Two Dimensional Formulas and Polynomial Time Tautology Checking

Lokman Kolukısa

October 6, 2005

Contents

1 Introduction

Finding whether a Boolean formula is a tautology or not in a feasible time is an important problem of computer science. Many algorithms have been developed to solve this problem. Here a polynomial time tautology checking method for Boolean formulas consisting only OR, AND,NOT operator signs, variable signs and parenthesis will be present.

Boolean functions(or any functions in general) can be represented in various ways. We assume that the Boolean functions given is a string consist of AND, OR, NOT signs, variable signs and parenthesis written as usual.

There are infinite number of different formulas representing the same function. For example $x \vee y$ is equal to $x \vee y \vee x \vee y..x \vee y$ where $x \vee y$ is duplicated as many times as desired. Based on this fact we can say that all the different strings denoting the same function is equivalent(functionally).

In fact only 2^{2^n} different functions may be denoted with n different variables. The tautology checking problem may be expressed as "is a given formula a different writing of the function 1".

The most straightforward way of checking whether a formula is tautology or not is to evaluate the formula for each interpretations of the variables. The number of interpretation for n different variables is 2^n. The number of different variables in a formula is at most $s/2$ where s is the length of the formula. So the function must be evaluated $2s/2$ times. That is the complexity is $O(s2^{s/2})$.

In this paper we first give a method to reduce a given statement formula to 1 if it is a tautology and to 0 if not. Then in the following section statement formulas are converted to graphs. Although this graph is very similar to an AND-OR graph, there are minor differences. In the other sections, a polynomial time algorithm to solve this graph is presented. The signs $\vee, \cdot(dot), \tilde{}$ will be used to denote OR,AND and NOT operators respectively.

2 Statement Formulas

Theorem 1 *$f(x_1, .., x_i, .., x_n)$ is a tautology if and only if $f(x_1, .., 0, .., x_n)f(x_1, .., 1, .., x_n)$ is a tautology.*

Proof 1 *Trivial.*

According to this proposition, from a formula, we can obtain another one which has one less variable and still preserves the tautological property of the original one. This is a variable reduction operation and can be illustrated as

$f(x_1, .., x_i, .., x_n) \xrightarrow{x_i} f(x_1, .., 0, .., x_n)f(x_1..1..x_n)$

Let call this v-reduction.

All the variables of a Boolean formula can be eliminated by a series of v-reduction. What would be in hand at last is either a 1 or 0. Since each v-reduction preserves the tautological property of the original formula, a 1 at hand means the original formula is a tautology an 0 means it is not a tautology.

The obvious way of doing reduction is to get two formulas r_1 and r_2 as follows; replace x_i with 0 in the initial formula to obtain r_1 and replace x_i with in the initial formula to obtain r_2, parenthesis them if necessary and put an AND sign between them. Also all the 1's and 0's are removed from the formula by applying the identities

$0x = 0, 1x = x, 0x = x, 1x = 1, \not{0} = 1, 1 = 0.$

All of these process takes polynomial time. However the length of $(r_1) \cdot (r_2)$ which is the resulting formula would be $2s$ approximately in the worst case. Doing this for $s/2$ variables would give us a complexity of $O(s \cdot 2^{s/2}$, so it seems that this process could not be achieved in polynomial time.

2.1 Simple Form

Each Boolean formula has a disjunctive normal form or DNF. In a disjunctive normal form, each elementary product either contain x_i or $\widetilde{x_i}$ or neither contain x_i nor $\widetilde{x_i}$ as follows.

$x_i \cdot \alpha_1 \vee .. \vee x_i \cdot \alpha_n \vee \widetilde{x_i} \cdot \beta_1 \vee .. \vee \widetilde{x_i} \cdot \beta_m \vee \delta_1 \vee .. \vee \delta_l$

Here δ_k is the elementary product including neither x_i nor $\widetilde{x_i}$. The elementary products which includes both x_i and $\widetilde{x_i}$ are either $x_i \cdot \alpha_i$ or $\widetilde{x_i}.\beta_j$. If an α_i includes x_i or $\widetilde{x_i}$, then they can be eliminated by the following equivalences.

$$x_i \cdot f(x_i) = x_i \cdot f(1) x_i \cdot f(\widetilde{x_i}) = x_i \cdot f(0)$$

Similarly if any β_j includes x_i or $\widetilde{x_i}$, they are eliminated by

$$\begin{aligned} \widetilde{x_i} \cdot f(x_i) &= \widetilde{x_i} \cdot f(0) \\ \widetilde{x_i} \cdot f(\widetilde{x_i}) &= \widetilde{x_i} \cdot f(1) \end{aligned}$$

Thus, for any i, j, k the terms α_i, β_j, δ_k do not include x_i or $\widetilde{x_i}$. Such a formula can be written as

$x_i \cdot \alpha \vee \widetilde{x_i} \cdot \beta \vee \delta$

where $\alpha = \alpha_1 \vee .. \vee \alpha_n$, $\beta = \beta_1 \vee .. \vee \beta_m$ and $\delta = \delta_1 \vee .. \vee \delta_l$

This is called as "simple form". Now let see the reduction in this form.

$$\begin{aligned} x_i \cdot \alpha \vee \widetilde{x_i} \cdot \beta \vee \delta &\xrightarrow{x_i} (\beta \vee \delta) \cdot (\alpha \vee \delta) \\ &= \beta \cdot \alpha \vee \delta \end{aligned}$$

After this point on, we will assume the NOT operator is applied to the variable symbols only. f a formula is not in this form, then an equivalent formula in this form can be obtained in polynomial time.

The DNF form of a given formula can be produced by applying the rule $x \cdot (y \vee z) = x \cdot y \vee x \cdot z$ to the formula recursively. Then the right hand side of the v-reduction could be obtained by examining each elementary product to mark them as α_i,β_j or δ_k and by doing the necessary operations to get $\alpha \cdot \beta \vee \delta$. However, the number of elementary product is not polynomial according to the n different variables. For example the size of the formula

$$(x_1 \vee x_2) \cdot (x_3 \vee x_4) .. (x_{n-1} \vee x_n)$$

is $6n/3 - 1$ ad the size of the DNF form of it, given below, is $4.2^{n/2} - 1$.

$x_1 \cdot x_3 \cdot x_5 .. x_{n-1} \vee .. \vee x_2 \cdot x_4 \cdot x_6 .. x_n$

Therefore, obtaining all the elementary formula in this way dos not take polynomial time.

So let investigate the other ways of reducing the formula. t can easily be seen that, if a formula does not have x_i but includes $\widetilde{x_i}$, then the simple form would be $\widetilde{x_i} \cdot \beta \vee \delta$ and the reduction becomes as

$$\begin{aligned} \widetilde{x_i} \cdot \beta \vee \delta &\xrightarrow{x_i} (\beta \vee \delta) \cdot \delta \\ &= \delta \end{aligned}$$

But this can be achieved simply by putting 0 instead of $\widetilde{x_i}$ and eliminating 0's from the formula. This process can be applied to the original formula in polynomial time. Similarly for the formulas which does not include $\widetilde{x_i}$ but having x_i, reducing is achieved as

$$x_i \cdot \alpha \vee \delta \;\xrightarrow{x_i}\; (\alpha \vee \delta) \cdot \delta$$
$$= \delta$$

Again the reduced form can be obtained from the original one by replacing x_i with 0 and eliminating all 0s

Thus any variable which occurs either only alone or only with negation can be eliminated by just replacing them with 0. This takes polynomial time. As a result, a formula is obtained in which all the variables occur at least twice; one as alone and the other with negation. Now if we evaluate the formula by examining every interpretations of them, the complexity would be $O(s.2^{s/2})$ since the number of different variables is $s/4$ at most in such a formula.

So we will need to consider only the formulas in which every variable occurs both alone and with negation. If the formula is not of this type then it can be converted to this type in polynomial time as just explained.

AND paths Consider the v-reduction again.

$f(x_i, \widetilde{x}_i) = x_i \cdot \alpha \vee \widetilde{x}_i.\beta \vee \delta \xrightarrow{x_i} \beta \cdot \alpha \vee \delta$

Here is the question is that "are there alternate ways of obtaining the right hand side of the reduction from the left hand side". The answer is yes and as shown below, the formula can be reduced by replacing x_i with β_i and $\widetilde{x}_i$ with 0.

$$x_i \cdot \alpha \vee \widetilde{x}_i.\beta \vee \delta \;\xrightarrow{x_i}\; \underset{\beta}{\cancel{x_i}} \cdot \alpha \vee \underset{0}{\cancel{\widetilde{x}_i}}.\beta \vee \delta$$
$$= \beta \cdot \alpha \vee \delta$$

The same result can be obtained by replacing x_i with 0 and $\widetilde{x}_i$ with α or by replacing x_i with β and $\widetilde{x}_i$ with α.

$$x_i \cdot \alpha \vee \widetilde{x}_i.\beta \vee \delta \;\xrightarrow{x_i}\; \underset{\beta}{\cancel{x_i}} \cdot \alpha \vee \underset{0}{\cancel{\widetilde{x}_i}}.\beta \vee \delta$$
$$= \beta \cdot \alpha \vee \delta$$

$$x_i \cdot \alpha \vee \widetilde{x}_i.\beta \vee \delta \;\xrightarrow{x_i}\; \underset{\beta}{\cancel{x_i}} \cdot \alpha \vee \underset{0}{\cancel{\widetilde{x}_i}}.\beta \vee \delta$$
$$= \beta \cdot \alpha \vee \delta$$

α and β are just the terms conjunct ed with x_i and $\widetilde{x}_i$ respectively.

This idea can be applied to any formula even it is not in DNF form.

$$f(x_i, \widetilde{x}_i) \;\xrightarrow{x_i}\; f(\beta, 0)$$
$$f(x_i, \widetilde{x}_i) \;\xrightarrow{x_i}\; f(0, \alpha)$$
$$f(x_i, \widetilde{x}_i) \;\xrightarrow{x_i}\; f(\beta, \alpha)$$

Definition 1 *If two terms t_1 and t_2 are conjunct ed then t_2 is an a-term of t_1 and vice verse*

For example in $(x \vee y) \cdot z$, $(x \vee y)$ is an A-term of z and z is an A-term of $(x \vee y)$.

Then the idea "replace x_i with β" becomes "replace each occurrence of x_i with the A-term of $\widetilde{x_i}$". Similarly, the idea "replace $\widetilde{x_i}$ with α" becomes "replace each occurrence of $\widetilde{x_i}$ with the A-term of x_i"

Finding AND-Links In any formula, each occurrence of x_i in a formula is in the following form.
$..l_m \cdot (u_k \vee .. \vee l_2 \cdot (u_i \vee l_1 \cdot x_i \cdot r_1 \vee u_{i-1}) \cdot r_2 \vee .. \vee u_1) \cdot r_n \vee ..$

The A-term of x_i for this occurrence is $l_m..l_2 \cdot l_1 \cdot r_1 \cdot r_2..r_n$. Let this be α_k where k represents the k'th occurrence of x_i. If α_k contains x_i or $\widetilde{x_i}$, then they can be eliminated by replacing them with 1 and 0 respectively, since α_k is an A-term of x_i . The entire A-term of x_i then becomes $\alpha = \alpha_1 \vee .. \vee \alpha_n$ where n is the number of occurrences of x_i in the formula.

Finding an A-term of x_i takes polynomial time. One algorithm to do this is the following.

Algorithm 2.1 *Find all the occurrences of x_i. Repeat the following for each occurrence of x_i.*

1. *Let T be an empty set and x_i is the last A-term.*
2. *Do the following steps toward right until the end of the formula*
   (a) *If t_1 is the last A-term and the formula is in the form $..t_1 \cdot t_2..$ then make t_2 the last A-term and put it into T.*
   (b) *If the formula is in the form $..(.. \vee t \cdot t_i \vee .. \vee t_n) \cdot r..$ and t_i is the last A-term, then make r the last A-term and put it into T.*
3. *Starting from x_i, do the following steps toward left until the end of the formula.*
   (a) *If t_2 is the last A-term and the formula is in the form $..t1 \cdot t2..$ then make t_1 the last A-term and put it into T.*
   (b) *f t is the last A-term and the formula is in the form $..l \cdot (t1 \vee .. \vee t \cdot t_i \vee .. \vee t_n)..$, then make l is the last A-term and put it into T.*

The A-term of the this occurrence would be $a_1 \cdot .. \cdot a_n$ where $a_i \in T$

In this case, the A-term of x_i, i.e. α, is the disjunction of the and-terms of all the occurrences of x_i. If the number of occurrences of x_i is n and the length of the formula is s, this operation takes ns times. If there are x_i or $\widetilde{x_i}$ terms in α, then they are eliminated by the equations $\alpha(x_i, \widetilde{x_i}) = \alpha(1, 0)$. β is found similarly. The term δ can be found by replacing x_i and $\widetilde{x_i}$ with 0. All of these jobs take polynomial time.So the reduced formula $\alpha \cdot \beta \vee \delta$ can be found in polynomial time. However, the length of the formula found may be larger than the original one. If the length of the formula found is $2s$ in average then the length seems to increase exponential in each reduction. On the other hand, the number of different variables would be reduced by one and thus the length can not increase so much. From here, one may prove that the length can not increase exponentially. However, we will assume that this process can not be done in polynomial time.

The algorithm above replaces the concept of being an A-term with the concept of going from one term to another. To visualize this concept, we will use – symbol instead of $\cdot$. By this notation, a statement formula would look like
$..l_m\text{–}(u_k \vee .. \vee l_2\text{–}(u_i \vee l_1\text{–}x_i\text{–}r_1 \vee u_{i-1})\text{–}r_2 \vee .. \vee u_1)\text{–}r_n \vee ..$

Thus – represents both reachability and AND-links. Two terms are conjuncted to each other if they are reachable to each other by the following rules.

1. In t_1-t_2, from t_1 one can reach to t_2 and vice verse.
2. In $(t_1 \vee t_2)$-t_3, t_3 is reached from t_1 or from t_2 and from t_3, t_1 or t_2 can be reached.
3. If t_2 is reachable from t_1 and t_3 is reachable from t_2, then t_3 is reachable from t_1.

Now consider the idea "replace x_i with β and replace $\widetilde{x}_i$ with α" again.

$$\begin{aligned} x_i\text{-}\alpha \vee \widetilde{x}_i\text{-}\beta \vee \delta \ &\stackrel{x_i}{\rightarrow}\ (\beta \vee \delta)\text{-}(\alpha \vee \delta) \\ &= \beta\text{-}\alpha \vee \delta \end{aligned}$$

Note that after replacement α becomes conjuncted to β and vice verse. To make two term conjuncted, it is sufficient to made them reachable from each other. So the idea can be realized by

$$x_i\text{-}\alpha \vee \widetilde{x}_i\text{-}\beta \vee \delta \ \stackrel{x_i}{\rightarrow}\ 1\text{-}\alpha \vee 1\text{-}\beta \vee \delta$$

Here the path between α and β represents reach ability. A better realization is as follows

$$x_i\text{-}\alpha \vee \widetilde{x}_i\text{-}\beta \vee \delta \ \stackrel{x_i}{\rightarrow}\ 1\text{-}\alpha \vee 1\text{-}\beta \vee \delta$$

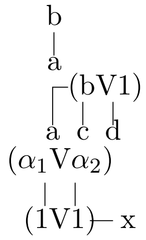
3 Two Dimensional Formulas

Once we escape from the restrictions of one dimension and feel free to draw links in two dimensions, then the amount of choices to express a function will be increased. Consider the following examples.

a–b =

a–$(b$–$c \vee d)$ =

x–$\alpha_1 \vee x$–α_2 =



Two dimensional formulas can easily be obtained from statement formulas as shown in the figure.

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

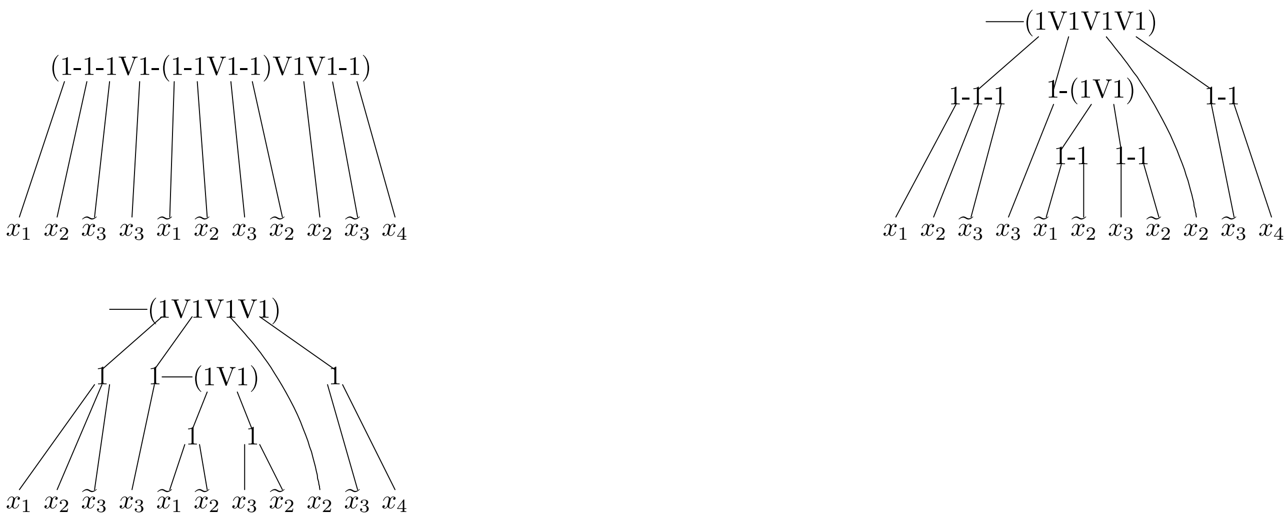


Figure 1: Two dimensional equivalences of a statement formula

The third form is a tree structure composed of arches, literals and the nodes $-1-$ and $(1V..V1)$. The first node is an AND-node and the second one is an OR-node. The OR-node has a main arc and several base arches. Every formula has a root arc.

To show that the above equivalences are true, it is enough to show that the statement formula and the corresponding two dimensional forms takes the same value at every interpretation of the variables.

In the following example, the two dimensional formula and the value of each arc for an interpretation is shown. As seen the arches has directions. The value of each arc is found as follows: If an arc is connected to a variable then the value of the arc becomes the value of that variable. For an OR-node, if one of the outgoing base arches is 1 then the main incoming arc becomes 1. Otherwise it becomes 0. For an AND-node if one of the outgoing arches is 0, then the incoming arc becomes 0. Otherwise it becomes 1. For this interpretation the root arc takes the value 0, so this means that the value of this formula is 0 for this interpretation.

Arches and Formulas Consider the formula given in the figure. In the formula, all the arches are labeled to differentiate them from each other.

The arches are labeled by small letters. A label near an arc normally represents the arc in the downward direction. If an arc has no vertical direction, i.e. it is drawn in the left-rigth direction, then the label near it denotes the arc directed to right. The reverse of an arc is marked by an $'$ sign above the label of it. For example the reverse arc of the arc y is written as y'. Obviously, the reverse of the reverse of an arc is the arc itself.

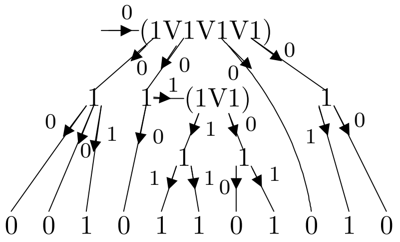


Figure 2: The values of the arches for the interpretation $x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0$

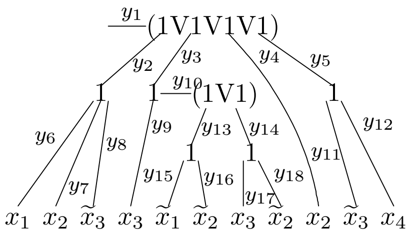
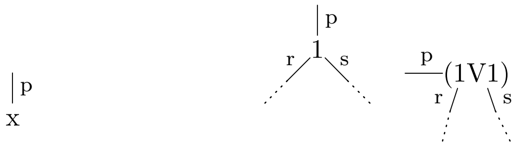


Figure 3: Labeling arches

Each arc is either connected to a literal or an AND-node or an OR-node. If an arc p is connected to a literal, let write this as px where x is the literal. If it is connected to an AND-node, i.e. it is the incoming arc of an AND-node, let write this as p–$(r_1, .., r_n)$ where $r_1, .., r_n$ are the outgoing arches. If it is an incoming main arc of an OR-node, let write this as p–$(p_1 \vee .. \vee p_n)$ where $r_1, .., r_n$ are the outgoing arches.



(a) px

(b) $p -$ (r, s)

(c) $p - (r \vee s)$

Figure 4: Connection types

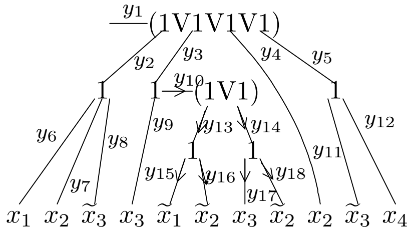
For any arc p, let $|p|$ be represent the value of the arc in any interpretation. The value of an arc p in any interpretation is found as follows.

1. For px $|p| = |x|$
2. For p–$(r_1, .., r_n)$ $|p| = |r_1| \cdot .. \cdot |r_n|$ holds.
3. For p–$(r_1 \vee .. \vee r_n)$ $|p| = |r_1| \vee .. \vee |r_n|$ holds.

Each arc represents a formula and this is the part of the whole formula which affects the value of this arc. If the value of an arc depends on the value of an literal, this means that there is a path from that arc to this literal. A path from an arc to a literal can be seen as a way on which the value of the literal propagates back to the arc. So the formula of an arc is the sub formula which is scanned by following the paths beginning from that arc. For example the sub formula beginning from the arc y_{10} is shown in the figure.

Obviously an arc is the root of its formula. For example the arc y_{10} is the root of the sub formula found starting from y_{10}.

Each formula and therefore each arc has a statement equivalent. The statement equivalence of a two dimensional formula is a statement formula which has the same value with the two dimensional



(a) The formula of y_{10}

Figure 5: Each arc represents a formula

formula in every interpretation. Let $\|p\|$ denotes the statement equivalence of the formula beginning from arc p. Then for example $\|y_{10}\| = \widetilde{x_1} \cdot \widetilde{x_2} \vee x_3 \cdot \widetilde{x_2}$ would be true because the statement formula $\widetilde{x_1} \cdot \widetilde{x_2} \vee x_3 \cdot \widetilde{x_2}$ and the two dimensional formula beginning from y_{10} has the same value in every interpretation.

3.1 Reduction

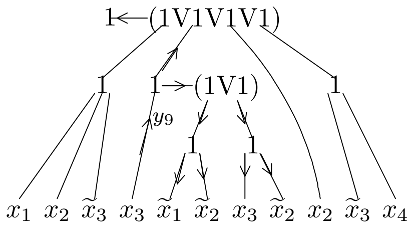
In a statement formula, we can get the reduced form without converting it into simple form as follows:

$f(x_i, \widetilde{x_i}) \xrightarrow{x_i} f(\beta, \alpha)$

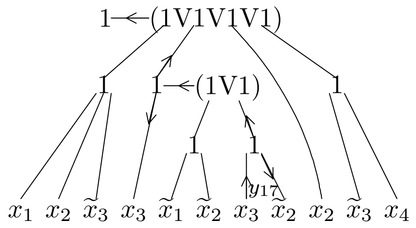
So the reduction for x_3 is simple. replace every occurrence of x_3 with the A-term of $\widetilde{x_3}$ and replace every occurrence of $\widetilde{x_3}$ with the A-term of x_3.

The problem here is to find the A-term of a variable. However, in two dimensional formulas, finding the A-term of a variable is easy. Let the arc connected to an occurrence of the variable x is p. Then the A-term of the that occurrence of x is the sub formula starting from p'.

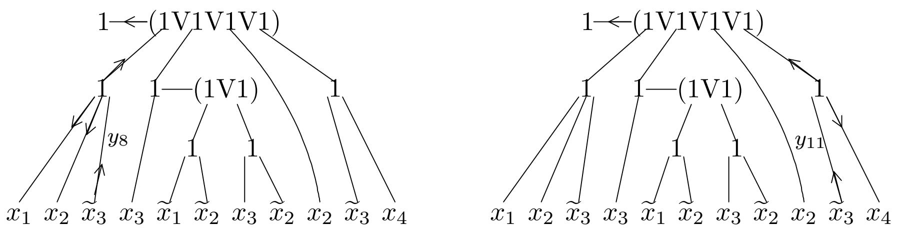
In the figure, the A-terms found for every occurrence of x_3 and $\widetilde{x_3}$ are shown.



