. On the other hand since $O(s) < 1$ $O(s) \neq O(s')$. So the path can not be in the given form and thus unidirectional. This means that starting from r there is MD(monotonically decreasing) e-sum whose all paths are unidirectional. No path in this e-sum could terminate with 1 because this is a MD e-sum. So this is an invalid e-sum. Thus r is an invalid arc.* ■

However, if $O(k) = 1$ then this does not mean that the formula is valid. Now apply the following algorithm.

$STAGE_2$ algorithm.

Algorithm 2.6 *1. For all OR-nodes $p \dashv r_1, ..r_n$, for any i, if $O(r)_i = 1$ then mark r_i as MA(Maximal Arc).*

2. Do the following for all arches until there is no such an arc.

(a) For $p \dashv (r_1, .., r_n)$ if $O(p) = 1$ and there is a r_j such that $O(r)_j < O(p')$ and r_j is marked as MA then make $O(p) = O(r)_j$.

(b) For $p \bullet (r_1, ..r_n)$ if $O(p) = 1$ and there is a r_j such that $O(r)_j < O(p')$ then make $O(p) = O(r)_j$.

(c) For $p \vdash r$, if $O(p) = 1$ and $O(r) < O(p')$ then make $O(p) = O(r)$.

3. Apply EXPAND algorithm.

If at the end of this algorithm, the order of the root arc is still 1 then the formula is valid and thus the original formula is a tautology. Otherwise, it is invalid and the original formula is not a tautology.

Invalidity Proved by the previous theorem.

Validity

Definition 4 *1. If $O(p) = 1$ and $O(p)_i < 1$ then p is called a pseudo 1 source.*

2. A pseudo 1 source p becomes a true source when $O(p)_i$ becomes 1.

The proof is based on the fact that all the 1 sources are pseudo sources initially. If an arc is invalid and a pseudo source then it will never become a true source at the end of $STAGE_1$ algorithm. The $STAGE_2$ algorithm will remove all the pseudo 1 sources and mark the root arc as having an order less than 1, if it has been marked as 1 because of the existence of such pseudo sources.

To be continued....

Complexity Let for an arc p, $O(p) < 1$ and $O(p) \neq O(p')$. The complexity proof is based on the fact that with $STAGE_1$ algorithm once the orders on p and p' is differentiated, they will never be the same unless their order becomes 1.

To be continued...