(a) the formula of y_9 is an A-term of x_3



(b) Another A-term of x_3



(c) The A-terms of $\widetilde{x_3}$

Figure 6: The A-terms of x_3 ve $\widetilde{x_3}$

The A-term of a variable then is the disjunction of all the A-terms of it.

As seen clearly, the reverse root arc must be end with a 1. We will start the root arc with 1 after here on.

This example illustrates a new type of connection where the incoming arc of an OR-node is one of the base arches. Let $(p_1 \vee .. \vee p_n) - r$ represents an OR-node where $p_1, .., p_n$ are incoming and r is outgoing arches. That is $p - (p_1, .., p_n)$ and $(p'_1, .., p'_n) - p'$ represent the same OR-node, however, in the first representation, the direction is from the main arc to the base arches while in the second one the direction is from base arches to the main arc.

Additional rule for interpretation: For $(p_1 \vee .. \vee p_n)$-r and for any $i = 1, .., n$, $|p_i| = |r|$ becomes true.

The reduction can be realized in two ways as shown in the figure. In both of these reductions, x_3 is replaced by the A-term of $\widetilde{x_3}$ and $\widetilde{x_3}$ is replaced by the A-term of x_3. I.e. in the reduced formula, the arches y_9 and y_{17} takes the value of the A-term of the variable $\widetilde{x_3}$ and the arches y_8 and y_{11} takes the value of the A-term of the variable x_3

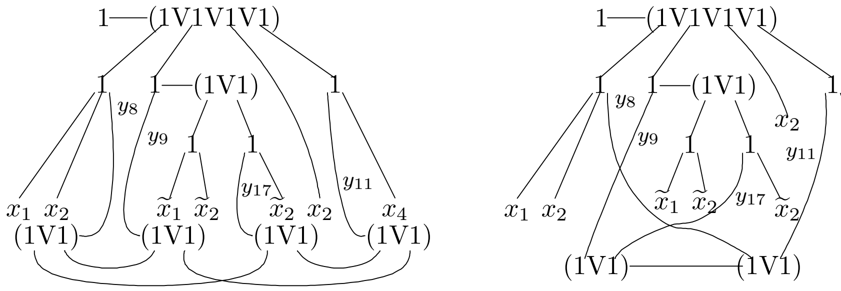


Figure 7: Two ways of reduction by x_3: the arches y_9 and y_{17} takes the value of the disjunction of y'_8 and y'_{11} and the arches y_8 and y_{11} takes the value of the disjunction of y'_9 and y'_{17}

The second method of reduction is more simple and it seems that there is no advantage of using the first method. The second method also lead another way of representing the two dimensional formulas. It is shown in the figure.

In such graphs, the "alone" variables written only once and the others written twice one for itself and another for its negation. Then the reduction by x_i is simple: If both of the variable and its negation exist then replace $\widetilde{x_i}$ and x_i with 1 and connect them with an arc. If only the variable itself or its negation exist then replace it by 0. In this article, we always use this way of reduction.

For the reduction be valid, the reduced two dimensional formula must be equal to its statement counterpart in every interpretation. Let find the reduced form of the statement formula by x_3.

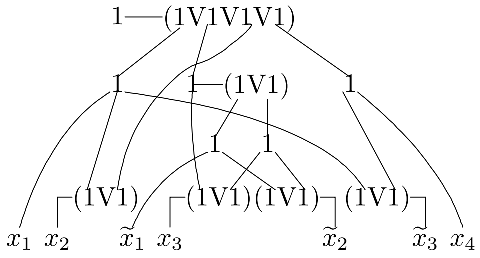
$$
\begin{aligned}
&x_1 \cdot x_2 \cdot \widetilde{x_3} \vee x_3 \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee x_3 \cdot \widetilde{x_2}) \vee x_2 \vee \widetilde{x_3} \cdot x_4 \\
&\qquad \stackrel{x_3}{\rightarrow} \quad (x_1 \cdot x_2 \cdot 1 \vee 0 \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee 0 \cdot \widetilde{x_2}) \vee x_2 \vee 1 \cdot x_4) \\
&\qquad\qquad \cdot (x_1 \cdot x_2 \cdot 0 \vee 1 \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee 1 \cdot \widetilde{x_2}) \vee x_2 \vee 0 \cdot x_4) \\
&\qquad\qquad = (x_1 \cdot x_2 \vee x_2 \vee x_4) \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee \widetilde{x_2} \vee x_2) \\
&\qquad\qquad = (x_2 \vee x_4) \cdot 1 \\
&\qquad\qquad = x_2 \vee x_4
\end{aligned}
$$

I.e. the reduced form of the statement formula becomes $x_2 \vee x_4$. So the reduced two dimensional formula must be equal to this.

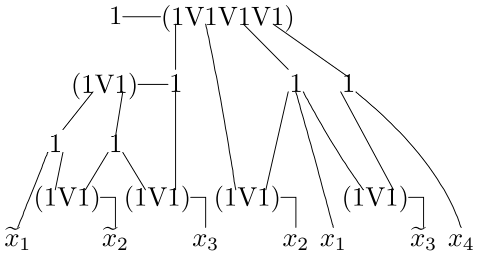
In the figure, the values of the reduced two dimensional formula for the interpretations $(x_2 = 1)$, $(x_2 = 0, x_4 = 1)$ and $(x_2 = 0, x_4 = 0)$. The reduced two dimensional formula and the reduced statement formula have the same value for every interpretation, thus the reduction is valid.

Obtaining the two dimensional form of a statement formula as mentioned above takes polynomial time. For each different variables, one or two OR-nodes are formed. For each literal an arc is connected to this nodes. So the total number of arches becomes the number of parenthesis+the number of literals+2*number of different variables. It takes O(s) times to scan and obtain the graph.

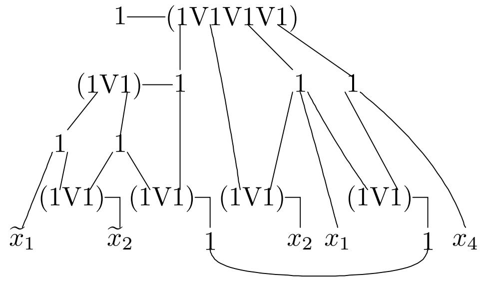
The last form of the formula after reducing by all variables are shown in the figure.



(a) Another form of the formula

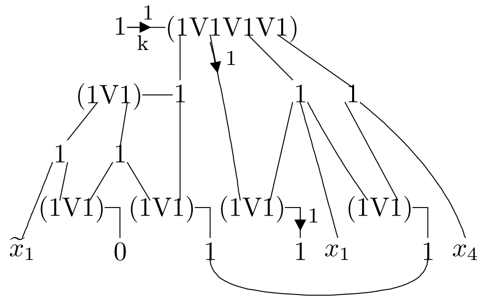


(b) A more simple drawing

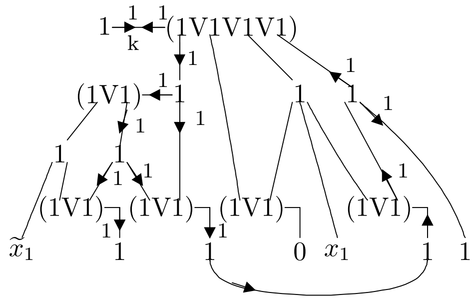


(c) the reduced form by x_3

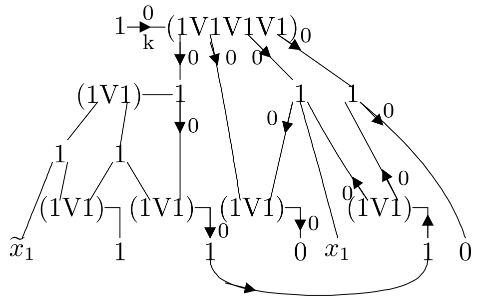
Figure 8: Another form of the sample formula. Each variable and its negation is written only once. The A-term of a variable is the sub formula starting from the reverse of the arc connected to the variable



(a) For $|x_2| = 1$ $|k| = 1$



(b) For $|x_2| = 0$ and $|x_4| = 1$ $|k| = 1$



(c) For $|x_2| = 0$ and $|x_4| = 0$ $|k| = 0$

Figure 9: The interpretations for the reduced formula. The values of the arches which does not affect the result are not shown

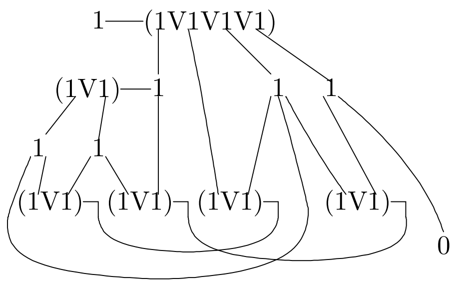


Figure 10: The totally reduced form of the example formula

Now what at hand is a pure graph and we should decide whether a formula is a tautology or not by investigating such a graph. The question here is: "which properties of such a graph differentiate a tautology from a non tautology one"

Before investigating the answer for this question, let change the notation and introduce some definitions.

Let use instead of $(1V..V1)$ and $\bullet$ instead of $-1-$.

The formulas given in the figure is redrawn with this notation and shown in the figure.

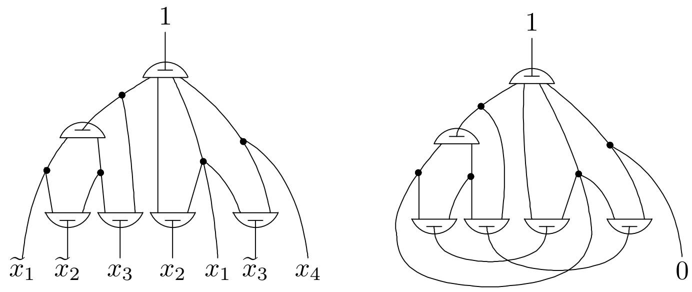


Figure 11: The sample formula and its last reduced form redrawn with new notation

3.2 Some Definitions

Formulas A formula obtained from a statement one as mentioned above is an unreduced formula. If it is reduced by means of any variables are the reduced formulas. If a formula is reduced by all the variables then this is a totally reduced formula. In a totally reduced formula there is no variable.

We want to express a two dimensional formula by means of its arches and nodes. Due to this new notation, we will write the nodes as follows. $p \bullet (p_1, .., p_n)$ represents an AND-node where p is the incoming arc and $p_1, .., p_n$ are the outgoing arches. $p \dashv (p_1, .., p_n)$ represents an OR-node where p is the main incoming arc and $p_1, .., p_n$ are the base outgoing arches. Naturally, the node $p \bullet (r, s)$ can be written as $r' \bullet (p', s)$ or $s' \bullet (p', r)$. Similarly, the node $p \dashv (p_1, .., p_n)$ can be written as $(p'_1, .., p'_n) \vdash p'$.

(a) $p \bullet (r, s)$ or $r' \bullet (p', s)$ or $s' \bullet (p', r)$ or $\bullet(r, s, p')$

(b) $p \dashv (r, s)$ or $(r', s') \vdash p'$

Figure 12: Nodes and their representation in statement forms

We have said that each arc represents a formula. Since each arc is connected to other arches via nodes, each formula can be thought as composed of other formulas via nodes and connection types. An AND-node represents a conjunction of subformulas and an OR-node represents a disjunction. Let the formula of an arc p be represented by $p\{\}$. If p is connected to an AND-node then $p\{\}$ is a conjunction, if it is the main arc of an OR-node then it is a disjunction.

Then we can set the following recursive rules for the expression of a two dimensional formula in statement form

1. The formula of p be $p\{\}$.

2. If p and beyond is in the form px then $p\{\} = px$,

3. If p and beyond is in the form pr then $p\{\} = pr\{\}$.

4. p and beyond is in the form $p \bullet (p_1, .., p_n)$ then $p\{\} = p \bullet \{p_1\{\}, .., p_n\{\}\}$.

5. p and beyond is in the form $p \dashv (p_1, .., p_n)$ then $p\{\} = p \dashv \{p_1\{\}, .., p_n\{\}\}$.

6. p and beyond is in the form $(p, ..) \vdash r$ then $p\{\} = p \vdash r\{\}$.

For example for the sub formula given in the figure the following equivalences would be valid.

$$\begin{aligned} y_3\{\} &= y_3 \bullet \{y_{12}\{\}, y_{19}\{\}, y_{13}\{\}\} \\ &= y_3 \bullet \{y_{12} \vdash y_{18}\{\}, y_{19}x_1, y_{13} \vdash y_{20}\{\}\} \\ &= y_3 \bullet \{y_{12} \vdash y_{18}x_2, y_{19}x_1, y_{13} \vdash y_{20}\widetilde{x_3}\} \end{aligned}$$

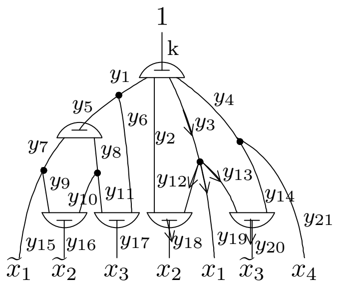


Figure 13: The statement form of the formula starting from y_3 is $y_3 \bullet \{y_{12} \vdash y_{18}x_2, y_{19}x_1, y_{13} \vdash y_{20}\widetilde{x_3}\}$

The reason to choose such a notation is that when the parenthesis are opened, then we get the paths of the formula.

$$y_3\{\} = \{y_3 \bullet y_{12} \vdash y_{18}x_2, y_3 \bullet y_{19}x_1, y_3 \bullet y_{13} \vdash y_{20}\widetilde{x_3}\}$$

Paths A path is denoted by writing arches,nodes and literals in a sequence. For example, the path begin with y_3 and end with x_2 is written as $y_3 \bullet y_{12} \vdash y_{18}x_2$. Here $\vdash$ represents a base-to-main OR-node between the arches y_{12} and y_{18}. Similarly $\bullet$ and $\dashv$ represents an AND-node and a main-to-base OR-node between the arches. The direction of path is from left to right. If we do not need to describe the types of nodes between two arches, we use the notation $p.r$. Here it is indicated that there is

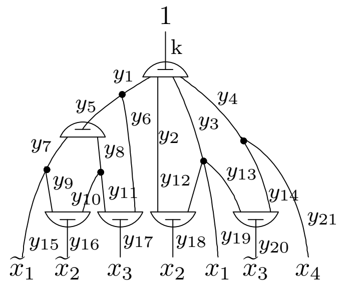


Figure 14: Sample formula

exactly one node between p and r, but the type of node is not described. Two subsequent arches are tied to each other with an AND-node. So if r immediately comes after p we will write this as $p \bullet r$ or pr. Several subsequent arches can be thought as one arc and one arc can be thought as composed of several subsequent arches.

For each path there is a reverse path. For example the reverse of $y_3 \bullet y_{12} \vdash y_{18} x_2$ is $x_2 y'_{18} \dashv y'_{12} \bullet y'_3$. $\dashv$ and $\vdash$ are the reverse of each other. The reverse of a variable such as x is itself. The reverse of a $\bullet$ is a $\bullet$ again.

The notation $p\underline{A}r$ shows a path which begins with p and end with r. The paths are denoted with capital letters underlined. The reverse of the path $p\underline{A}r$ is written as $r'\underline{A}'p'$. On the other hand, the notation $p..r$ denotes any path begins from p and end at r. The reverse of $p..r$ is denoted as $r'..p'$. $p..$ and $..r$ shows the paths beginning from p and end with r respectively.

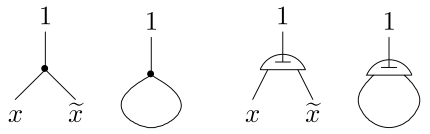
The path relation is transitive. I.e if $p..r$ and $r..s$ exist then a path $p..r..s$ exists.

If a path begins with a literal and end with a literal then it is a complete path. If a path is not complete then it is a subpath.

Each formula can be denoted as a set of its paths.

3.3 Invalid Terms and Reduction

Now to get an idea which form the valid formulas and invalid ones takes let look at the following formulas.



(a) $x \cdot \widetilde{x}$ (b) $x \vee \widetilde{x}$

Figure 15: An invalid and a valid formula and their reduced forms

In the disjunctive normal form of a formula each elementary product either includes x_i or $\widetilde{x}_i$ or does not include them. Then the formula can be written as

$x_i \cdot \alpha \vee \widetilde{x}_i \cdot \beta \vee x_i \cdot \widetilde{x}_i \cdot \delta \vee \gamma$

Here the term $x_i \cdot \widetilde{x}_i \cdot \delta$ is invalid and eliminated directly or indirectly.

$$\begin{aligned} x_i \cdot \alpha \vee \widetilde{x}_i \cdot \beta \vee x_i \cdot \widetilde{x}_i \cdot \delta \vee \gamma \ \xrightarrow{x_i}\ & (0 \cdot \alpha \vee 1 \cdot \beta \vee 0 \cdot 1 \cdot \delta \vee \gamma) \cdot (1 \cdot \alpha \vee 0 \cdot \beta \vee 1 \cdot 0 \cdot \delta \vee \gamma) \\ & = \alpha \cdot \beta \vee \gamma \end{aligned}$$

However, if we reduce the formula by the way used in two dimensional formulas, there is no term eliminated.

$$
\begin{aligned}
f(x_i, \widetilde{x}_i) &\stackrel{x_i}{\to} f(\beta, \alpha) \\
x_i \cdot \alpha \vee \widetilde{x}_i \cdot \beta \vee x_i \cdot \widetilde{x}_i \cdot \delta \vee \gamma &\stackrel{x_i}{\to} \beta \cdot \alpha \vee \alpha \cdot \beta \vee \alpha \cdot \beta \cdot \delta \vee \gamma
\end{aligned}
$$

I.e. invalid terms are preserved after reduction. In any formula the invalid elementary products does not affect the truth of the formula, otherwise they would not be eliminated. Then after any reduction. the value of the formula is determined by the value of elementary products not invalid. If in a formula the invalid products are not eliminated then in the last reduced form of an invalid formula all the elementary products becomes invalid, because the last reduced form of an invalid formula is inconsistent and only the formulas whose all elementary products are inconsistent can be inconsistent.

Thus if in a last reduced form, it is found that every elementary product is inconsistent, then we can decide that the original formula is invalid, otherwise it is valid. However, before applying this idea, we must know what is an elementary product and how they can be found in a two dimensional formula.

4 E-products and Two Dimensional Formulas

4.1 Finding E-products in Unreduced Formulas

In order to find the e-products of a statement formula, one way is to apply the equivalence $x \cdot (y \vee z) = x \cdot y \vee x \cdot z$ to the formula until it can not be applied to any part of the formula. We can use the same method in two dimension. However, such methods modify the formula. The following algorithm finds an e-product without changing the original formula.

Algorithm 4.1 *1. put the formula to the stack.*

2. Do the followings until there is no more element in the stack.

(a) pop an element from the stack.

(b) If this is a disjunction, i.e. the current element is in the form $(a_1 \vee .. \vee a_n)$ then choose one of a_i and push to the stack.

(c) If the current element is in the form $(a_1 \cdot .. \cdot a_n)$ then push all the a_i's to the stack.

(d) If the current element is a literal then put it to VAR.

3. All the literals in VAR forms an e-product. Each different selection in a disjunction leads to different e-products.

For example let find an e-products of $x_1 \cdot x_2 \cdot \widetilde{x_3} \vee x_3 \cdot (\widetilde{x_1} \cdot \widetilde{x_2} \vee x_3 \cdot \widetilde{x_2}) \vee x_2 \vee \widetilde{x_3} \cdot x_4$ by this way. The following figure shows how to find an e-product. The elements marked by an underline shows the selection made in each disjunction.

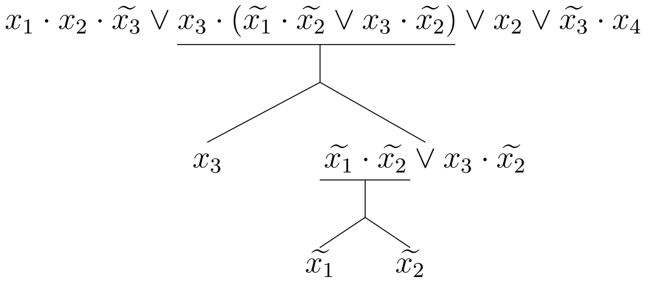


Figure 16: Finding e-products in a statement formula

Now the corresponding algorithm to find an e-product in a two dimensional formula is given below.

Algorithm 4.2 *1. Begin from the root of the formula and push it to the stack.*

2. Do 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. is in the form $p \dashv (r_1, .., r_n)$ then for one $i = 1, .., n$, push exactly one r_i to the stack.

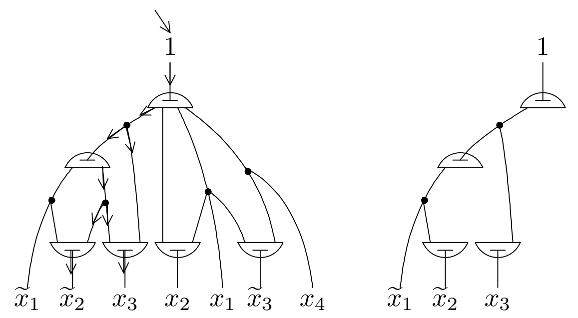
(c) If p is connected to an AND-node, i.e. is in the form $p \bullet (r_1, .., r_n)$, then for all $i = 1, .., n$, push r_i to the stack.

(d) If p and its connection is in the form $p \vdash r$ or pr then push the arc r to the stack.

(e) If p is connected to a variable, i.e. is in the form px then put the variable to VAR.

3. *The sub formula scanned is an e-product. Its statement correspondence is the conjunction of the variables in VAR.*

Let call this algorithm e-product scanning algorithm. In the figure, finding an e-product with this algorithm is shown. The arrows shows the selection and the numbers near the arrows shows the sequence.



(a) scanning of an e-product

(b) The e-product found

Figure 17: Finding an e-product in a two dimensional formula

The following figure illustrates all the other e-products found for this formula. While scanning, selection of different arches in OR-nodes leads to different e-products. Although unnecessary, the OR-nodes in the e-products are shown to make the illustrations more clear.

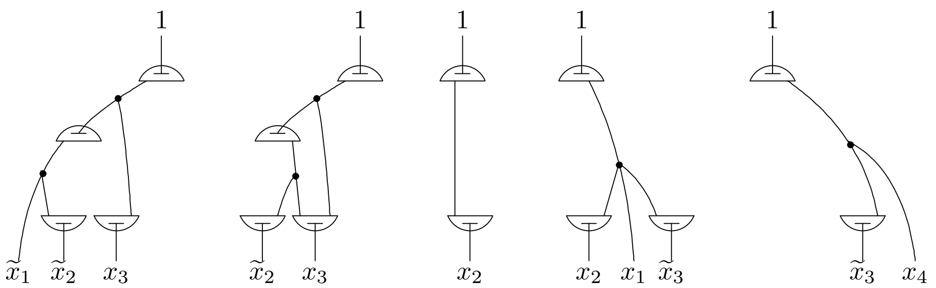


Figure 18: The e-products found

As seen, the e-products found in unreduced formulas is in the form of a tree composed of AND-nodes, arches and literals as terminals. In an e-product there is at least one path from one terminal to the other.

Here, it is necessary to make a distinction between the scanning of an e-product and the e-product itself. In scanning the arches has a direction, however, in the e-products found the arches are not directed and does not depend on the scanning. To get an e-product image from the scanning image, it is enough to remove direction signs from the arches. In an e-product, there is at least one path from one terminal to all the other. An e-product is seen as a tree from all the terminals and no terminal is more important that from the others in that respect. So it can be said that there is a symmetry in the e-products.

On the other hand an e-product can be found only by scanning. The scanning of an e-product only finds the paths from the one terminal to the other terminals. I.e. there can be many different scanning of an e-product. However, as seen only one scanning is enough to find it.

We will investigate the relationship between an e-product and the scanning of it later.

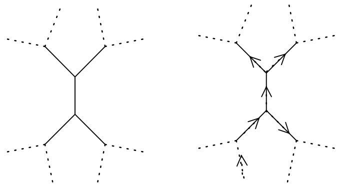


Figure 19: An e-product is seen as a tree from all its terminals and can be found by only one scanning

Not only we can mention the e-products of a formula, but also the e-products of sub formulas beginning from an arc. For example in the following, a sub formula beginning from p and the e-products of it are shown.

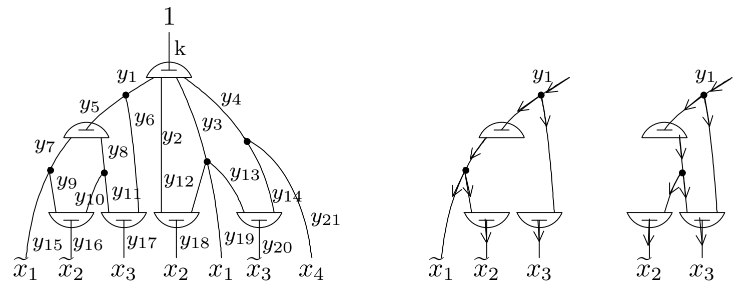


Figure 20: e-products beginning from an arc

Then we should change the scanning algorithm to start the scan from an arc. The following is such an algorithm.

Algorithm 4.3 1. *This algorithm finds an e-product beginning from* p

2. *Put the arc* p *to the stack.*

3. *Do the followings until the stack is empty.*

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

   (b) *If* p *and its connection is in the form* $p \dashv (r_1, .., r_n)$ *then for any* i *push* r_i *to the stack.*

   (c) *If* p *and its connection is in the form* $p \bullet (r_1, .., r_n)$ *then for all* $i = 1, .., n$, *push* r_i *to the stack.*

   (d) *If* p *and its connection is in the form* $p \vdash r$ *or* pr, *then put* r *to the stack.*

   (e) *If* p *and beyond is a variable, then put the variable to VAR.*

4. *The sub graph scanned is an e-product. Its statement correspondence is found by conjunction of the variables in VAR.*

Paths and e-products Each e-product scan can be written as a set of its paths. For example the paths of the e-product in the figure can be written as

$$\{1k.y_1.y_5.y_8.y_{10}.y_{16}\widetilde{x}_2,\ 1k.y_1.y_5.y_8.y_{10}.y_{11}.y_{17}x_3,\ 1k.y_1.y_6.y_{17}x_3\}$$

Not each path set corresponds to an e-scan. In the path set of an e-scan, each path shares a common AND-node with each other paths.

Let call the structures in which every path has a common AND-node with each other path as simple products. Naturally each e-product is a simple product. However, not every simple product is an e-product. In an e-product, every path forked on an AND-node is included in the path set.

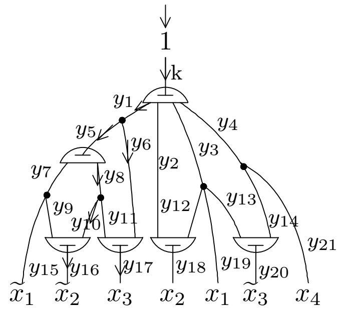
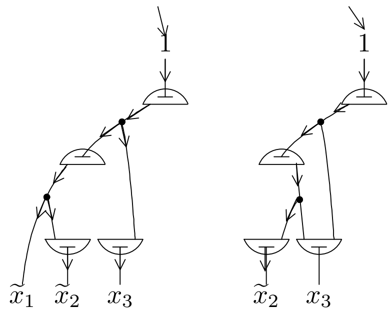


Figure 21:



(a) An e-product scan

(b) A simple product scan

Figure 22:

Definition 2 *If all the paths spanned from an and-node of a simple product are the members of the simple product, then this is an e-product.*

In each simple product, any two paths are forked from an AND-node or any two paths has a common AND-node. Let $⅄$ describe simple products and λ describe e-products. In this case, $⅄a$ and λb denotes that a is a simple product and b is an e-product respectively.

The statement correspondence of any simple product is the disjunction of all the literals of it. So it is possible to define statement correspondences of paths. Let the statement correspondence of a path be the terminal element of that path. E.g. $\|Ax\| = x$. In this case, we can define the following equivalences.

$\|⅄a\| = \|⅄\{A_1, .., A_n\}\| = \|A_1\| \cdot .. \cdot \|A_n\|$

E.g. for $\lambda a = \lambda\{1k.y_1.y_5.y_8.y_{10}.y_{16}\widetilde{x}_2,\ 1k.y_1.y_5.y_8.y_{10}.y_{11}.y_{17}x_3,\ 1k.y_1.y_6.y_{17}x_3\}$

$$\begin{aligned}
\|\lambda a\| &= \|1k.y_1.y_5.y_8.y_{10}.y_{16}\widetilde{x}_2\| \cdot \|1k.y_1.y_5.y_8.y_{10}.y_{11}.y_{17}x_3\| \cdot \|1k.y_1.y_6.y_{17}x_3\| \\
&= \widetilde{x}_2 \cdot x_3 \cdot x_3 \\
&= \widetilde{x}_2 \cdot x_3
\end{aligned}$$

would be valid.

Each e-product can be though as formed from other e-products as follows.

Therefore we can replace the e-product scanning algorithm given with a recursive one as follows.

Algorithm 4.4 *The algorithm for EPSA(p)*

This algorithm scans an e-product starting from p. λp represents the e-product thus scanned.

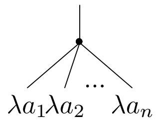


Figure 23: An e-product is formed from the other ones

1. *Do the followings starting from p.*
   (a) *If p and its connection is in the form px then $\lambda p = px$.*
   (b) *If p and its connection is in the form $p \bullet (r_1, .., r_n)$ then for each $r_1, .., r_n$ apply this algorithm. As a result, $\lambda p = p \bullet \{\lambda r_1, .., \lambda r_n\}$ holds.*
   (c) *If p and its connection is in the form $p \dashv (r_1, .., r_n)$, then for only one r_i, apply this algorithm. As a result $\lambda p = p \dashv \lambda r_i$ holds.*

So the paths of an e-product can be found from the paths of other e-products.

4.2 E-products and Formulas

4.2.1 Equality and Equivalence

Let the statement form of the formula starting from p be $p\{\alpha\}$ where $\{\}$ is used to differentiate a formula from the paths and arches.

Equality in two dimensional forms are different than in the statement forms. Now, to talk about the equality, it is not enough to consider only the values, the structure must also be considered.

Definition 3 *If two formulas has the same paths then they are equal.*

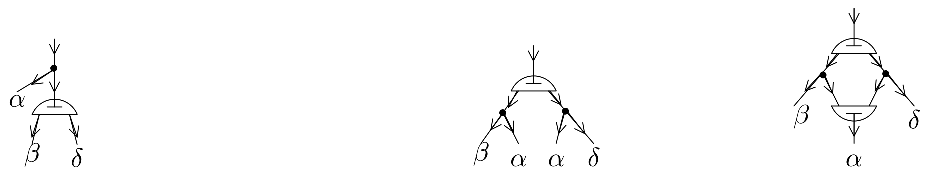
We show the equality with $=$ as usual.

On the other hand we will use the term "equivalence" for the equality when we consider only the values in an interpretation.

Definition 4 *If two formulas have the same values in every interpretations then they are equivalent. Otherwise they are not equivalent.*

Now we will use $\cong$ to describe equivalence. For example $\underline{A}x \cong x$ and $p\{\alpha\} \cong p$ is true because $|\underline{A}x| = |x|$ and $|p\{\alpha\}| = |p|$ is true in every interpretation. Again according to this definition, $p\alpha \cong \|p\{\alpha\}\|$ becomes true, because $p\{\alpha\}$ is a two dimensional formula and $\|p\{\alpha\}\|$ is a statement formula but they have the same values in the same interpretations.

Since every formula is a composition of other formulas, the statement correspondence of every formula can be formed from the statement correspondence of the other formulas.



(a) $p \bullet \{r\{\alpha\}, s \dashv \{ş\{\beta\}, t\{\delta\}\}\} \cong \|r\{\alpha\}\|(\|ş\{\beta\}\| \vee \|t\{\delta\}\|)$

(b) $p \dashv \{r \bullet \{s\{\alpha\}, ş\{\beta\}\}, t \bullet \{u\{\alpha\}, v\{\delta\}\}\} \cong \|s\{\alpha\}\|\|ş\{\beta\}\| \vee \|u\{\alpha\}\|\|v\{\delta\}\|$

Figure 24: Some formulas and their statement correspondences. These formulas are all equivalent since their statement correspondences are equal

To show that they are equivalent, it is enough to show that their statement correspondences are equal.

$$\begin{aligned} p \bullet \{r\{\alpha\}, s \dashv \{ş\{\beta\}, t\{\delta\}\}\} &\cong \|r\{\alpha\}\|(\|ş\{\beta\}\| \vee \|t\{\delta\}\|) \\ &= \|r\{\alpha\}\|\|ş\{\beta\}\| \vee \|r\{\alpha\}\|\|t\{\delta\}\| \\ &\cong p \dashv \{r \bullet \{s\{\alpha\}, ş\{\beta\}\}, t \bullet \{u\{\alpha\}, v\{\delta\}\}\} \end{aligned}$$

Naturally, all the equalities of the statement formulas are valid in two dimensional formulas too. In the following, some equivalences of this type are shown.

Definition 5
1. *The expression $\vee\{a_1, .., a_n\}$ shows that the value of the expression is equal to the disjunction of the values of its elements.*
2. *The expression $\wedge\{a_1, .., a_n\}$ shows that the value of the expression is equal to the conjunction of the values of its elements.*

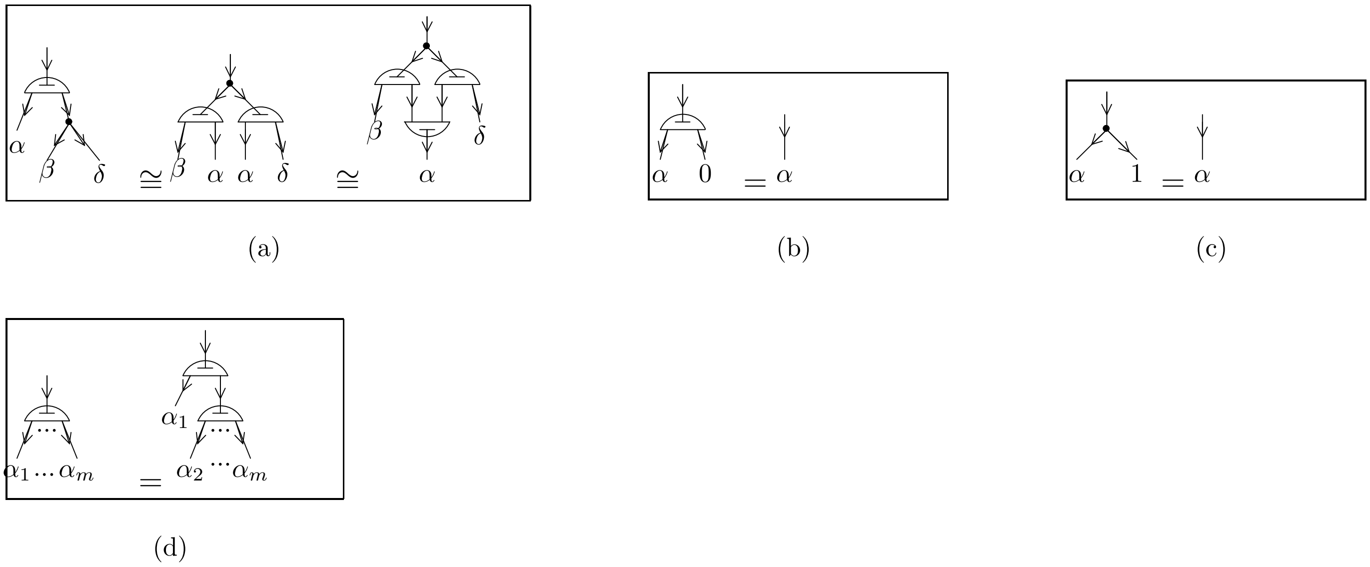


Figure 25: Some equivalences and equalities

E-products and the Value of the Formulas We know that in statement formulas, the value of the formula is equal to the disjunction of the values of its e-products. We will show this is true for two dimensional formulas too.

Each formula can be written as a set of its e-products. Then we can write the equation $p\{\} = \{\lambda p_1, .., \lambda p_n\}$ where λp_i is the i'th e-product beginning from p.

Each formula is either a conjunction or disjunction of other formulas. So the e-product of a formula is composed of the e-product of other formulas.

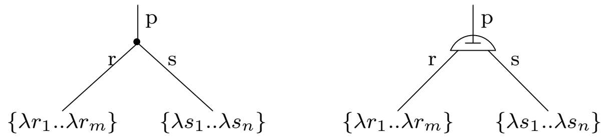


Figure 26: The types of formulas and e-products

In this case, some e-products chosen from p are shown below.

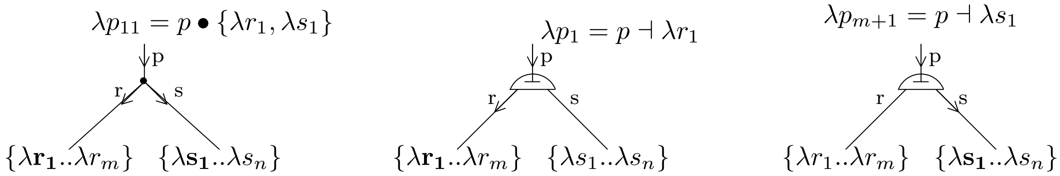


Figure 27: Some e-products found

For each type of formula, the relationship among the e-products are given below.

- If p and the next element are in the form px then $\lambda p = px$
- Let p and the next element be $p \bullet (r, s)$. In this case $p\{\} = p \bullet \{r\{\}, s\{\}\}$ would be true. If $r\{\} = \{\lambda r_1, .., \lambda r_m\}$ and $s\{\} = \{\lambda s_1, .., \lambda s_n\}$ is true, then for any $i = 1, .., m$ and $j = 1, .., n$, $\lambda p_{ij} = p \bullet \{\lambda r_i, \lambda s_j\}$ would be true.

- If p and the next element are in the form $p \dashv (r,s)$, then $p\{\} = p \dashv \{r\{\}, s\{\}\}$ would be true. According to the e-product search algorithm, after p either r or s must be chosen. In that case, the number of all possible e-product would be $m+n$. If $r\{\} = \{\lambda r_1, .., \lambda r_m\}$ and $s\{\} = \{\lambda s_1, .., \lambda s_n\}$ is valid then $\lambda p_i = p \dashv \lambda r_i$ and $\lambda p_{m+j} = p \dashv \lambda s_j$ would be valid.

Now by using these relations, we show if $p\{\} = \{\lambda p_1, .., \lambda p_k\}$ is true, then $\|p\| = \|\lambda p_1\| \vee .. \vee \|\lambda p_k\|$ would be true by induction.

Theorem 1 *In an unreduced formula, let $p\{\} = \{\bigcup_{i=1}^{k} \lambda p_i\}$ be valid. Then $\|p\| = \bigvee_{i=1}^{k} \|\lambda p_i\|$ would be true.*

Proof 1 *1. If p and the next element are in the form px then $\lambda(p) = \{px\}$ would be true. In this case $|p| = x$ is valid.*

2. Let $p\{\} = p \bullet \{r\{\}, s\{\}\}$ be true. Now assume $\|r\| = (\bigvee_{i=1}^{m} \|\lambda r_i\|)$ and $\|s\| = (\bigvee_{j=1}^{n} \|\lambda s_j\|)$ is true where $r\{\} = \{\lambda r_1, .., \lambda r_m\}$ and $s\{\} = \{\lambda s_1, .., \lambda s_n\}$ is true. In this case we find

$$\begin{aligned}
\|p\| &= \|r\|\|s\| \\
&= (\bigvee_{i=1}^{m} \|\lambda r_i\|)(\bigvee_{j=1}^{n} \|\lambda s_j\|) \\
&= \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \|\lambda r_i\|\|\lambda s_j\|
\end{aligned}$$

On the other hand for any $i = 1, .., m$ and $j = 1, .., n$, we know $\lambda p_{ij} = p \bullet \{\lambda r_i, \lambda s_j\}$ is true. From here we deduce $\|\lambda p_{ij}\| = \|p \bullet \lambda r_i\|\|p \bullet \lambda s_j\| = \|\lambda r_i\|\|\lambda s_j\|$. Then

$$\begin{aligned}
\|p\| &= \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \|\lambda r_i\|\|\lambda s_j\| \\
&= \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \|\lambda p_{ij}\|
\end{aligned}$$

is found.

3. Let p and the next element are in the form $p \dashv (r,s)$. Now assume $\|r\| = (\bigvee_{i=1}^{m} \|\lambda r_i\|)$ and $\|s\| = (\bigvee_{j=1}^{n} \|\lambda s_j\|)$ is true where $r\{\} = \{\lambda r_1, .., \lambda r_m\}$ and $s\{\} = \{\lambda s_1, .., \lambda s_n\}$ is true. In this case, since $\|p\| = \|r\| \vee \|s\|$ is true, we find
$\|p\| = \|r\| \vee \|s\| = \|\lambda r_1\| \vee .. \vee \|\lambda r_m\| \vee \|\lambda s_1\| \vee .. \vee \|\lambda s_n\|$
We know for any $i = 1, .., m$ and $j = 1, .., n$, $\lambda p_i = p \dashv \lambda r_i$ and $\lambda p_{m+j} = p \dashv \lambda s_j$ is true. This means that $\|\lambda p_i\| = \|p \dashv \lambda r_i\| = \|\lambda r_i\|$ and $\|\lambda p_{m+j}\| = \|p \dashv \lambda s_j\| = \|\lambda s_j\|$ is true. From here

$$\begin{aligned}
\|p\| &= \|\lambda r_1\| \vee .. \vee \|\lambda r_m\| \vee \|\lambda s_1\| \vee .. \vee \|\lambda s_n\| \\
&= \|\lambda p_1\| \vee .. \vee \|\lambda p_m\| \vee \|\lambda p_{m+1}\| \vee .. \vee \|\lambda p_{m+n}\|
\end{aligned}$$

is found.

4.3 Symmetry in e-products

In a two dimensional formula, the A-term of x is the formula beginning from the reverse of the arc connected to x. Thus the sub formula scanned from x is equivalent to the conjunction of x and its A-term. For example in the figure, the subformula scanned is the conjunction of the A-term of $\widetilde{x}_2$ and $\widetilde{x}_2$ itself.

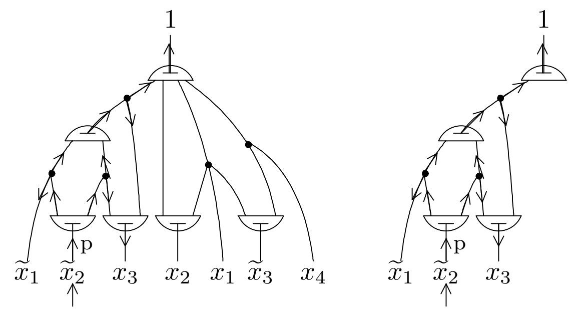
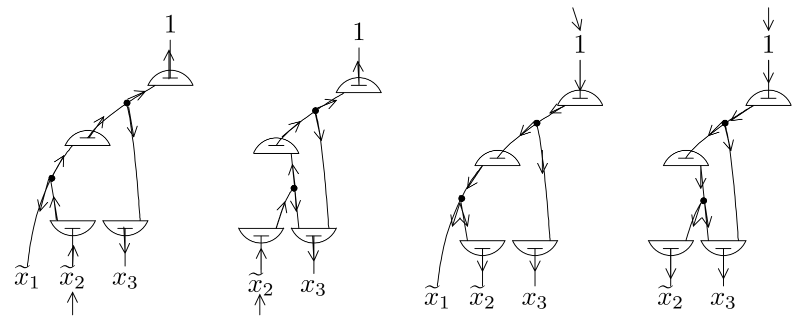


Figure 28: The subformula beginning from $\widetilde{x}_2$

On the other hand, this term could also be found from root. In a DNF of a formula every e-product includes either x_i or $\widetilde{x}_i$ or both x_i and $\widetilde{x}_i$. Therefore the x_i term must be equivalent to the disjunction of the e-products scanned beginning from the root and including x_i



(a) The e-products scanned from $\widetilde{x}_2$

(b) The e-products scanned from the root and includes $\widetilde{x}_2$

Figure 29: The e-products scanned from $\widetilde{x}_2$ and the e-products including $\widetilde{x}_2$ and scanned from the root should be equivalent

In an e-product there is a path from any terminal to another terminal and whichever terminal we begin to scan, we should scan the same e-product as shown in the figure.

Let write an e-product $\lambda a\langle u_1, .., u_n\rangle$ where $u_1, .., u_n$ are the terminals of the e-product λa. Then we can write the e-product shown in the figure as $\lambda\langle 1, \widetilde{x_1}, \widetilde{x_2}, x_3\rangle$. Indicate the scanning from a particular terminal by an arrow directing to that terminal. For example the four different scanning made from four different terminals can be shown as

$\lambda\langle \Rightarrow p'_1, dglx_1, r_1, x_3\rangle$

$\lambda\langle p'_1, \Rightarrow\widetilde{x}_1, r_1, x_3\rangle$

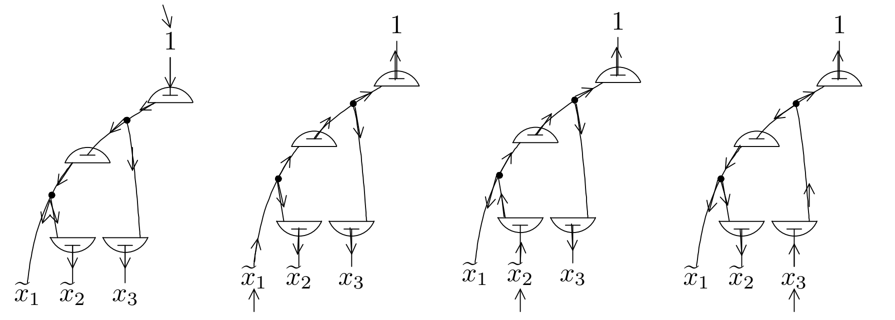
$\lambda\langle p'_1, \widetilde{x}_1, \Rightarrow r_1, x_3\rangle$

$\lambda\langle p'_1, \widetilde{x}_1, r_1, \Rightarrow x_3\rangle$

We can said different scanning of the same e-products.

However, now look at the example in the figure

Here while from k and $\widetilde{x_2}$, there is only one scan, from x_3 it is possible to do two distinct scan. We have mentioned that the selection of different base arches of an OR-node leads different e-products.



(a) $\lambda\langle \Rightarrow 1, \widetilde{x_1}, \widetilde{x_2}, x_3\rangle$ (b) $\lambda\langle 1, \Rightarrow\widetilde{x_1}, \widetilde{x_2}, x_3\rangle$ (c) $\lambda\langle 1, \widetilde{x_1}, \Rightarrow\widetilde{x_2}, x_3\rangle$ (d) $\lambda\langle 1, \widetilde{x_1}, \widetilde{x_2}, \Rightarrow x_3\rangle$

Figure 30: Whichever the starting terminal, the same e-product is scanned

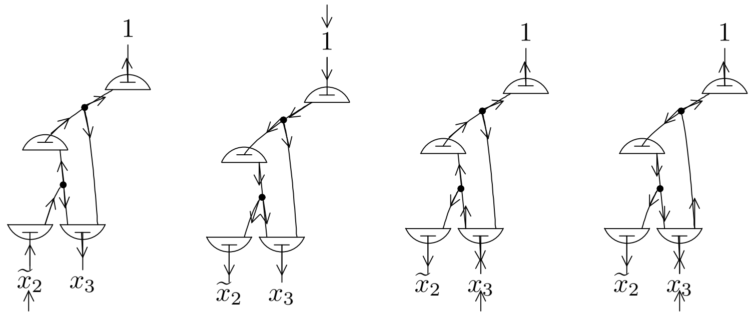


Figure 31: There are two scans starting from x_3 in this e-product

Then, is there two distinct e-product here? To see this is not true, let change the e-product into another form.

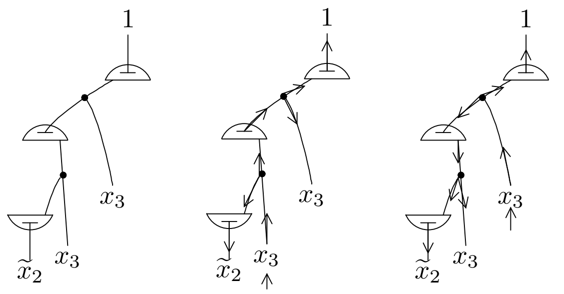


Figure 32: Another form of the e-product and its scanning

Therefore, selection of different base arches in an OR-node does not always lead to different e-products. From x_3 there are two scanning, but on the other hand, from the other terminals, there are two paths to x_3.

However as seen, to find an e-product, only a scan is enough. I.e. to find an e-product there is no need to do all possible scans.

Now we will show that all the terminals of an e-product takes the same value at every interpretation.

Theorem 2 *In an e-products, if all the terminals are literals, then the value of each terminal is the same.*

Proof 2 *Let $\{x_1, .., x_n\}$ be the terminals of an e-product. For any of x_i terminal, let p_i be the arc tied to it. In this case, the value of x_i terminal becomes $|x_i||p'_i|$. From p_i there is at least one path to all the terminals except x_i. Therefore, $\|p_i\| = \wedge_{j=1 j\neq i}^{n} x_j$ would be true. Then $x_i\|p_i\| =_{i=1}^{n} x_j$ would be true.*

This means that the value of an e-product whose all terminals are literals does not depend on the scan.

If one of the terminals is a 1 it is obvious that it does not affect the value of the e-product.

4.4 Regular Formulas

Then it is possible to scan all the e-products including x by starting from the root. However, this does not guarantee that all the e-products that can be found from x can also be found from the root. Two examples that demonstrate this are shown in the figure.

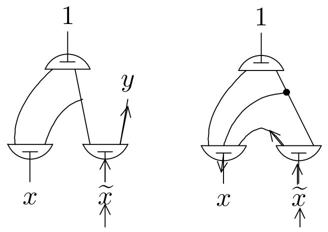


Figure 33: The e-products scanned from $\widetilde{x}$ but can not be scanned from the root

However, such formulas do not derived from a statement formula by the method given. Any formula derived by the given method does not include any e-product which can be scanned from a terminal but can not be scanned from the root.

Definition 6 *If the root is a terminal of every e-product in a formula then this is a regular formula.*

This definition in fact gives us a hint to define the root also.

Definition 7 *If any literal* 1 *is a terminal of every e-product in a formula then this is a root of the formula.*

Therefore, there can be more than one root of a formula. For example the following formula has three roots.

Now we will show that all the roots are equivalent.

Theorem 3 *All the roots gets the same value at the same interpretation.*

Proof 3 *Let the roots be $k_1, .., k_n$. This means that every product of the formula has $k_1, .., k_n$ as terminals. For any e-product every terminal of it is equivalent. I.e. every terminal gets the same value at the same interpretation. Since any of the root is the disjunction of all the e-products, all the roots are equivalent.*

Then the value of a formula can be found from any of its roots. On the other hand, new roots can be created by adding arches to all the e-products and taking the disjunction of them.

In fact we can create an equivalence of the sample formula and show its values in some interpretations as below.

In addition to the interpretation rules, another rule is given below.

An additional rule for the interpretation: If p and the next element are in the form pxr then $|p| = |x||r|$.

This lead us to express a formula in a more simple way, because we don't have to draw the roots anymore. For example the sample formula can be shown as in the figure.

In this article, we will always draw the formulas with their roots.

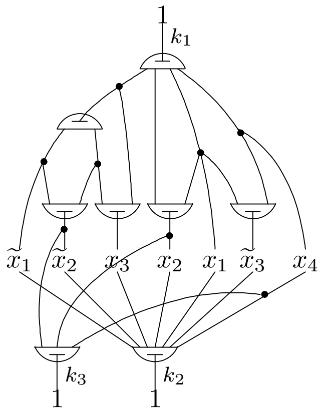
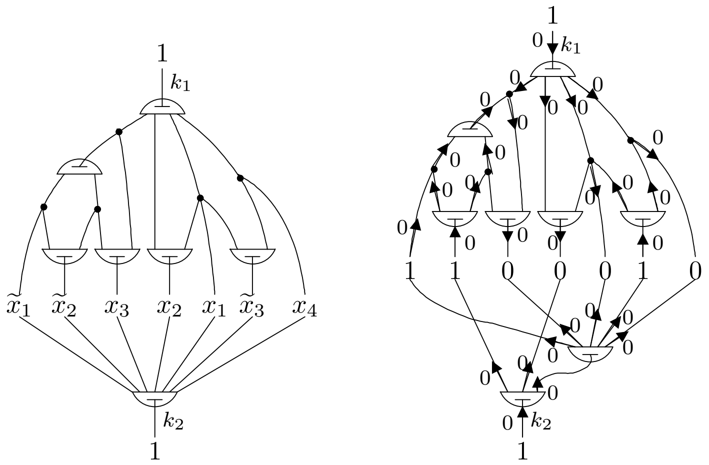


Figure 34: A formula can have more than one root



(a) An equivalence of the example formula

(b) For $(x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0)$ $|k_1| = |k_2| = 0$

Figure 35: The formulas shown from k_1 and k_2' are equivalent

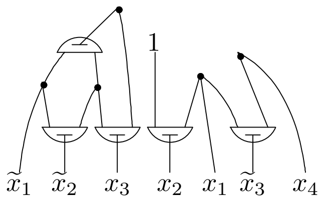


Figure 36: An equivalence of the example formula where the roots are removed

4.5 Finding E-products in Reduced formulas

As shown in the figure, there are paths from x to x' and from x' to x in an e-product including both x and x'. However, after an x reduction, some closed paths appear.

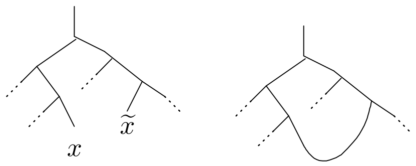


Figure 37: An invalid e-product produce closed paths when reduced

For example, in the following figure, an invalid e-product of the two dimensional form of the statement formula $x \vee x \cdot \widetilde{x} \vee \widetilde{x} \cdot y$ and the new form of the e-product after reduction are shown

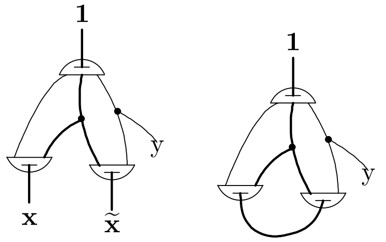
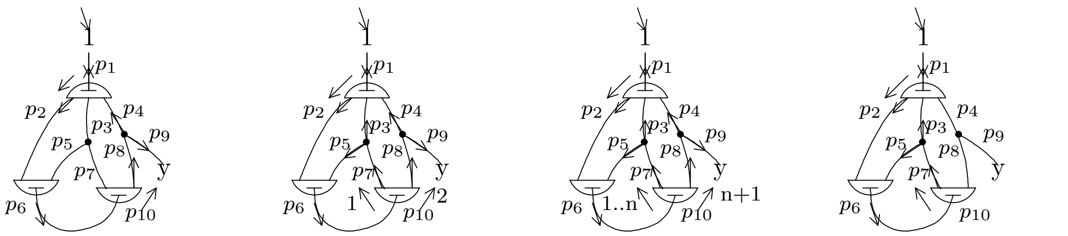


Figure 38: An invalid e-product and the reduced form of it

It seems that the closed paths indicate invalidity. If this is so, a way to finding whether a formula is a tautology or not is to examine every e-product in the last reduced form of a formula and find whether all have a closed path. If all of them have closed paths then we can decide that the formula is not a tautology. However, to be able to do this we have to know what is an elementary product and how to find one in a reduced formula. One way of doing this is to use the algorithm already given without any change on it.

An example In the figure, some of the e-product scans in a reduced formula are shown.



(a) An e-product; After p'_{10}, p'_8 is selected

(b) Another e-product: After the first time p'_{10} is passed p'_7 is selected, then p'_8 is selected in the second pass

(c) For the first n pass of p'_{10}, p'_7 is selected; for the $n+1$th pass, p'_8 is selected

(d) The e-product scan where for every pass of p'_{10}, the arc p'_7 is selected

Figure 39: Finding some of the e-products beginning from the arc p_1

In the figure, the arrows near the OR-nodes show the selections made on the nodes and the numbers near them show the sequence of the selections. As seen, there is no bound on the number of different e-products since an OR-node can be passed infinite number of times and in each pass, a different selection can be made.

After p_1 , p_2 is selected for all the e-product scans given in the figure. In the first scan, the arc p'_8 is selected after p'_{10} for the OR-node $p'_{10} \dashv (y'_8, p'_7)$. In the second scan, p'_7 is selected after p'_{10} and this leads to p_6 for the second pass. The arc p'_8 is chosen after passing p'_{10} for the second time,.

Infinite Paths As seen, the paths may be endless in a reduced formula. For example, in the last scan given in the figure, whenever the arc p'_{10} is encountered, the arc p'_7 is selected on the OR-node. Then there is a path which is $p_1.p_2.(p_6.p'_{10}.p'_7.p_5.)^\infty$ in the scan.

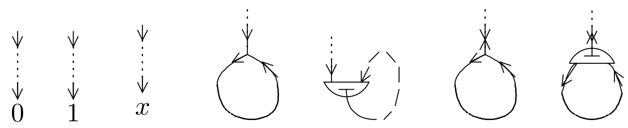
$(\underline{A})^\infty$ shows that the path $\underline{A}$ is repeated infinity number of times. The infinite paths can be called "closed" also. In general, $(X)^n$ shows that X is repeated n times. For this reason, the equation $AB^0C = AC$ is accepted as true.

A path can be beginless. For example the reverse of an endless path is begin-less. If a path is beginless or begins with a literal and endless or end with a literal then is is an exact path.

Each path is either infinite or terminate with a literal. In a path, a node or arc can be passed any number of times. Such a path is a cyclic path. We can classify the path types as follows.

Definition 8 *1. If a path is in the form ..p..p′.. then it is a returning path.*

2. If a path is in the form ..p..p.. it is a cyclic path.

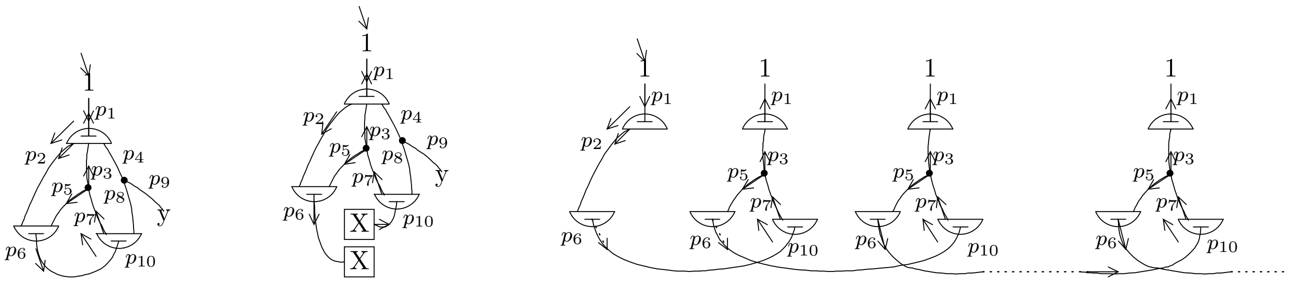


(a) paths end with a literal (b) cyclic paths (c) returning paths

Figure 40: Types of paths

A path can be both cyclic and returning. For example *..p..r..r′..p..* is both cyclic and returning.

An e-product can have infinite number of paths. For example for the e-product where whenever p'_{10} is passed p'_7 is selected, the number of paths is infinite. To find these paths let write the e-product recursively as follows.



(a) (b) (c) expansion

Figure 41: $1p_1.p_2.p_6X$ and $X = p'_{10}.p'_7.\{p_5.p_6X, p'_3.p'_1 1\}$

$$
\begin{aligned}
X &= p'_{10}.p'_7.\{p'_3.p'_1 1, p_5.p_6 X\} \\
&= \{p'_{10}.p'_7.p'_3.p'_1 1, p'_{10}.p'_7.p_5.p_6 X\}
\end{aligned}
$$

From here

$$
\begin{aligned}
X &= \{p'_{10}.p'_7.p'_3.p'_1 1, p'_{10}.p'_7.p_5.p_6 X\} \\
&= \{\underline{A}, \underline{B}X\} \\
&= \{\underline{A}, \underline{B}\{\underline{A}, \underline{B}X\}\} \\
&= \{\underline{A}, \{\underline{BA}, \underline{BB}X\}\} \\
&= \{\underline{A}, \underline{BA}, \underline{BB}X\} \\
&= . \\
&= . \\
&= \{\underline{A}, \ddot{\bigcup}_{i=1} \underline{B}^i \underline{A}, \underline{B}^\infty\} \\
&= \{p'_{10}.p'_7.p'_3.p'_1 1, \{\ddot{\bigcup}_{i=1} (p'_{10}.p'_7.p_5.p_6.)^i p'_{10}.p'_7.p'_3.p'_1 1\}, (p'_{10}.p'_7.p_5.p_6)^\infty\}
\end{aligned}
$$

Here $\ddot{\bigcup}_{i=1}$ means from $i = 1$ to infinity but not including infinity. In general, for $l >= k$, $\{\bigcup_{i=k}^{l} \underline{AB}^i\underline{C}\} = \{\underline{AB}^k\underline{C}, \underline{AB}^{k+1}, .., \underline{AB}^l\underline{C}\}$ is valid. For the obvious reasons $\underline{AB}^\infty\underline{C} = \underline{AB}^\infty$ must be valid. In addition, $\underline{AB}^0\underline{C} = \underline{AC}$ is accepted as valid. In this case, the above set can be written as

$\{\bigcup_{i=0}^{\infty} (p'_{10}.p'_7.p_5.p_6)^i p'_{10}.p'_7.p'_3.p'_1 1\}$

Then

$X = \{\bigcup_{i=0}^{\infty} 1p_1.p_2.p_6.(p'_{10}.p'_7.p_5.p_6)^i p'_{10}.p'_7.p'_3.p'_1 1\}$

would be true.

E-products and the Value of Reduced Formulas We have proved that in an initial formula, the disjunction of the values of the e-products are equal to the value of the formula. However, we can not use an induction method alone to prove a similar theorem for reduced formulas because there may be no termination and therefore initial conditions may not be set. However, with the following logic we feel that it is also true for the reduced formulas. Now think of $..px$ and $\widetilde{x}r..$ in unreduced formula. We know that, for the formulas of the arches p and r, the property "the value of the formula is equal to the disjunction of the values of all the e-products of it" is true. p has this property because of the form px. With reduction it becomes $..p.$ş$.r..$ and if r has this property, then p will continue to have this property by induction. However, there is no reason to assume that this property is no longer true for r after reduction. I.e. the reduction does not change this property for any of the arches in the formula[1]. So this means that this property is still be true for the root. However, whether this logic is actually true or not is not important while finding whether a formula is a tautology or not. We are totally free to make any assumptions and the only criteria here is that whether the reduction preserves the tautological property of the formula under the given assumptions.

Axiom 1 *In any two dimensional formula, the value of the formula is equal to the disjunction of the values of all the e-products of it.*

[1]Maybe this can be accepted as a new continuation rule

4.6 Values of Infinite Paths

We have said that the value of an e-product is equal the conjunction of the values of its paths and the values of a path is equal to the value of its last element.

$$\begin{aligned} \|A_i\| &= \text{the last element of } a_i \\ \lambda\{A_1, .., A_n\} &= \|A_1\| \cdot .. \cdot \|A_n\| \end{aligned}$$

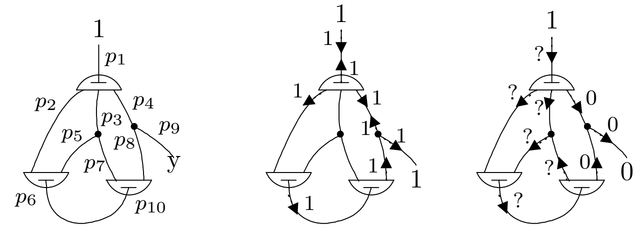
However, the infinite or closed paths does not have a last element. Then what are the values of them?

To find what would be the value of an infinite path, let consider the formula given in the figure.

The statement equivalence of the sample formula is $x \vee x \cdot \widetilde{x} \vee \widetilde{x} \cdot y$. If we do an x reduction, we get

$$\begin{aligned} x \vee x \cdot \widetilde{x} \vee \widetilde{x} \cdot y \ &\xrightarrow{x}\ (0 \vee 0 \cdot 1 \vee 1 \cdot y) \cdot (1 \vee 1 \cdot 0 \vee 0 \cdot y) \\ &= y \end{aligned}$$

Then this would be the statement equivalence of the two dimensional formula when reduced.In the following figures, we show the value of the sample formula for different interpretations.

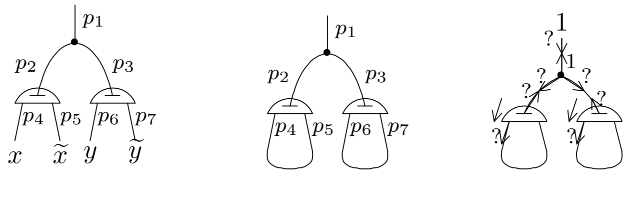


(a) Sample reduced formula (b) For $y = 1$ (c) For $y = 0$

Figure 42: What are the values of the arches on the closed path $p_1.p_2.(p_6.p'_{10}.p'_7.p_5.)^\infty$? The values of the arches which do not affect the result are not shown

So as seen, the values of arches on a closed path can not be find directly. However, for the result be true, the question marks must be replaced by 0. From here we conclude that the value of this closed path must be 0.

On the other hand, consider the following example.



(a) The sample formula (b) The reduced formula (c) An interpretation?

Figure 43: What are the values of the arches on the closed path $(p_1.(p_2.p_4.p'_5.p'_2.p_3.p_6.p'_7.p'_3.)^\infty$?

The statement equivalence of the original formula is $(x \vee \widetilde{x}) \cdot (y \vee \widetilde{y})$ and we get 1 when reduced. However, as seen, in the reduced two dimensional form, there is a closed path and again the value of this path can not be found directly. But, to make the result be true, it must be 1. So the value of this closed path must be 1.

However, there is a difference between the first and second closed paths. In the first example, the closed path is not a returning path, but the second closed path is a returning path. So it seems that returning in a path is an important property of that path.

Axiom 2 *The value of an unidirectional infinite path is* 0

Axiom 3 *The value of an infinite path in the form* $(\underline{A})^\infty$ *is* 1 *where* $\underline{A}$ *is a returning path.*

4.7 Analyzing E-products

E-products We know that in the statement formulas, the e-products in the reduced formulas are composed from the e-products of the unreduced ones as follows.

$(a_1 \vee .. \vee a_k) \cdot x \vee (b_1 \vee .. \vee b_l) \cdot \widetilde{x} \vee (c_1 \vee .. \vee c_m) \cdot x \cdot \widetilde{x} \vee d_1 \vee .. \vee d_n$

$\xrightarrow{x} (a_1 \vee .. \vee a_k) \cdot (b_1 \vee .. \vee b_l) \vee d_1 \vee .. \vee d_n$

It should be true for two dimensional formulas too.

To scan an e-product, it is not necessary to begin from root and scan the rest in a sequential order. In the following, a scan first beginning from p and then beginning from the root are shown. The scan starting from the root stops at p. This scan can be seen in another view. Here first an e-product beginning from p is scanned and this is removed from the e-product beginning from k. Let this be λa. Then the rest of the e-product is scanned starting from the root. As a result, the whole e-product is scanned.

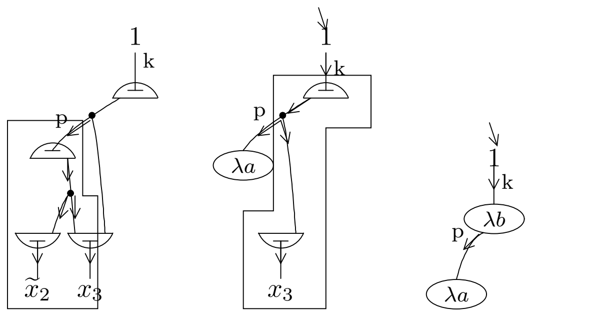
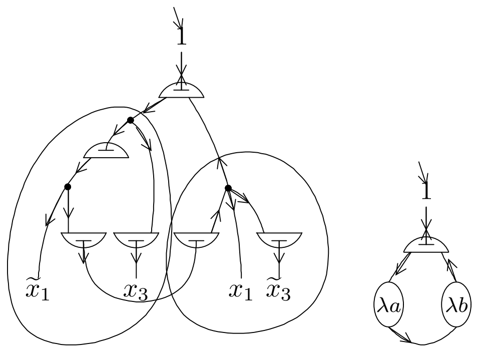


Figure 44: An e-product can be scanned seperately

Here λb is an elementary scan starting from k and end at p. Thus we expand the concept of e-product by accepting that an e-product can have arches as leaves.

The e-products of the reduced formulas are made of from the e-product of unreduced ones.

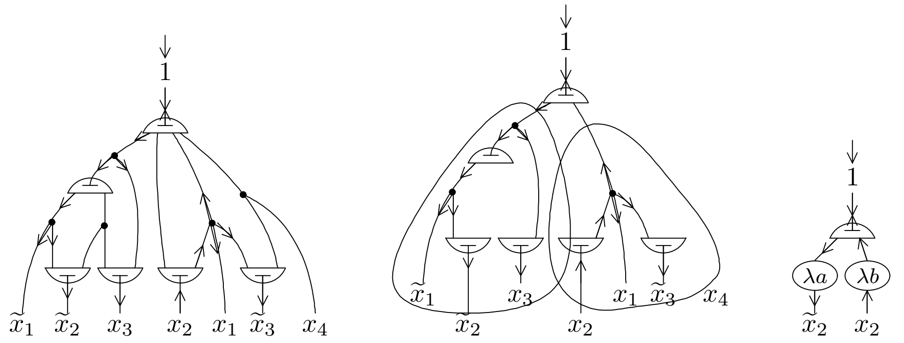


(a) In a reduced formula a scanning and symbolizing

Figure 45: A scan in a reduced formula can be seen as a composition of two scans in the unreduced formula

This means that the e-products in reduced formulas is included by the given algorithm. So the given e-scan algorithm can be used directly.

Any e-product including $p..rx$ path can be seen as the composition of an e-product which include $p..r$ path and rx path. For example, the following e-product is composed of an e-product with open terminal and $r\widetilde{x}_2$



(a) The scanning of two products and symbolizing

Figure 46:

Open terminal e-products If a terminal is end with an arc this is an open terminal, if it ends with a literal then we call this as a real terminal.

Then in an e-product there can be three types of terminals.

- Real terminals: The terminals with a literal.
- open terminals: the terminals with an arc
- Infinite terminals: the infinite paths are accepted as terminated with "infinite terminals"

An e-product can be shown as $\lambda t\langle a_1,..,a_m,s_1,..,s_n,x_1,..,x_o\rangle$ or $\lambda t\langle a_1,..,a_m\rangle\langle s_1,..,s_n\rangle\langle x_1,..,x_o\rangle$ where $a_1,..,a_m$ are open terminals, $s_1,..,s_n$ are infinite terminals and $x_1,..,x_o$ are real terminals.

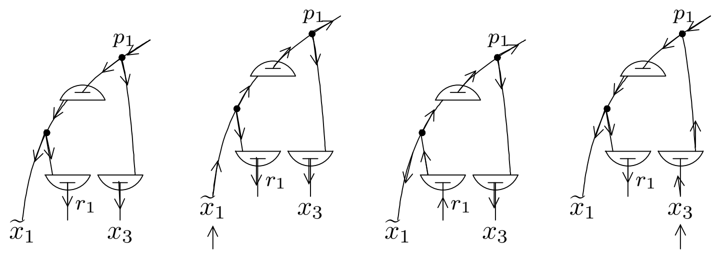


Figure 47: Scannings in an open terminal e-product

Theorem 4 *In an e-product all the real terminals are equivalent.*

Proof 4 *Let $\langle x_1,..,x_m\rangle$ be the real terminals and $\langle y_1,..,y_n\rangle$ be all the other terminals of an e-product. Let for any terminal x_i, p_i be the arc connected to this terminal. I.e. there is a path $..p_ix_i$ in the e-product. It is known that there is at least one path from x_i to all the other terminals. So from p_i', there is always a path to all the terminals except x_i. This means that $p_i' \cong \bigwedge_{j=1,j\neq i}^{m} x_j \cdot \bigwedge_{j=1}^{n} y_j$. Thus for the scan starting from x_i, $x_i \cdot \|p_i'\| \cong \bigwedge_{j=1}^{m} x_j \cdot \bigwedge_{j=1}^{n} y_j$ would be true.*

Theorem 5 *Let $\langle a_1,..,a_m\rangle$ be the open terminals and $\langle b_1,..,b_n\rangle$ be all the other terminals of an e-product λt. Then $\lambda t\langle a_1,.,\rightarrow a_i,..,a_m\rangle\langle b_1,..,b_n\rangle \cong (\bigwedge_{j=1,j\neq i}^{m} a_j)\cdot(\bigwedge_{j=1}^{n} b_j)$ would be true.*

Proof 5 *Trivial.*

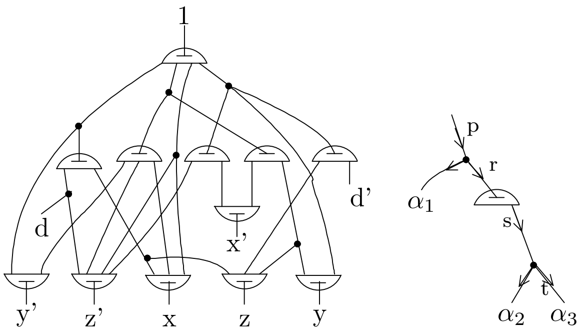


Figure 48: A product determined by the path $\gtrdot p.r.s.t$ and its abbreviation

Paths, products, e-products We can talk about the formulas determined by a path.

let show a product determined by a path $\underline{A}$ as $\Lambda(\underline{A})$. For example the product determined by $p.r.s.t$ is written as $\Lambda(p.r.s.t)$.

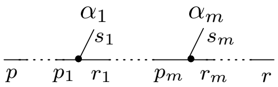
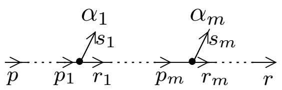


Figure 49:

Theorem 6 *$\Lambda(p\underline{A}r)$ be a product. Then $|\gtrdot\Lambda(\underline{A})| = X \cdot |r|$ and $|p\Lambda(\underline{A})r\lessdot| = X \cdot |p'|$ be true.*

Proof 6 *Let $p\underline{A}r = p\underline{\ddot{O}}_1 p_1 \bullet r_1 \underline{S}_1 \bullet .. \bullet \underline{\ddot{O}}_m p_m \bullet r_m \underline{S}_m r$ be true. Let there is no AND-node in $\underline{\ddot{O}}_i$ and $\underline{S}_i$. Let for each $p_i \bullet r_i$, the arc forked from this AND-node be s_i. I.e. The AND-node is in the form $p_i \bullet (s_i, r_i)$. In this case, $|\gtrdot\Lambda(\underline{A})|$ is found as follows.*

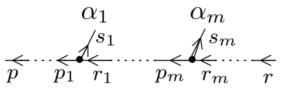
$$
\begin{aligned}
|\gtrdot\Lambda(p\underline{A}r)| &= |p| \\
&= |p_1| \\
&= |s_1| \cdot |r_1| \\
&= |s_1| \cdot |p_2| \\
&= |s_1| \cdot (|s_2| \cdot |r_2|) \\
&= . \\
&= . \\
&= |s_1| \cdot (|s_2| \cdot (...(|s_m| \cdot |r_m|)..) \\
&= |r_1| \cdot (|r_2| \cdot (...(|s_m| \cdot |r|)..) \\
&= |s_1|.|s_2|....|s_m|.|r|
\end{aligned}
$$



On the other hand, $r'\underline{A}'p' = r'\underline{S}'_m r'_m \bullet p'_m \bullet .. \bullet \underline{S}'_1 r'_1 \bullet p'_1 \underline{\ddot{O}}'_1 p'$ would be true and for each $r'_i \bullet p'_i$, the arc forked from this AND-node would be s_i. .e. the AND-node is in the form $r'_i \bullet (p'_i, s_i)$. In this case, similarly,

$$|\gtrdot\Lambda(r'\underline{A}p')| = |s_1|.|s_2|....|s_m|.|r|$$

is found.



I.e. $|p| = X \cdot |r|$ *and* $|r'| = X \cdot |p'|$ *would be true where* $X = |s_1|.|s_2|....|s_m|$

The product determined by $p\underline{A}r$ can be seen as a set of e-product which have $p\underline{A}r$. Any e-product having $p\underline{A}r$ can be seen as a part of the product determined by $p\underline{A}r$.

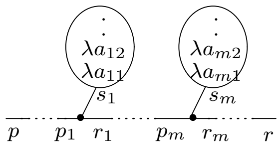


Figure 50: The product determined by $p\underline{A}r$ can be seen as a set of e-product which have $p\underline{A}r$

Now let show that there is always a scan of an e-product which starts one of the terminals and gives the e-product.

Theorem 7 *Let there is an e-product scan from p. As a result of this, let the terminals* $u_1, .., u_n$ *be found. There is always an e-product scan starting from* u_i *and giving the same e-product.*

Proof 7 *As seen in the figure, if there is a path* $u_1..u_2$ *then there is also a path* $u'_2..u'_1$.

Figure 51:

Symbolizing In fact, not only formulas, but we can say the e-products of sub formulas too. The sub formulas may be any part of the formula. In this case we can think of a sub formula as a composition of sub paths. For example the following is a sub formula and its e-products.

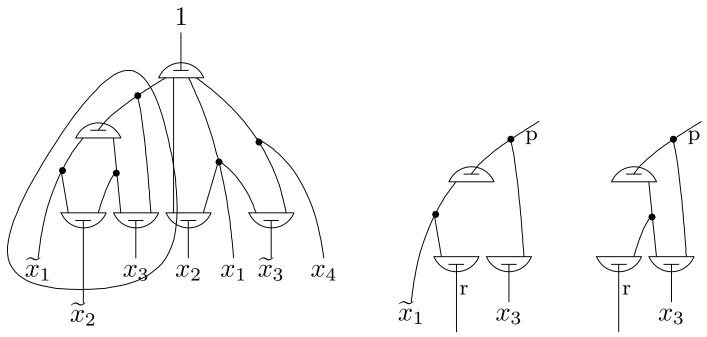


Figure 52: A sub formula and its e-products

Then can we symbolize an equivalent of the sample formula as follows.

Here λa_1 and λa_2 are the e-products whose open terminals are p'_1 and r_1, λb_1 and λb_2 are the e-products whose open terminals are p'_2 and r_2. λc is the e-product with open terminal p'_3. Thus λa_1 can be written as $\lambda a_1\langle p'_1, r_1\rangle$ where other terminals of λa_1 are not indicated. However, writing this

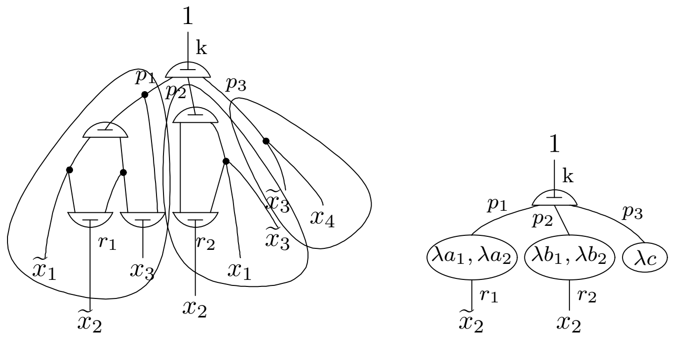
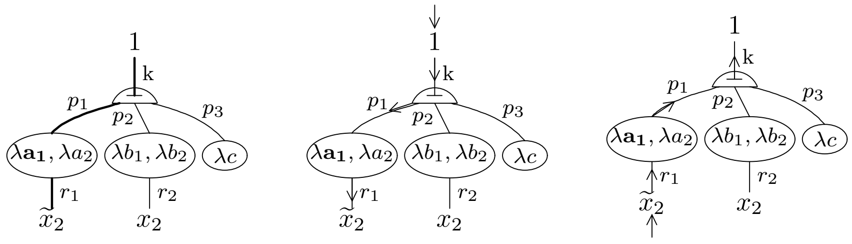


Figure 53: A formula and symbolizing e-products

as $p_1\lfloor\lambda a_1\rfloor r_1$ will make life easier. I.e. we accept $p_1\lfloor\lambda a_1\rfloor r_1 \equiv \lambda a_1\langle p_1', r_1\rangle$ as true. In this case, we can write this e-product from reverse as $r_1'\lfloor\lambda a_1\rfloor p_1'$. The scan done from an terminal is shown by an arrow directed to that terminal. For example the scan of this e-product from the arc p_1 is written as $\Rightarrow p_1\lfloor\lambda a_1\rfloor r_1$ or $r_1'\lfloor\lambda a_1\rfloor p_1'\Leftarrow$.



(a) $1k.p_1\lfloor\lambda a_1\rfloor r_1\widetilde{x_2}$ (b) $\Rightarrow 1k.p_1\lfloor\lambda a_1\rfloor r_1\widetilde{x_2}$ (c) $1k.p_1\lfloor\lambda a_1\rfloor r_1\widetilde{x_2}\Leftarrow$

Figure 54: An e-product and scanning of it

We need to show the e-products in a statement form. Here one of the e-products is formed by joining the path $1k.p_1$, the e-product $p_1\lfloor\lambda a_1\rfloor r_1$ and the path r_1x.

Definition 9 *Let $A = \{p_1, .., p_m\}$ and $B = \{r_1, .., r_n\}$ be two path sets. Then we define $A \sqcup B$ as the following.*

1. *If $\underline{A}s \in A$ and $s\underline{B} \in B$ then $\underline{A}s\underline{B} \in A \sqcup B$*
2. *If $\underline{A}s \in A$ and there is no path in the form $s..$ in B then $\underline{A}s \in A \sqcup B$*

In this case we can write the e-product shown in the figure as $1k.p_1 \sqcup p_1\lfloor\lambda a_1\rfloor r_1 \sqcup r_1x$.

4.8 A general formula

Now consider a general formula such as the following

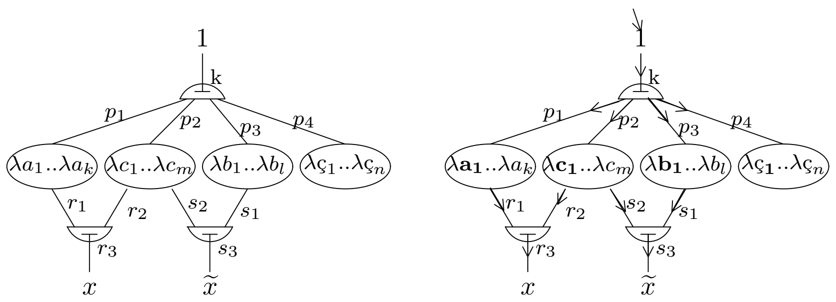
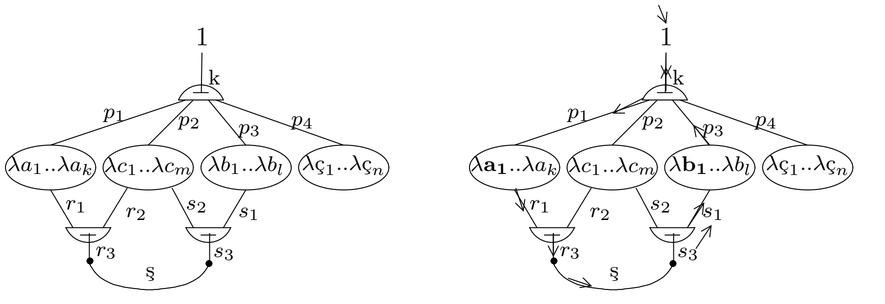


Figure 55: A general formula and some e-product scans

Write this formula as
$1k.\{p_1\lfloor\alpha\rfloor r_1.r_3x,\ p_2\lfloor\lfloor\delta\rfloor r_2.r_3x\rfloor s_2.s_3\widetilde{x},\ p_3\lfloor\beta\rfloor s_1.s_3\widetilde{x},\ p_4\lfloor\gamma\rfloor\}$
or
$1k.\{p_1\lfloor\alpha\rfloor..x,\ p_2\lfloor\lfloor\delta\rfloor..x\rfloor..\widetilde{x},\ p_3\lfloor\beta\rfloor..\widetilde{x},\ p_4\lfloor\gamma\rfloor\}$
where $\alpha = \{\lambda a_1, .., \lambda a_k\}$, $\beta = \{\lambda b_1, .., \lambda b_l\}$, $\delta = \{\lambda c_1, .., \lambda c_m\}$ and $\gamma = \{\lambda ç_1, .., \lambda ç_n\}$.
Notice that the e-products here has only real or open terminals since this is an unreduced formula.
Now reduce the formula and try to find e-products of it.



(a)

(b)
$\twoheadrightarrow 1k.p_1\lfloor\lambda a_1\rfloor r_1.r_3.§.s_3'.s_1'\lfloor\lambda b_1\rfloor p_3'.k'1$

Figure 56: Reduced formula and a scan of an e-product

Here x and $\widetilde{x}$ are replaced by an AND-node and the arc § is put in between. Now the e-products λa_i, λb_j and λc_k are vanished and and new e-products which are composition of these ones are produced. However, the $\lambda ç_l$ does not vanish and they are still the e-products of the reduced formula.

Some of the e-scans are shown below.

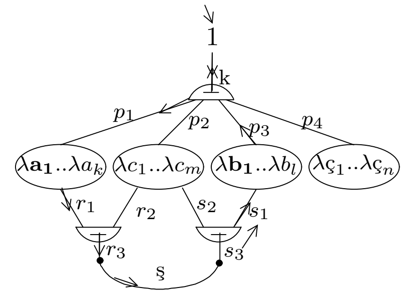
As seen the number of e-products thus scanned is infinite.

Returning Paths and E-products Unfortunately, finding e-products is not as simple as seen above. In an e-product, there can be a returning path as follows.

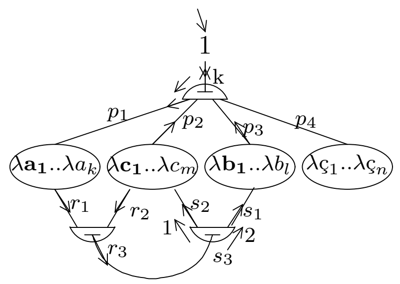
The possibility of a returning path in an e-product increases the number of e-products that can be found. In the figure, some of the e-product scans are shown. Here the e-products $\lambda ç_1, .., \lambda ç_n$ are not shown because they are not change with reduction.

Here in λb_1 there is at least one $s_1'..s_1$ path. I.e. there is more than one s_1 terminal. In the figure, the existence of such path are indicated by an + sign under the e-products text.

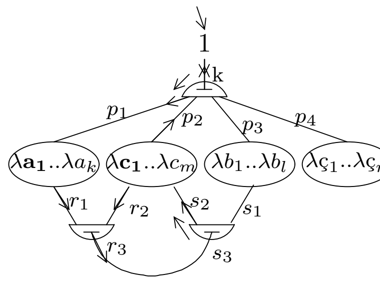
As seen there are e-product scans in addition to the found in previous figure.



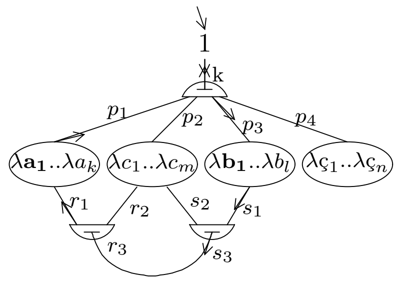
(a) $\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..s'_1\lfloor\lambda b_1\rfloor p'_3..1$



(b) $\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..s'_2\lfloor\lambda c_1 p'_2..1\rfloor r_2..s'_1\lfloor\lambda b_1\rfloor p'_3..1$



(c) $\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..(s'_2\lfloor\lambda c_1 p'_2..1\rfloor r_2..)^\infty$



(d) $\Rightarrow 1..p_3\lfloor\lambda b_1\rfloor s_1..r'_1\lfloor\lambda a_1\rfloor p'_1..1$

Figure 57: Some e-sacns in reduced formula

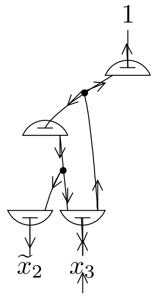
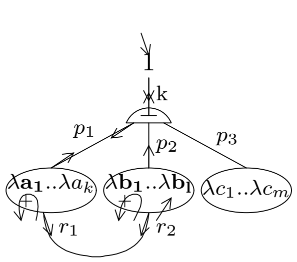
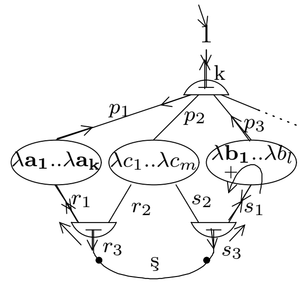


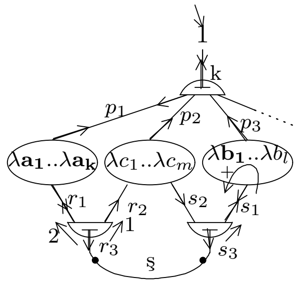
Figure 58: A path in an e-product can return



(a)
$\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..\S..s'_1\lfloor\lambda b_1..1\rfloor s_1..\S$
$..r'_1\lfloor\lambda a_1\rfloor p'_1..1$

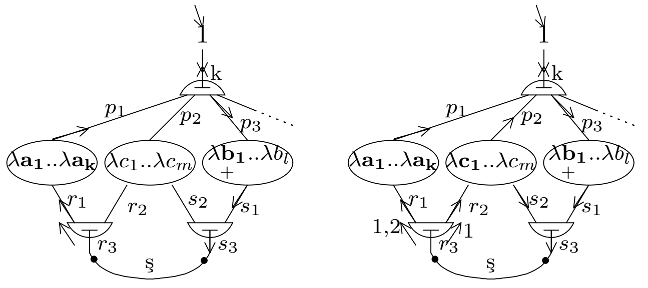


(b)
$\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..\S..s'_1\lfloor\lambda b_1..1\rfloor s_1..\S$
$..r'_1\lfloor\lambda a_k\rfloor p'_1..1$



(c)
$\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..\S..s'_1\lfloor\lambda b_1..1\rfloor s_1..\S$
$..r'_2\lfloor\lambda c_1..1\rfloor s_2..\S..r'_1\lfloor\lambda a_k\rfloor p'_1..1$

Figure 59: Some e-product scans in a reduced formula where λb_1 has a path $s'_1..s_1$



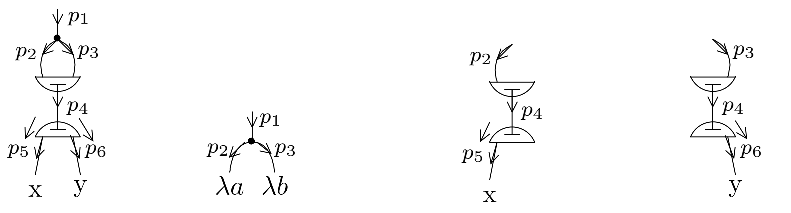
(a) $\Rightarrow 1k.p_3\lfloor \lambda b_1 s_1..\S' \sqcup$
$\S'..r'_1\lambda a_1 p'_1..1\rfloor s_1..\S' \quad \sqcup$
$\S'..r'_1\lfloor \lambda a_k\rfloor p'_1..1$

(b)

Figure 60: Some other e-product scans

Now let scan it on the other way.

In the second scan, both of the arches r'_1 and r'_2 are selected. See how this is possible in the example below.



(a) A scan and symbolizing

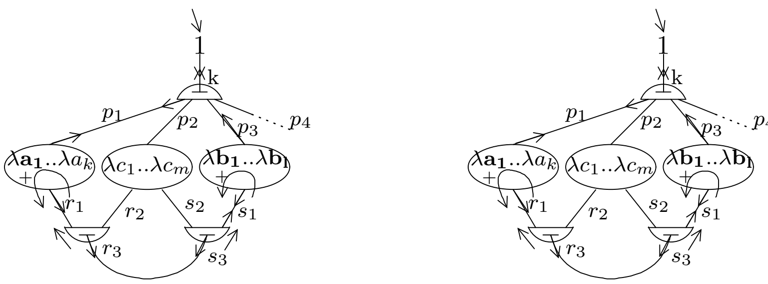
(b) λa

(c) λb

Figure 61: Such a scan is possible with the given algorithm

It is possible to do a scan given above with the algorithm given. Here the paths $\Rightarrow p_1.p_2.p_4.p_5x$ and $\Rightarrow p_1.p_3.p_4.p_6y$ belong to the same scan because in the given algorithm, after the paths are separated from an AND-node, the selection of outgoing arches in OR-nodes are independent from each other.

On the other hand if both of the e-products joined has returning paths, then the situation becomes more complex.



(a) $1..p_1\lfloor\lambda a_1\rfloor r_1..\S..s'_1\lfloor\lambda b_1..1\rfloor s_1$
$..\S'..r'_1\lfloor\lambda a_1..1\rfloor r_1..\S..s'_1\lfloor\lambda b_l\rfloor..1$

(b) $1..p_1\lfloor\lambda a_1\rfloor r_1..$
$(\S..s'_1\lfloor\lambda b_1 p'_3..1\rfloor s_1..\S'..r'_1\lfloor\lambda a_1..1\rfloor r_1..)^\infty$

Figure 62: Some e-product scans when two or more joined e-products have returning paths

As seen the number of e-products created by the returning products is infinite. Then how can we find the value of the reduced formula?

Fortunately, the fact that the e-products of a reduced formula are composed from the e-products of the unreduced formula helps us to find the value of the formula. I.e. the values of the e-products of the reduced formula can be expressed in terms of the values of the e-products of the unreduced one.

For example $1..p_1\lfloor\lambda a_k\rfloor r_1..s'_1\lfloor\lambda b_l\rfloor p'_3..1$ is an e-product formed by λa_k and λb_l. On the other hand, there are infinite number of e-products that include $p_1\lfloor\lambda a_1\rfloor r_1$ and $s'_1\lfloor\lambda b_1\rfloor p'_3$.

The e-products containing λa_i and λb_j may contain other e-products. For example

$$\Rightarrow 1..p_1\lfloor\lambda a_1\rfloor r_1..\text{ş}..s'_1\lfloor\lambda b_1..1\rfloor s_1..\text{ş}..r'_2\lfloor\lambda c_1..1\rfloor s_2..\text{ş}..r'_1\lfloor\lambda a_k\rfloor p'_1..1$$

contains $r_2\lfloor\lambda c_1..1\rfloor s_2$ and $r'_1\lfloor\lambda a_k\rfloor p'_1$ as well as $p_1\lfloor\lambda a_1\rfloor r_1$ and $s'_1\lfloor\lambda b_1\rfloor p'_3$.

In an e-product containing λa_i and λb_j, the terminals of both λa_i and λb_j are also the terminals of the e-product. The value of an e-product scan is the conjunction of its terminals. Therefore, we can write the value of an e-product containing λa_i and λb_j as $A \cdot B \cdot X$ where A and B are the conjunction of the values of the real terminals of λa_i and λb_j respectively. This means that the disjunction of all the e-products containing λa_i and λb_j would be $A \cdot B \cdot (X_1 \vee .. \vee X_k \vee ..)$.

Among the e-products containing λa_i and λb_j there is an e-product which contains λa_i and λb_j only. For example $1..p_1\lfloor\lambda a_1\rfloor r_1..(r'_2\lfloor\lambda b_1..1\rfloor r_2..r'_1\lfloor\lambda a_1..1\rfloor r_1)^\infty$ is an e-product formed by $p_1\lfloor\lambda a_1\rfloor r_1$ and $r'_2\lfloor\lambda b_1\rfloor p'_2$ only. If we could find that the value of it is $A_1 \cdot B_1$ where A_1 and B_1 are the conjunction of the values of the real terminals of $p_1\lfloor\lambda a_1\rfloor r_1$ and $r'_2\lfloor\lambda b_1\rfloor p'_2$ respectively, then we do not need to consider the other e-products containing $p_1\lfloor\lambda a_1\rfloor r_1$ and $r'_2\lfloor\lambda b_1\rfloor p'_2$ because, in this case, the value of the disjunction of all of these e-products would be $A_1 \cdot B_1 \cdot (1 \vee .. \vee X_k \vee ..) = A_1 \cdot B_1$.

The problem here is that whenever two returning paths are connected, then some infinite paths are created.

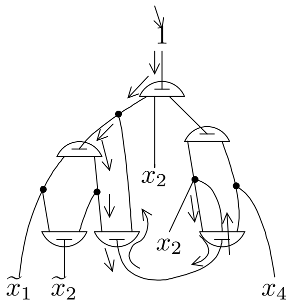


Figure 63: By connecting two returning e-products, some infinite paths appear

4.9 Another Approach to E-products

In unreduced formulas, we can though an e-product as a group of terminals which are connected to each other and A-term of each other.

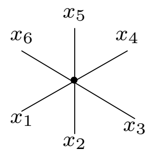


Figure 64: A group of A-terms where every terminal is a literal

We have said that if one term is reached from another term then they are an A-term of each other in statement formulas. However, reachability has transitive property. I.e. if x_2 is reached from x_1 and x_3 is reached from x_2 then x_3 is reached from x_1. This must be true for two dimensional formulas too.

In the following figure from x_1 $\widetilde{x}_3$ is reached and found directly. For this reason $\widetilde{x}_3$ is an A-term of x_1. On the other hand, from $\widetilde{x}_3$, x_4 can be reached. As a result of transitive property, x_4 should be reached from x_1 and therefore x_4 should be an A-term of x_1. In two dimensional formulas, the paths are exist to represent reachability. Therefore, there should be a path from x_1 to x_4.

Since x_4 is an A-term of x_1, there must be at least one e-product that contain both x_1 and x_4. However, the e-product scanning algorithm given before can not find such an e-product, because this algorithm does not consider this kind of transitive property. Therefore, a new scanning algorithm that consider it should be developed.

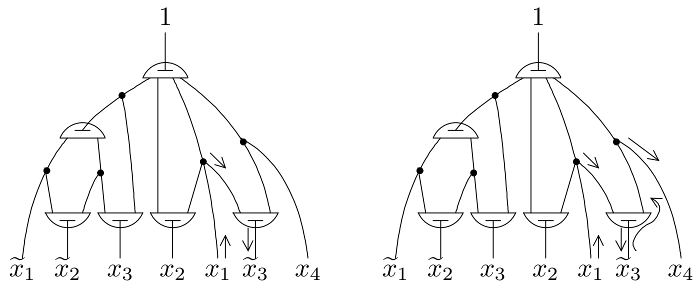


Figure 65: There should be a path from x_1 to x_4

A method to find such an e-product can be as follows: from a literal, scan an e-product with the given algorithm. Then from all the other literals found, scan other e-products and find the literals. Then again apply this method for the new literals found etc.

Let, for example, scan an e-product from x_4. The terminals of this e-product found are $\widetilde{x}_3$ and 1. They are the A-terms of x_4. Then scan an e-product from $\widetilde{x}_3$. As a result, 1, x_2 and x_1 are found as the A-terms of $\widetilde{x}_3$. However, since the A-terms of $\widetilde{x}_3$ are also the A-terms of x_4 because of transitive property, 1, x_2 and x_1 are also the A-terms of x_4.

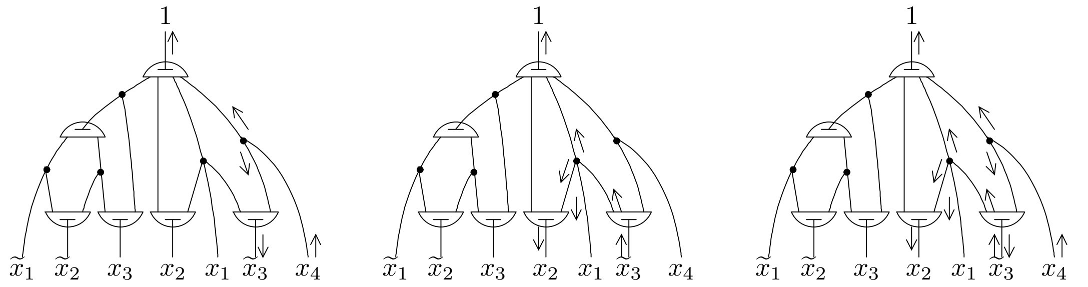
We can generalize this by the following algorithm. This is a slightly changed version of the old algorithm.

Algorithm 4.5 *1. start from the root of the formula*

2. Put the current element to TAR

3. repeat the following until TAR is empty.

(a) Pop an arc from TAR. let this be p.



(a) $\widetilde{x_3}$ and 1 are A-terms of x_4

(b) 1, x_2 and x_1 are A-terms of $\widetilde{x_3}$

(c) 1, $\widetilde{x_3}$, x_2 and x_1 are the A-terms of x_4

Figure 66: The A-terms of $\widetilde{x}_3$ are also the A-terms of x_4 because of transitive property

(b) *If this is an arc, follow the arc and push the element found to TAR.*

(c) *If this is an AND-node, put all the outgoing arches to TAR.*

(d) *If this is an OR-node then push one of the outgoing arches to TAR.*

(e) *If this is a literal, then put outgoing arc to TAR.*

4. *The subformula scanned is an e-product.*

The difference between this algorithm and the algorithm given already is that this algorithm[2] does not stop at terminals.

The General E-product Scanning Algorithm If we made that the returning is not only possible at terminals but possible on arches also, then the algorithm becomes as follows.

Algorithm 4.6 1. *This algorithm finds an e-product from p.*

2. *push the arc p to the stack.*

3. *repeat the following until the stack is empty.*

   (a) *pop an arc from the stack. Let it be p.*

   (b) *If p is the head arc of an OR-node, i.e. is in the form $p \dashv (r_1, .., r_n)$, then for one and only one $i = 1, .., n$, push r_i and p' to the stack.*

   (c) *If p is connected to an AND-node, i.e. is in the form $p \bullet (r_1, .., r_n)$, then for all $i = 1, .., n$, push all r_i and p' to the stack.*

   (d) *If p and the next element is in the form $p \vdash r$ or pr, then push r and p' to the stack.*

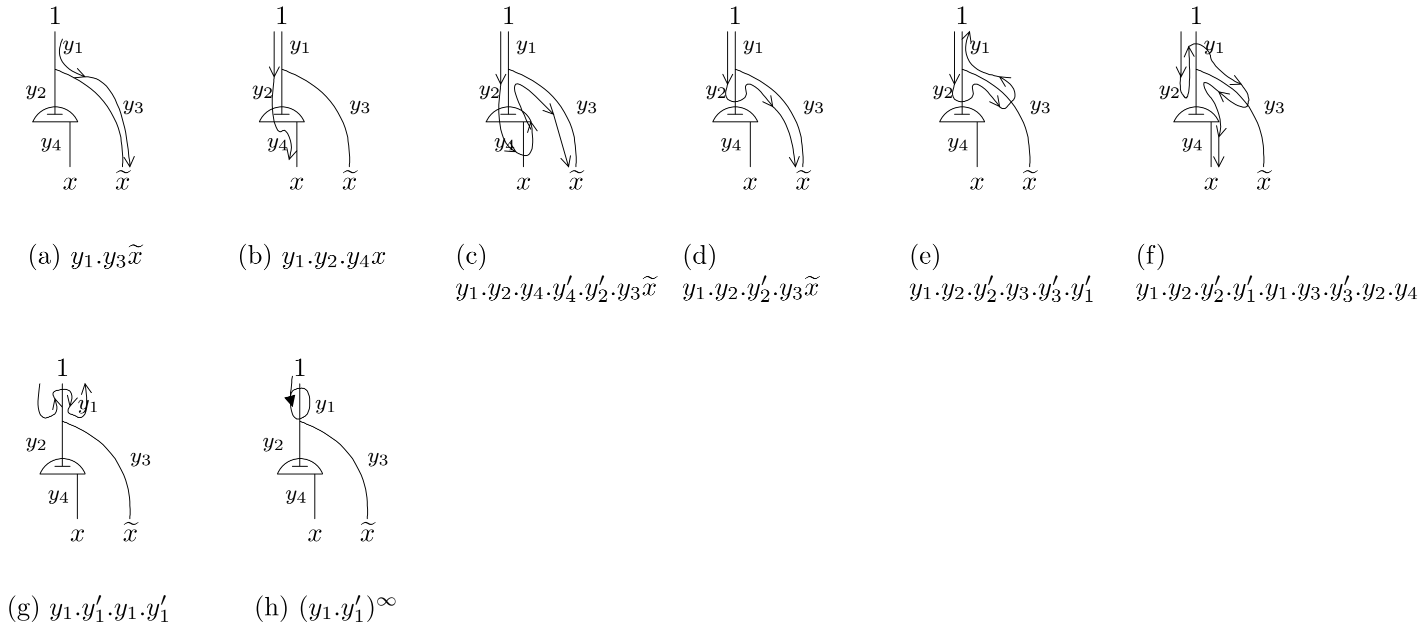
   (e) *If p and beyond is in the form px, then push push p' to the stack.*

4. *The subformula scanned is an e-product.*

Let call this as the general e-product scanning algorithm. The old algorithm is called normal e-product scanning algorithm. By this way, now it is possible to have paths in the form $..pp'...$ The figure illustrates some of the paths found in the sample formula.

Definition 10 *All of the paths in the form $..p.p'..$ are called immediately returning paths.*

[2] In this text, the term algorithm is used for all kind of procedures



(a) $y_1.y_3\widetilde{x}$

(b) $y_1.y_2.y_4x$

(c) $y_1.y_2.y_4.y_4'.y_2'.y_3\widetilde{x}$

(d) $y_1.y_2.y_2'.y_3\widetilde{x}$

(e) $y_1.y_2.y_2'.y_3.y_3'.y_1'$

(f) $y_1.y_2.y_2'.y_1'.y_1.y_3.y_3'.y_2.y_4$

(g) $y_1.y_1'.y_1.y_1'$

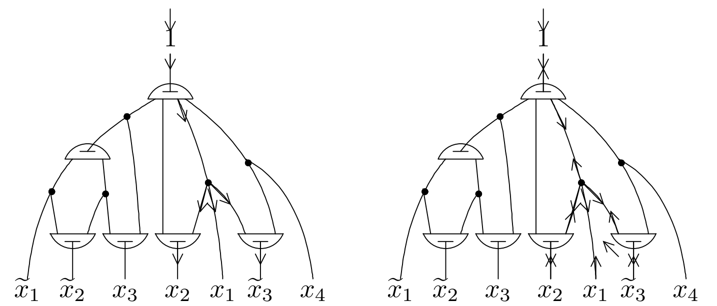
(h) $(y_1.y_1')^\infty$

Figure 67: Some paths found with the general e-scanning algorithm

Note that every "immediately" returning paths are returning paths.

As seen clearly, this is an endless procedure. For this reason, even in an unreduced formula, it produces infinite number of paths and e-products.

In an unreduced formula, the e-products found by the normal algorithm are finite and in fact this algorithm is enough to find all the e-products of the formula. The e-products scanned by the normal algorithm can also be scanned by the general procedure as shown in the figure.



(a) An e-product scan by the normal algorithm

(b) A scanning of the same e-product by the general algorithm

Figure 68: The e-products found by the normal algorithm can also be found by the general algorithm

In fact, the general scanning algorithm can be reduced to normal scanning algorithm by some changes on nodes.

Consider the node $p \bullet (r_1, .., r_n)$. If we write this node as $p \bullet (p', r_1, .., r_n)$ then the processing of an AND-node in general algorithm becomes the processing of an AND-node in the normal algorithm. Let transform all the nodes as follows.

- pr becomes $p \bullet (p', r)$.
- $p \dashv (r_1, .., r_n)$ composed from two nodes. One is $p \bullet (p', a)$, and the another one is $a \dashv (r_1, .., r_n)$.

- $p \bullet (r_1, .., r_n)$ becomes $p \bullet (p', r_1, .., r_n)$.
- $p \vdash r$ composed from two nodes. One is $p \bullet (p', a)$ and the other is $a \vdash r$.
- px becomes $p \bullet (p', x)$.

Thus choosing the reverse arc of an arc becomes the choosing of one of the outgoing arches of the nodes after transform. Moreover, there is no new node type. All the nodes are still OR-nodes or AND-nodes. I.e. the normal algorithm can be used instead of the general algorithm with these changes. The meaning of this is that all the theorems valid for the normal algorithm is also valid for the general algorithm if the theorems or proofs do not require to refer to immediately returning paths.

Skeletons [3]

In any e-products, there are both returning and nonreturning paths. Then we can express the path set of an e-product scan as follows:

$\Rrightarrow\lambda a = \Rrightarrow\lambda\{\underset{\rightarrow}{\underline{A}}, \underset{\leftrightarrow}{\underline{A}}\}$

where $\underset{\rightarrow}{\underline{A}}$ and $\underset{\leftrightarrow}{\underline{A}}$ describes all returning paths and all non returning paths of λa respectively.

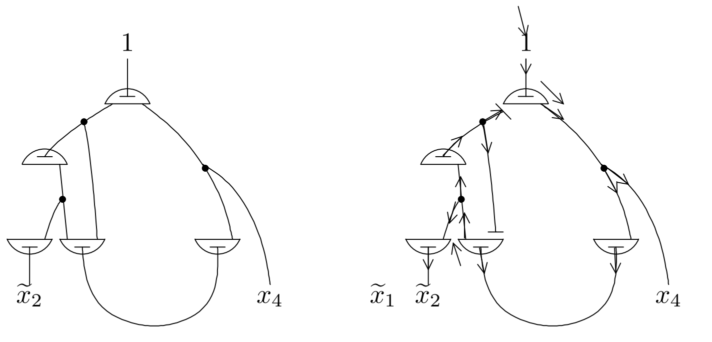
We define a skeleton of an e-product scan as follows.

Definition 11 *Let λa be a skeleton of λb*

1. *For $\underset{\rightarrow}{\underline{A}} \in \lambda b$, $\underset{\rightarrow}{\underline{A}} \in \lambda a$ also*
2. *For $\underset{\leftrightarrow}{\underline{A}} \in \lambda b$, let $\underset{\leftrightarrow}{\underline{A}} = \underline{B}p\underline{C}p'\underline{D}$ where $\underline{B}p\underline{C}$ is not a returning path. Then $\underline{B}p\underline{C}p' \in \lambda b$*

For the paths of the type $\underline{B}p\underline{C}p'$ p' is the returning point of the path and the only reason we put p' here is to not loss the information that this path is produced from a returning one and the first returning point is p'.

For example, in the figure, the skeleton of an e-product scan are shown.



(a) An e-product

(b) A skeleton of the e-product

Figure 69: The skeleton of an e-product scan. The returning points are marked with a disconnected arc terminated with a perpendicular line

We can classify the e-products according to their skeletons. Let $\Rrightarrow\{\lambda a_1, .., \lambda a_m, ..\}$ the set of all e-products and $\Rrightarrow\{\lambda b_1, .., \lambda b_n, ..\}$ be the set of all skeletons of these scans. In this case, for any λa_i there is exactly one skeleton λb_j, however for every skeleton λb_j there are more than one e-product scan.

[3]Not complete yet

Definition 12 *The value of the paths of a skeleton λa*

1. *For $\underset{\rightarrow}{\underline{A}} \in \lambda b$, $\underset{\rightarrow}{\underline{A}} \cong$ terminal value*

2. *For $\underline{B}p\underline{C}p' \in \lambda b$, let $\underline{B}p\underline{C}p' \cong 1$*

I.e. the value of a skeleton is the disjunction of the values of its unidirectional paths.

Definition 13 *Let $X \sqcup Y$ be as follows*

1. *for $\underline{A}r \in X$ and $r\underline{B} \in Y$ $\underline{A}r\underline{B} \in X \sqcup Y$*

2. *for $\underline{A}r \in X$ such that there is no $r\underline{B} \in Y$ exist, $\underline{A}r \in X \sqcup Y$*

Theorem 8 *Let consider the skeleton of an e-product scan. There is at least one e-product scan having this skeleton and the value of it is equal to the value of the skeleton.*

Proof 8 *Let $\Rightarrow p\lambda a\langle T_1\rangle\langle T_2\rangle$ be a scan of the e-product $\lambda a\langle p\rangle\langle T_1\rangle\langle T_2\rangle$ where $\langle T_1\rangle$ are the terminals of unidirectional paths and λT_2 are the terminals of returning paths starting from p. Let $\Rightarrow p\lambda a\langle T_1\rangle\langle T_3\rangle$ be the skeleton of this scan where $\langle T_1\rangle$ be the terminals of unidirectional paths and $\langle T_3\rangle$ be the returning points of the other paths formed from the returning paths of $\Rightarrow p\lambda a$. Note that all returning points are the open terminals of the skeleton. Now we will show that there is an e-product scan $\Rightarrow p\lambda a\langle T_1\rangle$ whose value is equal to the value of the skeleton.*

Let consider the open path $\Rightarrow p\underline{A}r\underline{B}r'$ in the path set. Here $r' \in \langle T_3\rangle$ and let it is not in $\langle T_1\rangle$. Since r' is an open terminal, it is expandable. In this case we can express the e-scanning as $E_0 = \Rightarrow p\lambda(a:\underline{A})r\lambda(b:\underline{B})r'$. Now expand the scan from r' and let it becomes $E_1 = \Rightarrow p\lambda(a:\underline{A})r\lambda(b:\underline{B})r' \sqcup \Rightarrow r'\lambda(a:\underline{A})p'$. If , there are the paths in the form $r'..r$ in $\Rightarrow r'\lambda(a:\underline{A})p'$ then do the expansion as follows: $E_2 = E_1 \sqcup \Rightarrow r\lambda(b:\underline{B})r'$. By this way we can expand all the open terminals r and r'. At last, there will be no open terminals r and r' in the scan. The infinite paths produced are in the form $..r..r'$ and thus the values of them would be 1.

When we apply this process to all open terminals in $\langle T_3\rangle$, we get a general e-product scan and all the open terminals in $\langle T_3\rangle$ are vanished.

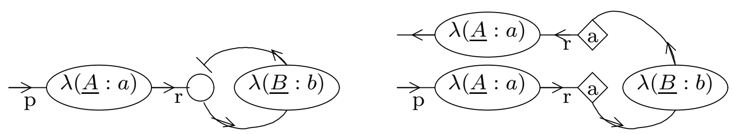


Figure 70:

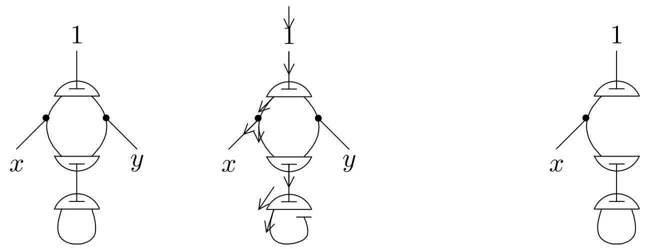
We call such e-products primary e-products.

Theorem 9 *The value of a formula is equal to the value of disjunction of all the primary e-products.*

Proof 9 *Let $\langle T_1\rangle$ be the terminals of unidirectional paths of an e-product. We have proved that there is an e-product whose all paths ends with terminals in T_1 or in the form $\underline{A}(\underset{\leftrightarrow}{B})^\infty$. Since the value of any e-products having $\langle T_1\rangle$ can be written as $\wedge\{T_1\} \cdot X$ and the value of primal e-product is $\wedge\{T_1\}$ the theorem is proved*

Theorem 10 *In a formula, if the value of any returning path is accepted as 1, the value of the formula does not change.*

Proof 10 *A result of the previous theorems.*



(a) A formula

(b) A skeleton

(c) A primary e-product corresponding to the skeleton

Figure 71: A skeleton and its primary e-product are equivalent

5 General Form and Reduction (not complete)

5.1 Abbreviation

Similar to statement formulas, two dimensional formulas can be abbreviated too. We can see the abbreviations as black box. The details of selected part of the formula is hide and only the connections to the rest of the formula are drawn. For example, in figure, some of the abbreviations are shown.

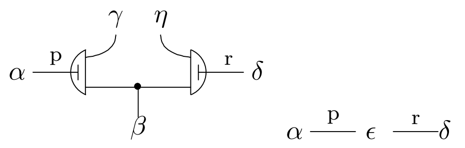
Figure 72: Sample formula and some abbreviations

The black box either has no open terminal or has several open terminals.

In the expression $\overset{p}{-}\alpha$, α is the symbolizing form of the formula whose one of and the only open terminal is p'. In this regard, the following two formula are not equal to each other, so the symbols representing them are also not equal. On the other hand, the root of these formulas are equivalent, therefore the symbols representing them are also equivalent.

However, the abbreviation of formulas having more than one open terminal is not so easy. Let see the example given in the figure.

Can we abbreviate this formula as shown? Should ϵ seen as the same from both p and r' in $\overset{p}{-}\epsilon\overset{r}{-}$ ϵ? Is the value of this formula is the value of p or the value of r'?



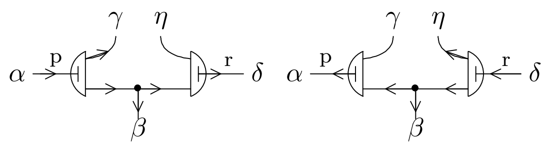
(a) (b)

Figure 73: Can this formula be abbreviated as this?

If ϵ had been a literal, then $|p| = |\epsilon| \cdot |r|$ and $|r'| = |\epsilon| \cdot |p'|$ would have been true. For example, for $\overset{p}{-}x\overset{r}{-}$, the variable x has only one value for each interpretation and $|p| = |x| \cdot |r|$ and $|r'| = |x| \cdot |p'|$ are true. If we use one atomic symbol for a formula, then we should accept that this symbol takes only one value for each interpretation. I.e. an atomic symbol should be treated as if it were a literal. Then we will do the abbreviation as follows.

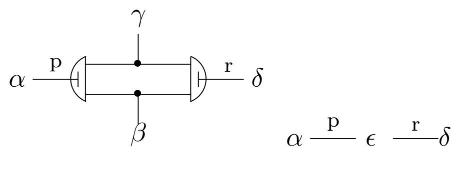
Definition 14 *A formula can be abbreviated as* $\overset{p}{-}\epsilon\overset{r}{-}$ *iff it is true that* $|p| = X \cdot |r|$ *and* $|r'| = X \cdot |p'|$. *Here it is accepted as* $X = |\epsilon|$.

However, the given sample formula does not hold this rule and therefore can not be abbreviated as this.



(a) (b)

Figure 74: This formula does not hold the relation $|p| = X \cdot |r|$ ve $|r'| = X \cdot |p'|$



(a) (b)

Figure 75: This formula validates the relation $|p| = X \cdot |r|$ ve $|r'| = X \cdot |p'|$ and can be abbreviated as shown. Here $X = |\epsilon| = |\gamma| \cdot |\beta|$ is true

On the other hand the following formula can be abbreviated as this because it holds the given criteria.

We want to write this abbreviation in statement form also. Let show $\overset{p}{-}\epsilon\overset{r}{-}$ as $p\lfloor\alpha\rfloor r$. Here $\lfloor\rfloor$ shows that the only open terminals of the formula are p' and r.

Theorem 11 *If a formula can be abbreviated as $p\lfloor\alpha\rfloor r$, then in all the e-product found from p, there are p..r paths and in all the e-products found from r', there are $r'..p'$ paths.*

Proof 11 *If a formula can be abbreviated as $p\lfloor\alpha\rfloor r$ then $|p| = X \cdot |r|$ and $|r'| = X \cdot |p'|$ would be true. Assume that in the e-products having p does not have p..r and/or in the e-products having r' does not have $r'..p'$. Let the e-products having p be $p\{\lambda a_1, .., \lambda a_m\}$ and the e-products having r' be $r'\lambda b_1, .., \lambda b_n$. In this case for some i's $|p\lambda a_i| \neq X \cdot |r|$ would be true. Therefore, $|p| = |\lambda a_i| \cdot |r| \vee .. \vee |\lambda a_{i-1}| \cdot |r| \vee |\lambda a_i| \vee |\lambda a_{i+1}| \cdot |r| \vee |\lambda a_m| \cdot |r|$ and $|p| \neq X \cdot |r|$ would be true.*

Now investigate how to abbreviate the formulas which does not have this property

Our choice is to abbreviate such formulas, i.e. the formulas which is not seen the same from p and r' as follow.

$\overset{p}{—}\alpha$ or $\overset{p}{—}\alpha$	represents the formula scanned from p
$\overset{p}{—}\alpha\overset{r}{—}$ or $\overset{p}{—}\alpha\overset{r}{—}$	represents a formula where all the e-products contain a $p..r$ path
$\overset{p}{—}\alpha:\delta:\beta\overset{r}{—}$ or $\overset{p}{—}\alpha\;\delta\;\beta\overset{r}{—}$	α represents the e-products found from p but not contain $p..r$ paths, β represents the e-products found from r' but not contain $r'..p'$ paths and δ represents the set of e-products which contain $p..r$ paths.

By this way, not only the formulas with two open terminals, but the formulas having more than two terminals can also be abbreviated. To do this, for every sub formula containing one open terminal, an eclipse is drawn as connected to that terminal. The intersection areas of the eclipses represents the formulas which have the open terminals where the eclipses intersected are connected.The following figure show this.

5.2 Simple Forms and Reduction

After all of these, now we are ready to investigate which form the invalid e-products take.

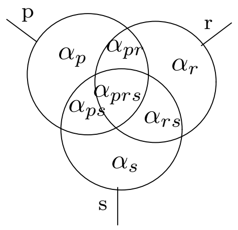


Figure 76: Abbreviation of a formula with three open terminals

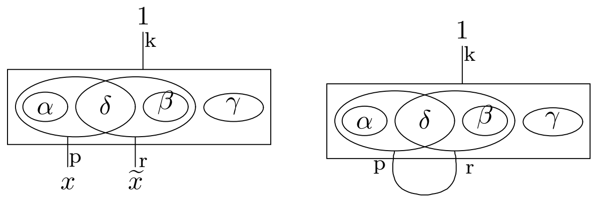


Figure 77: An abbreviation of a regular formula

Here α is the set of e-products having $k..p$ paths, β is the set of e-products having $k..r$ paths, δ is the set of e-products having both $k..p$ and $k..r$ paths and γ is the set of e-products having neither of these kind of paths.

In an unreduced formula, there is no unidirectional infinite path.

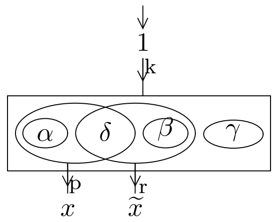


Figure 78: In an unreduced formula, there is no unidirectional infinite path

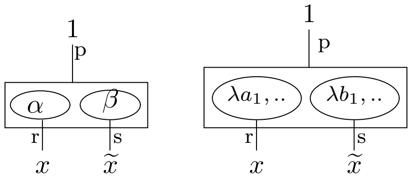
We can write this formula as $xp'\lfloor\alpha:\delta:\beta\rfloor r\widetilde{x}$ in statement form. The scans done in the formula are shown below.

$$\begin{aligned}
\Rightarrow 1k\{\} \quad \cong \quad & \{\Rightarrow 1k\lfloor\alpha\rfloor px \vee \\
& \Rightarrow 1k\lfloor\delta\rfloor < \begin{matrix} px \\ r\widetilde{x} \end{matrix} \vee \\
& \Rightarrow 1k\lfloor\beta\rfloor r\widetilde{x} \vee \\
& \Rightarrow 1k\lfloor\gamma\rfloor\}
\end{aligned}$$

$\Rightarrow k\lfloor\alpha\rfloor p$ can be written as $\Rightarrow\{kA_1, kA_2k', kA_3p\}$ where kA_3p is the set of all paths in the form $k..p$, kA_2k' is the set of all paths in the form $k..k'$ and kA_1 is the set of all the other paths. Note that since there is no unidirectional infinite paths, the paths in $\Rightarrow kA_1$ are all either returning or finite.

5.2.1 The simple form with only α and β

where $p\lfloor\alpha\rfloor r$ is the set of all e-products whose at least one path is in the form $p..r$ and the only open terminals are p' and r.

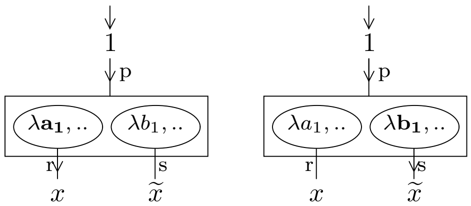


(a) (b)

Figure 79: The simple form

The value of the formula is found by scanning it. The scan is written as

$$\begin{aligned} & \{\Rightarrow 1p\lfloor\alpha\rfloor rx, \Rightarrow 1p\lfloor\beta\rfloor s\widetilde{x}\} \\ = \quad & \Rightarrow 1p\{\lfloor\alpha\rfloor rx, \lfloor\beta\rfloor s\widetilde{x}\} \end{aligned}$$



(a) $\Rightarrow 1p\lfloor\lambda a_1\rfloor rx$ (b) $\Rightarrow 1p\lfloor\lambda b_1\rfloor s\widetilde{x}$

Figure 80: two e-product scans

Let $p\lfloor\alpha\rfloor r = p\{\lfloor\lambda a_1\rfloor, .., \lfloor\lambda a_i\rfloor, ..\}r$. Then a scan $\Rightarrow p\lfloor\lambda a_i\rfloor r$ can be written as $\Rightarrow\lambda\{pA_{i1}, pA_{i2}p', pA_{i3}r\}$ where $\Rightarrow pA_{i3}$ is the set of all paths from p to r, $\Rightarrow pA_{i2}p'$ is the set of all paths from p to p' and $\Rightarrow pA_{i1}$ is the set of all the other paths. Note that no path in $\Rightarrow p\lfloor\lambda a_i\rfloor r$ is in the form $\Rightarrow p..r..$ or $\Rightarrow p..p'..$. I.e. the open terminals r and p' can only be the last element of the paths. Since the value of an e-product is the conjunction of the value of its paths and the value of path is equivalent to its terminal, it can be written

$$\begin{aligned} \Rightarrow p\lfloor\lambda a_i\rfloor r &= \Rightarrow\lambda\{pA_{i1}, pA_{i2}p', pA_{i3}r\} \\ &\cong \Rightarrow\{\text{⅄}\{pA_{i1}\} \wedge \text{⅄}\{pA_{i2}p'\} \wedge \text{⅄}\{pA_{i3}r\}\} \\ &\cong \Rightarrow\{\text{⅄}\{pA_{i1}\} \wedge p' \wedge r\} \\ &\cong \|\Rightarrow\text{⅄}\{pA_{i1}\}\| \cdot \|\Rightarrow p'\| \cdot \|\Rightarrow r\| \end{aligned}$$

Now can we express the paths of the scan $\Rightarrow 1p\lfloor\lambda a_1\rfloor rx$ in terms of the paths of the scan $\Rightarrow p\lfloor\lambda a_1\rfloor r$. We can express the scan as joining of $\Rightarrow 1k$,$\Rightarrow p\lfloor\lambda a_1\rfloor r$ and $\Rightarrow rx$ as follows

$\Rightarrow 1p - \Rightarrow p\lfloor\lambda a_1\rfloor r - \Rightarrow rx$

or

$\Rightarrow 1p \bowtie \Rightarrow p\lfloor\lambda a_1\rfloor r \bowtie \Rightarrow rx$

Joining e-products The scan $\rightarrowtriangle 1p$ is an e-product with open terminal p. So $\rightarrowtriangle 1p = \rightarrowtriangle \lambda\{1p\}$. The scan $\rightarrowtriangle rx$ is an e-product with open terminal r'. So $\rightarrowtriangle rx = \rightarrowtriangle \lambda\{rx, rr'\}$. Now if we join $\rightarrowtriangle 1p$ and $\rightarrowtriangle p\lfloor \lambda a_1 \rfloor r$ we get an e-product scan whose the only open terminal is r.

Theorem 12 $\rightarrowtriangle p\lfloor \lambda a\rfloor r \bowtie \rightarrowtriangle r\lfloor \lambda b\rfloor = \rightarrowtriangle p\lfloor \lambda a\rfloor r(\sqcup \rightarrowtriangle r\lfloor \lambda b\rfloor \sqcup \rightarrowtriangle r'\lfloor \lambda a\rfloor p')^{\infty}$

Proof 12

Theorem 13 $\rightarrowtriangle p\lfloor \lambda a\rfloor r \bowtie \rightarrowtriangle r\lfloor \lambda b\rfloor \cong \rightarrowtriangle \{1p\lfloor \lambda a\rfloor r1 \wedge 1r\lfloor \lambda b\rfloor \wedge p'\}$ *is true.*

Proof 13 *We know that*

$$\rightarrowtriangle p\lfloor \lambda a\rfloor r \bowtie \rightarrowtriangle r\lfloor \lambda b\rfloor \cong \rightarrowtriangle p\lfloor \lambda a\rfloor r(\sqcup \rightarrowtriangle r\lfloor \lambda b\rfloor \sqcup \rightarrowtriangle r'\lfloor \lambda a\rfloor p')^{\infty}$$

is true. Let $\rightarrowtriangle p\lfloor \lambda a\rfloor r = \rightarrowtriangle \lambda\{pA_1, pA_2p', pA_3r\}$, $\rightarrowtriangle r\lfloor \lambda b\rfloor = \rightarrowtriangle \lambda\{rB_1, rB_2r'\}$ *and* $\rightarrowtriangle r'\lfloor \lambda a\rfloor p' = \rightarrowtriangle \lambda\{r'A_4, r'A_3'p', r'A_5r\}$ *be true. Here* $r'A_4 \cong pA_1$ *is true. Then*

$$\begin{aligned}
\rightarrowtriangle p\lfloor \lambda a\rfloor r \bowtie \rightarrowtriangle r\lfloor \lambda b\rfloor &\cong \rightarrowtriangle p\lfloor \lambda a\rfloor r(\sqcup \rightarrowtriangle r\lfloor \lambda b\rfloor \sqcup \rightarrowtriangle r'\lfloor \lambda a\rfloor p')^{\infty}\\
&\cong \rightarrowtriangle \lambda\{pA_1, pA_2p', pA_3r\}(\sqcup \rightarrowtriangle \lambda\{rB_1, rB_2r'\} \sqcup \rightarrowtriangle \lambda\{r'A_4, r'A_3'p', r'A_5r\})^{\infty}\\
&\cong \rightarrowtriangle \lambda\{pA_1, pA_2p', pA_3r\}(\sqcup \rightarrowtriangle \lambda\{rB_1, rB_2r'A_4, rB_2r'A_3'p', rB_2r'A_5r\})^{\infty}
\end{aligned}$$

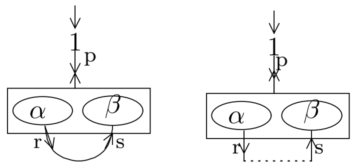


Figure 81: A scan can be seen as a join of two scans

Reduction We can write these scans as

$\rightarrowtriangle 1k\lfloor \alpha\rfloor p.r'..$
$\rightarrowtriangle 1k\lfloor \beta\rfloor r.p'..$
$\rightarrowtriangle 1k\lfloor \gamma\rfloor$

In fact these scans can be seen as joining of two scans. For example $\rightarrowtriangle 1k\lfloor \alpha\rfloor p.r'..$ is a join of the scans $\rightarrowtriangle 1k\lfloor \alpha\rfloor p$ and $\rightarrowtriangle r'\lfloor \beta\rfloor k'1$. We will show this as

$$\rightarrowtriangle 1k\lfloor \alpha\rfloor p \bowtie r'\lfloor \beta\rfloor k'1$$

. So we can write these scans as

$\rightarrowtriangle 1k\lfloor \alpha\rfloor p \bowtie r'\lfloor \beta\rfloor k'1$
$\rightarrowtriangle 1k\lfloor \beta\rfloor r \bowtie p'\lfloor \alpha\rfloor k'1$
$\rightarrowtriangle 1k\lfloor \gamma\rfloor$

Theorem 14 $\rightarrowtriangle p\lfloor \alpha\rfloor r \bowtie \rightarrowtriangle s\lfloor \beta\rfloor = \rightarrowtriangle p\lfloor \alpha\rfloor r(\sqcup \rightarrowtriangle s\lfloor \beta\rfloor \sqcup \rightarrowtriangle s'\lfloor \alpha\rfloor p')^{\infty}$

Proof 14 *We know that since r is joined to s, the only open terminal in $\Rightarrow p\lfloor\alpha\rfloor r \bowtie \Rightarrow s\lfloor\beta\rfloor$ is p' and this formula is the formula of p. If we can show that the only open terminal in $\Rightarrow p\lfloor\alpha\rfloor r(\sqcup \Rightarrow s\lfloor\beta\rfloor \sqcup \Rightarrow s'\lfloor\alpha\rfloor p')^\infty$ is p', and this is the formula of p then we can prove it.*

The scan $\Rightarrow p\lfloor\alpha\rfloor r$ can be expressed as $\Rightarrow\{pA_1, pA_2p', pA_3r\}$ where pA_3r, pA_2p' and pA_1 are the set of paths from p to r, from p to p' and from p to other terminals respectively. Similarly let $\Rightarrow s\lfloor\beta\rfloor = \Rightarrow\{sB_1, sB_2s'\}$. Then

$$\begin{aligned}\Rightarrow p\lfloor\alpha\rfloor r \sqcup \Rightarrow s\lfloor\beta\rfloor &= \Rightarrow\{pA_1, pA_2p', pA_3r\}\text{ş} \sqcup \text{ş}\Rightarrow\{sB_1, sB_2s'\}\\ &= \Rightarrow\{pA_1, pA_2p', pA_3r\text{ş}sB_1, pA_3r\text{ş}sB_2s'\}\\ &= E_1\end{aligned}$$

As seen in E_1 the open terminals are p' and s'. Now the scan starting from s' can be $\Rightarrow r'\lfloor\alpha\rfloor p'$ since s' is joined to r'. Let $\Rightarrow r'\lfloor\alpha\rfloor p' = \Rightarrow\{r'A_4, r'A_5r, r'A_3'p'\}$ where pA_1 and $r'A_4$ has the same terminals. So in

$$E_2 = E_1 \bowtie \Rightarrow\{r'A_4, r'A_5r, r'A_3'p'\}$$

, the open terminal s' will be vanished. However, in E_2, the open terminals would be p' and r. By this way we close the open terminals by expanding them and at last we get a formula with only open terminal p'

Theorem 15 *$\Rightarrow p\lfloor\alpha\rfloor r \rtimes \Rightarrow r\lfloor\beta\rfloor \cong \Rightarrow p\lfloor\alpha\rfloor r \sqcup \Rightarrow r\lfloor\beta\rfloor \sqcup \Rightarrow r'X$ is true.*

Proof 15

Theorem 16 *$\Rightarrow p\lfloor\alpha\rfloor r \rtimes \Rightarrow r\lfloor\beta\rfloor \cong \|\Rightarrow 1p\lfloor\alpha\rfloor r1\| \cdot \|\Rightarrow 1r\lfloor\beta\rfloor\| \cdot \|p'\|$ be true.*

Proof 16 *Let $\Rightarrow p\lfloor\alpha\rfloor r = \Rightarrow p\lfloor\lambda a_1, .., \lambda a_m\rfloor r$ and $\Rightarrow r\lfloor\beta\rfloor = \Rightarrow r\lfloor\lambda b_1, .., \lambda b_n\rfloor$ be true. In all the e-products formed here both λa_i and λb_j exist. For this reason, we can write the e-products which both includes λa_i nad λb_j as $\|\lambda a_i\| \cdot \|\lambda b_j\| \cdot X$. One of the e-products includes both λa_i and λb_j is $\Rightarrow p\lambda a_i r\lambda b_j$. However, the value of it would be $\|\lambda a_i\| \cdot \|\lambda b_j\|$. For this reason the value of the formula would be $\bigvee_{i=0}^{m}\bigvee_{j=0}^{n} \|\lambda a_i\| \cdot \|\lambda b_j\|$. On the other hand the value of the formula $\|\Rightarrow p\lfloor\alpha\rfloor r1\| \cdot \|\Rightarrow 1r\lfloor\beta\rfloor\|$ is also equal to this.*

Theorem 17 *$X \bowtie \{Y \vee Z\} \cong \{X \bowtie Y \vee X \bowtie Z\}$ is true.*

Proof 17

5.2.2 The full formula

The reduction is achieved by joining two arches p and r.

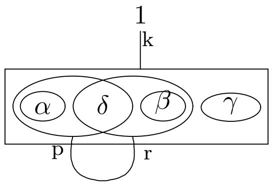


Figure 82: Reduction is equivalent to joining two arches

Definition 15 *The expression* $p\lfloor\alpha\rfloor r \overset{ş}{\bowtie} s\lfloor\beta\rfloor$ *shows that the arches* r *and* s *are joined via ş.*

As usual the scanning of $p\lfloor\alpha\rfloor r \bowtie s\lfloor\beta\rfloor$ is shown as $\Rrightarrow p\lfloor\alpha\rfloor r \bowtie s\lfloor\beta\rfloor$ and we accept the following equality

$$\Rrightarrow p\lfloor\alpha\rfloor r \bowtie s\lfloor\beta\rfloor = \Rrightarrow p\lfloor\alpha\rfloor r \bowtie \Rrightarrow s\lfloor\beta\rfloor$$

where $\Rrightarrow p\lfloor\alpha\rfloor r \bowtie \Rrightarrow s\lfloor\beta\rfloor$ denotes that the scans $\Rrightarrow p\lfloor\alpha\rfloor r$ and $\Rrightarrow s\lfloor\beta\rfloor$ are joined via the arches r and s.

Reduced Simple Form Write the formula including p as $1k\lfloor\alpha, \lfloor\delta\rfloor r\widetilde{x}\rfloor px$ and write the formula including r as $1k\lfloor\beta, \lfloor\delta\rfloor px\rfloor r\widetilde{x}$. By reduction instead of x r' is placed and instead of $\widetilde{x}$, p' is placed.

Now let find the e-products of it by the third algorithm.

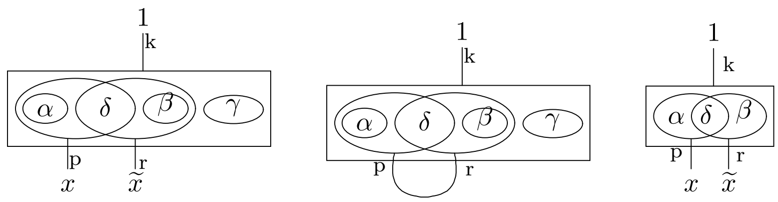
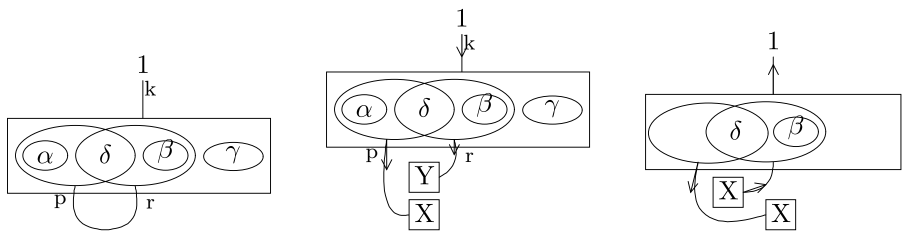
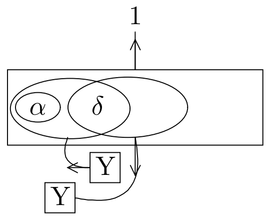


Figure 83: A formula and reduced form

In the figure some possible scans are shown.



(a) (b) $\overset{t\lambda}{\Rrightarrow} 1k\lfloor\alpha\rfloor pX$ (c) $\overset{t\lambda}{\Rrightarrow} X = \overset{t\lambda}{\Rrightarrow} r'\{\lfloor\beta\rfloor, \lfloor\delta\rfloor pX\}$



(d) $\overset{t\lambda}{\Rrightarrow} Y = \overset{t\lambda}{\Rrightarrow} p'\{\lfloor\alpha\rfloor, \lfloor\delta\rfloor rY\}$

Figure 84: Some scans

In this case the following scan groups are found.

$\Rrightarrow 1k\lfloor\alpha\rfloor p \rtimes \Rrightarrow X$
$\Rrightarrow 1k\lfloor\beta\rfloor r \rtimes \Rrightarrow Y$
$\Rrightarrow 1k\lfloor\lfloor\delta\rfloor p \rtimes \Rrightarrow X\rfloor r \rtimes \Rrightarrow Y$
$\Rrightarrow 1k\gamma$

The expansions of X and Y are as follows.

$$\begin{array}{rcl} \Rightarrow X & = & \Rightarrow r' \vee \{\lfloor \delta k'1 \rfloor p \rtimes \Rightarrow X, \lfloor \beta \rfloor k'1\} \\ & = & \Rightarrow \vee \{r'\lfloor \delta \rfloor p \rtimes \Rightarrow X, r'\lfloor \beta \rfloor\} \\ \Rightarrow Y & = & \bowtie p'\lfloor \lfloor \alpha \rfloor k'1 \vee \lfloor \lfloor \delta \rfloor k'1 \rfloor r \rtimes \Rightarrow Y \rfloor \end{array}$$

When we remove the direction from $\Rightarrow X$ and $\Rightarrow Y$ we get X and Y.
From here we find

$$\begin{array}{rcl} \Rightarrow X & = & \Rightarrow \vee \{r'\lfloor \delta \rfloor p \rtimes X, r'\lfloor \beta \rfloor\} \\ & = & \Rightarrow \vee \{r'\lfloor \delta \rfloor p \rtimes (\vee\{r'\lfloor \delta \rfloor p \rtimes X, r'\lfloor \beta \rfloor\}), r'\lfloor \beta \rfloor\} \\ & \cong & \Rightarrow \vee \{r'\lfloor \delta \rfloor p \rtimes r'\lfloor \delta \rfloor p \rtimes X, r'\lfloor \delta \rfloor p \rtimes r'\lfloor \beta \rfloor, r'\lfloor \beta \rfloor\} \\ & \cong & . \\ & \cong & . \\ & \cong & \Rightarrow \lfloor \bigvee_{i=0}^{\infty} (r'\lfloor \delta \rfloor p \rtimes)^i r'\lfloor \beta \rfloor \rfloor \end{array}$$

Similarly

$$\begin{array}{rcl} \Rightarrow Y & = & \Rightarrow p'\lfloor \alpha \vee \delta r \bowtie Y \rfloor \\ & = & \Rightarrow p'\lfloor \alpha \vee \delta r \bowtie p'\lfloor \alpha \vee \delta r \bowtie Y \rfloor \rfloor \\ & \cong & \Rightarrow p'\lfloor \alpha \vee \delta r \bowtie p'\alpha \vee \delta r \bowtie p'\delta r \bowtie Y \rfloor \\ & \cong & . \\ & \cong & . \\ & \cong & \lfloor \bigvee_{i=0}^{\infty} (p'\delta r \bowtie)^i p'\alpha \rfloor \end{array}$$

is found. Then the scans becomes as

$$
\begin{aligned}
\gg 1k\lfloor\alpha\rfloor p \bowtie X &\cong \gg 1k\lfloor\alpha\rfloor p \bowtie \lfloor \overset{\infty}{\underset{i=0}{\vee}} (r'\delta p \bowtie)^i r'\beta\rfloor \\
&\cong \gg \lfloor \overset{\infty}{\underset{i=0}{\vee}} 1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^i r'\beta\rfloor \\
\overset{t\lambda}{\gg} 1k\lfloor\beta\rfloor r - p'Y &= \overset{t\lambda}{\gg} 1k\lfloor\beta\rfloor r - \{\overset{\infty}{\underset{i=0}{\vee}} (p'\lfloor\delta\rfloor r-)^i p'\lfloor\alpha\rfloor\} \\
&= \overset{t\lambda}{\gg} 1k\{\overset{\infty}{\underset{i=0}{\vee}} \lfloor\beta\rfloor r - (p'\lfloor\delta\rfloor r-)^i p'\lfloor\alpha\rfloor\} \\
&= \overset{t\lambda}{\gg} \{\overset{\infty}{\underset{i=0}{\vee}} 1B_{kr} - (D_{p'r}-)^i A_{p'k}\} \\
\overset{t\lambda}{\gg} 1k\{\lfloor\delta\rfloor p.r'X\}r - p'Y &= \overset{t\lambda}{\gg} 1k\{\lfloor\delta\rfloor p - \{\overset{\infty}{\underset{i=0}{\vee}} (r'\lfloor\delta\rfloor p-)^i r'\lfloor\beta\rfloor\}\}r - \{\overset{\infty}{\underset{i=0}{\vee}} (p'\lfloor\delta\rfloor r-)^i p'\lfloor\alpha\rfloor\} \\
&= \overset{t\lambda}{\gg} 1k\lfloor\delta\rfloor < \begin{array}{l} p - \{\overset{\infty}{\underset{i=0}{\vee}} (\lfloor\delta k'1\rfloor_{r'p}-)^i \lfloor\beta k'1\rfloor_{r'}\} \\ r - \{\overset{\infty}{\underset{i=0}{\vee}} (p'\lfloor\delta\rfloor r-)^i p'\lfloor\alpha\rfloor\} \end{array} \\
&= \overset{t\lambda}{\gg} 1 < \begin{array}{l} D_{kp} - \{\overset{\infty}{\underset{i=0}{\vee}} (D_{r'p}-)^i B_{r'k}\} \\ D_{kr} - \{\overset{\infty}{\underset{i=0}{\vee}} (D_{p'r}-)^i A_{p'k}\} \end{array} \\
\overset{t\lambda}{\gg} 1k\lfloor\gamma\rfloor &
\end{aligned}
$$

At least, the disjunction of the values of the scans must be equal to $\alpha \cdot \beta \vee \gamma$. However, as seen there are some scans including the term δ and such scans must not affect the result.

$$
\begin{aligned}
\gg 1k\lfloor\alpha\rfloor p \bowtie X &\cong \gg \lfloor \overset{\infty}{\underset{i=0}{\vee}} 1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^i r'\beta\rfloor \\
&= \gg \{\underset{i=0}{\ddot{\vee}} (1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^i r'\beta) \vee (1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^\infty)\} \\
&\cong \gg \lfloor \underset{i=0}{\ddot{\vee}} ((1k\lfloor\alpha\rfloor p1) \wedge (1r'\delta p1\wedge)^i (1r'\beta)) \vee (1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^\infty)\rfloor \\
&= \gg \lfloor \underset{i=0}{\ddot{\vee}} A \wedge (D\wedge)^i B \vee 1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^\infty\rfloor \\
&\cong \gg \lfloor A \wedge B \vee 1k\lfloor\alpha\rfloor p \bowtie (r'\delta p \bowtie)^\infty\rfloor
\end{aligned}
$$

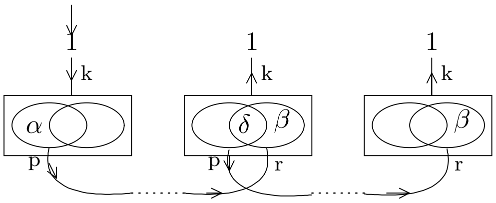
Now let separate the scans including δ and the others and investigate whether they have some differences.

Then the scans that have both α and β are the followings

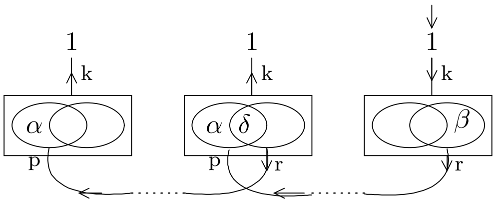
$$
\begin{aligned}
&\overset{t\lambda}{\gg} 1k\{\underset{i=0}{\ddot{\cup}} \lfloor\alpha\rfloor p - (r'\lfloor\delta\rfloor p-)^i r'\lfloor\beta\rfloor\} \\
&\overset{t\lambda}{\gg} 1k\{\underset{i=0}{\ddot{\cup}} \lfloor\beta\rfloor r - (p'\lfloor\delta\rfloor r)^i p'\lfloor\alpha\rfloor\} \\
&\overset{t\lambda}{\gg} 1k\{\lfloor\delta\rfloor p - \{\underset{i=0}{\ddot{\cup}} (r'\lfloor\delta\rfloor p-)^i r'\lfloor\beta\rfloor\}\}r - \{\underset{i=0}{\ddot{\cup}} (p'\lfloor\delta\rfloor r-)^i p'\lfloor\alpha\rfloor\}
\end{aligned}
$$

In fact these are the scans of the same sub formula. .e. we can write all of these scans as

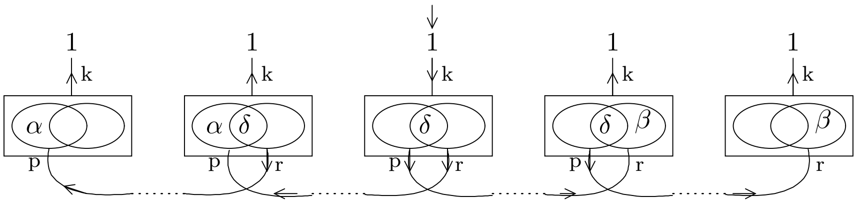
$$
\{\underset{i=0}{\ddot{\cup}} \lfloor\alpha\rfloor p - (r'\lfloor\delta\rfloor p-)^i r'\lfloor\beta\rfloor\}
$$



(a) $\overset{t\lambda}{\Rightarrow}1k\{\ddot{\bigcup}_{i=0}\lfloor\alpha\rfloor p-(r'\lfloor\delta\rfloor p-)^{i}r'\lfloor\beta\rfloor\}$



(b) $\overset{t\lambda}{\Rightarrow}1k\{\ddot{\bigcup}_{i=0}\lfloor\beta\rfloor r-(p'\lfloor\delta\rfloor r)^{i}p'\lfloor\alpha\rfloor\}$



(c) $\overset{t\lambda}{\Rightarrow}1k\{\lfloor\delta\rfloor p-\{\ddot{\bigcup}_{i=0}(r'\lfloor\delta\rfloor p-)^{i}r'\lfloor\beta\rfloor\}\}r-\{\ddot{\bigcup}_{i=0}(p'\lfloor\delta\rfloor r-)^{i}p'\lfloor\alpha\rfloor\}$

Figure 85: The scans including both α and β

.

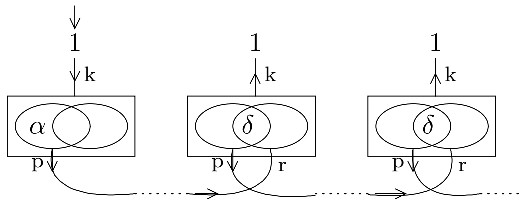
The other scans are the followings

$$\overset{t\lambda}{\Rightarrow}1k\lfloor\alpha\rfloor p\bowtie(r'\lfloor\delta\rfloor p\bowtie)^{\infty}$$
$$\overset{t\lambda}{\Rightarrow}1k\lfloor\beta\rfloor r\bowtie(p'\lfloor\delta\rfloor r)^{\infty}$$
$$\overset{t\lambda}{\Rightarrow}1k\{\lfloor\delta\rfloor p\bowtie(r'\lfloor\delta\rfloor p\bowtie)^{\infty}\}r\bowtie\{\bigcup_{i=0}^{\infty}(p'\lfloor\delta\rfloor r\bowtie)^{i}p'\lfloor\alpha\rfloor\}$$
$$\overset{t\lambda}{\Rightarrow}1k\{\lfloor\delta\rfloor p\bowtie\{\bigcup_{i=0}^{\infty}(r'\lfloor\delta\rfloor p\bowtie)^{i}r'\lfloor\beta\rfloor\}\}r\bowtie(p'\lfloor\delta\rfloor r.)^{\infty}$$
$$\overset{t\lambda}{\Rightarrow}1k\{\lfloor\delta\rfloor p\bowtie(r'\lfloor\delta\rfloor p\bowtie)^{\infty}\}r\bowtie(p'\lfloor\delta\rfloor r\bowtie)^{\infty}$$

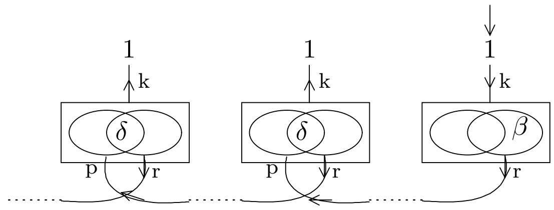
As seen in all of these scans δ^{∞} exists.

To be the result be true, the value of the all scans that include both α and β should be $\alpha\cdot\beta$ and the value of all the other scans must not affect the result.

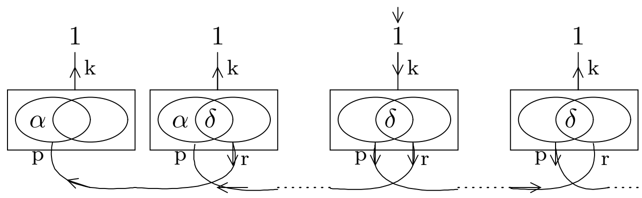
The terms which include both α and β We have shown that $\Rightarrow p\lfloor\alpha\rfloor r\lfloor\beta\rfloor\cong\|\overset{t\lambda}{\Rightarrow}p\lfloor\alpha\rfloor r1\|\cdot\|\overset{t\lambda}{\Rightarrow}r\lfloor\alpha\rfloor\|$ is true. From here



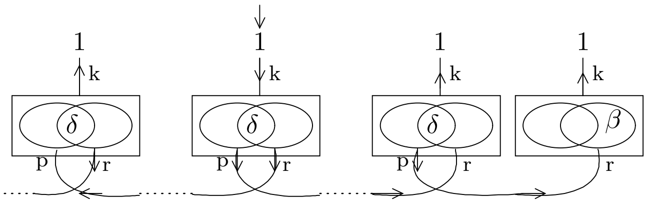
(a) $\overset{t\lambda}{\geqslant} 1k\lfloor\alpha\rfloor p \bowtie (r'\lfloor\delta\rfloor p \bowtie)^{\infty}$



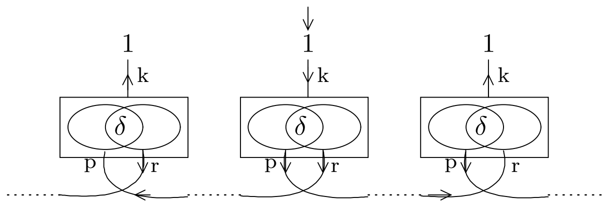
(b) $\overset{t\lambda}{\geqslant} 1k\lfloor\beta\rfloor r \bowtie (p'\lfloor\delta\rfloor r)^{\infty}$



(c) $\overset{t\lambda}{\geqslant} 1k\{\lfloor\delta\rfloor p \bowtie (r'\lfloor\delta\rfloor p \bowtie)^{\infty}\}r \bowtie \{\bigcup_{i=0}^{\infty}(p'\lfloor\delta\rfloor r \bowtie)^{i}p'\lfloor\alpha\rfloor\}$



(d) $\overset{t\lambda}{\geqslant} 1k\{\lfloor\delta\rfloor p \bowtie \{\bigcup_{i=0}^{\infty}(r'\lfloor\delta\rfloor p \bowtie)^{i}r'\lfloor\beta\rfloor\}\}r \bowtie (p'\lfloor\delta\rfloor r.)^{\infty}$



(e) $\overset{t\lambda}{\geqslant} 1k\{\lfloor\delta\rfloor p \bowtie (r'\lfloor\delta\rfloor p \bowtie)^{\infty}\}r \bowtie (p'\lfloor\delta\rfloor r \bowtie)^{\infty}$

Figure 86: The scans that does not include both α and β

$$
\begin{aligned}
& 1k\{\ddot{\bigcup_{i=0}}\lfloor\alpha\rfloor p - (r'\lfloor\delta\rfloor p-)^{i}r'\lfloor\beta\rfloor\} \\
\cong\ & 1k\lfloor\alpha\rfloor p - \{\ddot{\bigcup_{i=0}}(r'\lfloor\delta\rfloor p-)^{i}\}r'\lfloor\beta\rfloor \\
\cong\ & \|\overset{t\lambda}{\geqslant}1k\lfloor\alpha\rfloor p\| \cdot \|\overset{t\lambda}{\geqslant}\{\ddot{\bigcup_{i=0}}(r'\lfloor\delta\rfloor p-)^{i}\}r'\lfloor\beta\rfloor\| \\
\cong\ & . \\
\cong\ & . \\
\cong\ & \|\overset{t\lambda}{\geqslant}1k\lfloor\alpha\rfloor p1\| \cdot (\ddot{\bigvee_{i=0}}\|\overset{t\lambda}{\geqslant}r'\lfloor\delta\rfloor p1\|^{i}) \cdot \|\overset{t\lambda}{\geqslant}r'\lfloor\beta\rfloor\| \\
\cong\ & X \cdot (\ddot{\bigvee_{i=0}}Y^{i}) \cdot Z \\
\cong\ & X \cdot Z \vee X \cdot (\ddot{\bigvee_{i=1}}Y^{i}) \cdot Z \\
\cong\ & X \cdot Z \vee X \cdot Y \cdot Z \\
\cong\ & X \cdot Z \\
\cong\ & \|\overset{t\lambda}{\geqslant}1k\lfloor\alpha\rfloor p1\| \cdot \|\overset{t\lambda}{\geqslant}r'\lfloor\beta\rfloor\|
\end{aligned}
$$

Other Terms If we can accept the terms including $\geqslant(r'\delta p-)^{\infty}$ as 0, then the result would be true.

Theorem 18 *If* $\overset{t\lambda}{\geqslant} p\Lambda(\underline{A})r\Lambda(\underline{B})r'$ *exists, then* $\overset{t\lambda}{\geqslant} p\Lambda(\underline{A})r\Lambda(\underline{B}')r'$ *also exists.*

Proof 18

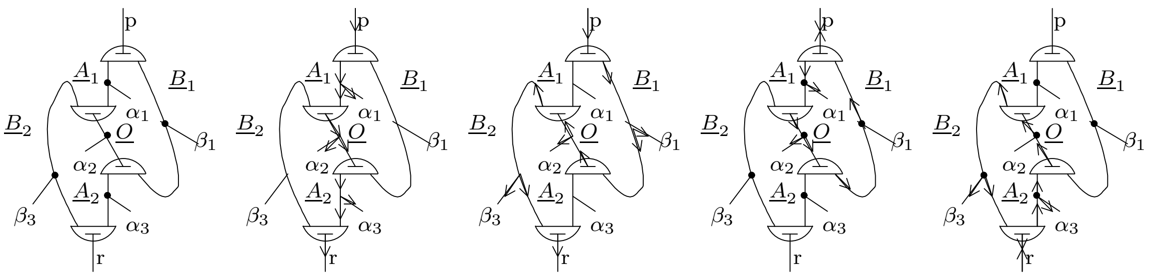
Theorem 19 *If there are two e-products in* $\overset{t\lambda}{\geqslant} p\lfloor\delta\rfloor r$ *has opposing arches then there is another e-product which covers it and does not include* r.

Proof 19 *Let the two e-products be* $p\lfloor\lambda(\underline{A}_1 : a)\rfloor r$ *and* $p\lfloor\lambda(\underline{B}_1 : b)\rfloor r$. *Now let* A_1 *and* B_1 *has opposite arches. I.e.* $p\underline{A}_1 r = p\underline{A}_{11}.\underline{G}.\underline{A}_{12}r$ *and* $p\underline{B}_1 r = p\underline{B}_{11}.\underline{G}'.\underline{B}_{22}r$ *be true. Then* $p\lfloor\lambda(\underline{A}_1 : a)\rfloor r$ *and* $p\lfloor\lambda(\underline{B}_1 : b)\rfloor r$ *can be expressed as* $p\lfloor\lambda(\underline{A}_{11} : a_1).\lambda(\underline{G} : g).\lambda(\underline{A}_{12} : a_2)\rfloor r$ *and* $p\lfloor\lambda(\underline{B}_{11} : b_1).\lambda(\underline{G}' : g).\lambda(\underline{B}_{22} : b_2)\rfloor r$ *respectively. However there will be a path* $p\underline{A}_{11}.\underline{G}.\underline{B}'_{11}p'$ *and therefore an e-product* $p\lfloor\lambda c\rfloor r = p\lfloor\lambda(\underline{A}_{11} : a_1).\lambda(\underline{G} : g).\lambda(\underline{B}'_{11} : b_1)\rfloor p'$. *Thus we get an e-product which does not have opposing paths and all the other paths composed from the two e-products. Now assume* $p\lfloor\lambda c\rfloor r$ *has a path* pC_1r *and this is either from* $p\lambda a$ *or* λb.

Theorem 20 *Any e-products in* $\overset{t\lambda}{\geqslant}(p\delta r-)^\infty$ *either include infinite unidirectional paths or there is another e-product whose value covers the value of it and whose one of the open terminals is not* r.

Proof 20 *Let* $p\delta r = p\{\Lambda(\underline{A}_1), .., \Lambda(\underline{A}_n)\}$ *be true. Now let* A_i *and* A_j *has opposite arches. I.e.* $p\underline{A}_i r = p\underline{A}_{i1}.\underline{G}.\underline{A}_{i2}r$ *and* $p\underline{A}_j r = p\underline{A}_{j1}.\underline{G}'.\underline{A}_{j2}r$ *be true. In this case let take any e-product* $\geqslant X - p\Lambda(\underline{A}_i)r - Y - p\Lambda(\underline{A}_j)r - ...$ *Then there will be an e-product such as* $\overset{t\lambda}{\geqslant} X - p\Lambda(\underline{A}_{i1}s\underline{A}_{i2})r - Y - p\Lambda(\underline{A}_{j1})s'$. *However, this implies that there is an e-product* $\overset{t\lambda}{\geqslant} pX - p\Lambda(\underline{A}_{i1})s\Lambda(\underline{A}'_{j1})p'$. *From here, it is understood that there is an e-product* $\overset{t\lambda}{\geqslant} p\Lambda(\underline{A}_{i1})s\Lambda(\underline{A}'_{j1})p'$ *and there is no* r *in this e-product. I.e. there is an e-product which does not include* δ *and covers this e-product.*

This logic is also valid for any e-products which have opposite arches. On the other hand, in the other e-products, there is an infinite one-way path.



(a) (b) $p.\Lambda(\underline{A}_1.\underline{O}.\underline{A}_2).r$ (c) $p.\Lambda(\underline{B}_1.\underline{O}'.\underline{B}_2).r$ (d) $p.\Lambda(\underline{A}_1.\underline{O}.\underline{B}'_1).p'$ (e) $r'.\Lambda(\underline{A}'_2.\underline{O}'.\underline{B}_2).r$

Figure 87:

Therefore, if the value of the one-way infinite paths are accepted as O, then the reduction will be valid.

Theorem 21 *If a formula is not a tautology, then in the totally reduced form, there are only invalid e-products.*

Proof 21 *Assume there is no elimination in any reduced form. Then apply the elimination to the totally reduced form. What remains at hand would be either 0 or 1. If the result of elimination is a* 0 *in a formula, this means that all the e-products in the formula are invalid. Since only if a formula is not a tautology, the totally reduced formula produce* 0, *the totally reduced form contains all invalid e-products iff the formula is not a tautology.*

Thus, in a totally reduced form of a non tautological formula, all e-products either include a path terminate with 0 or a unidirectional infinite path. This means that in the totally reduced form of a tautology, at least one e-product includes neither a path terminate with 0 nor a unidirectional infinite path. We know that in a totally reduced formula, each path is either infinite or terminate with a 1 or 0. If a path is not unidirectional then it is returning. Therefore in the totally reduced form of a tautology, all the paths of at least one e-product either terminate with 1 or return.

Definition 16 *1. Any unidirectional infinite path or any path terminates with* 0 *are invalid paths.*

2. Any path terminate with 1 *or any returning path are valid paths.*

Definition 17 *1. If an e-product has an invalid path then it is an invalid e-product.*

2. If all the paths of an e-product are valid then it is a valid e-product

6 Elementary sums

We can talk about the elementary sums of a formula as well as the e-products. In a totally unreduced formula, the e-sums can be found with the following algorithm.

Algorithm 6.1 *1. This algorithm finds an e-sum from p.*

2. push the arc p to the stack.

3. repeat the following until the stack is empty.

   (a) pop an arc from the stack. Let it be p. Then push p' to the stack.

   (b) If p and its connection is an OR-node, i.e. is in the form $p \dashv (dizir)$, then push r_i to the stack for every $i = 1, .., n$.

   (c) If p and its connection is an AND-node, i.e. is in the form $p \bullet (dizir)$, then for one $i = 1, .., n$, push r_i to the stack.

   (d) If p and beyond is in the form $p \vdash r$ or pr, then push r to the stack.

   (e) If p and beyond is in the form px, then push x to DEG.

4. The sub formula scanned is an e-sum. The statement correspondence of this is found by disjunction of literals in DEG.

In the figure, a valid e-sum and the reduced form of it by x reduction.

Figure 88: an e-sum and the reduced form of it

In a totally unreduced formula, choosing a path from all the e-sums is equivalent to scan an e-product of the formula. Similarly scanning an e-sum is equivalent to scan only one path in every e-product.

Then we can write the followings:

Theorem 22 *1. In any formula, choosing a path from all the e-sums is equivalent to scan an e-product of the formula.*

2. In any formula, choosing a path from all the e-products is equivalent to scan an e-sum of the formula.

Proof 22

Then in a totally reduced formula, if every e-product has an unidirectional infinite path or a path terminate with 0, then there is at least one e-sum whose all paths are either unidirectional infinite or terminate with 0.

Definition 18 *1. If all the paths of an e-sum are invalid then it is an invalid e-sum.*

2. If a path of an e-product is valid then it is a valid e-sum.

Validity of Arches

Definition 19 *An arc is valid iff it's formula has a valid e-product.*

The value of arches are not independent from each other. For example, for pr if r is valid then p is valid. In the following, how the validity of an arc affect other are shown.

1. For $p \bullet (r_1..r_n)$, let for all $i = 1, ..n$, r_i be valid. I.e. let starting from any r_i there is at least one valid e-product. Then there will be a valid e-product starting from p. Therefore p would be valid.

   Now let there is at least one r_i whose every e-product is invalid. I.e. every e-product includes an invalid path. In this case all the e-products starting from p includes an invalid path. Then p would be invalid.

2. For $p \dashv (r_1, .., r_n)$, let one r_i be valid. This means that from starting r_i there is a valid e-product. This means that from p there is a valid e-product. On the other hand let p is valid. Then this means that there is one valid e-product starting from p. I.e. there is an e-product whose every path is either returning or terminates with 1. However, this does not imply that any r_i is valid. Assume the valid e-product starting from p has a path $p \dashv r_i..p'...$ Then the e-product starting from r_i would have a path $r_i..p'..$ but this may not be returning.

An observation is that every arc of an invalid e-sum is invalid.

Figure 89:

In the figure, an e-sum whose every path is unidirectional and closed is shown.

Theorem 23 *any e-sum whose all paths are unidirectional infinite or end with* 0 *is invalid.*

Proof 23 *Whichever arc is selected, from it there is an e-sum whose all paths are invalid.*

Theorem 24 *Let p be an arc. f both p and p' is valid, then the all arches of an e-product starting from p is valid.*

Proof 24

Figure 90: A closed e-sum

7 Finding Tautology

7.1 A class of Algorithms

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

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 $p \dashv (r_1, .., r_n)$, $O(p) \geq \max(O(r_1), .., O(r_n))$.
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 7.1 *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 $\twoheadrightarrow\underline{A}$, often $\underline{A}$ is used.

Theorem 25 *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 25 *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 20 *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 7.2 *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 26 *Let $O(p') \geq O(p)$. Then all the paths of a MES scan starting from p are MDREG.*

Proof 26 *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 27 *If a path is MDREG and in the form $..r..r'..$ then $O(r) = O(r')$.*

Proof 27 *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 7.3 *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 28 *Let p is a REG arc. Then any min-max path starting from p is a MDREG path.*

Proof 28 *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.*

7.1.1 An algorithm

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

Algorithm 7.4

1. *Let $O(0) = 0$ and $O(1) = 1$.*
2. *Let initially every arc has the same order $\flat$ where $1 > 0 > \flat$.*
3. *Apply the EXPAND algorithm to the graph.*
4. *Repeat the following until there is no arc p such that $O(p) = \flat$. Initially $ORD = \{\}$*
   (a) *Choose an arc p such that $O(p) = \flat$. Let $O(p) = L$ where $0 < L < 1$ and L is not a member of ORD. Also $ORD \cup \{L\} \to ORD$. Here p is called the source of L and denoted by $p = S(L)$. Let $O(p) = L$ and $T(p) = \{L\}$.*
   (b) *Apply the EXPAND algorithm to the graph.*

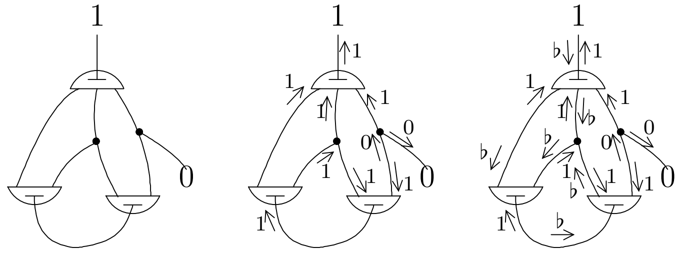
If at the end of the algorithm $O(k) = 1$ where k is the root arc, then the formula is valid and a tautology. Otherwise, it is not a tautology.

Algorithm 7.5 *EXPAND algorithm*

1. *Repeat the following for every arc of the graph until there is no change in the orders of them.*
   (a) *Get an arc p.*
   (b) *For $p \dashv (r_1, .., r_n)$, let $O_c(p) = \max(O(r_1), .., O(r_n))$. If $O(p) < 1$ and $O_c(p) = 1$ then $1 \to O(p)$ and $\{\} \to T(p)$. Otherwise if $O(p) < O_c(p)$ and for every $i = 1, .., n$ $O(r_i) \neq \flat$ then $O_c(p) \to O(p)$ and $T(r_k) \to T(p)$ where $O(r_k) = \max(O(r_1), .., O(r_n))$.*
   (c) *For $p \bullet (r_1, .., r_n)$, let $O_c(p) = \min(O(r_1), .., O(r_n))$. If $O(p) < O_c(p)$ and for every $i = 1, .., n$ $O(r_i) \neq \flat$ then $O_c(p) \to O(p)$ and $T(r_1) \cup .. \cup T(r_n) \to T(p)$*
   (d) *for $p \vdash r$, $O_c(p) = O(r)$. If $O(p) < O_c(p)$ and $O(r) \neq \flat$ then $O_c(p) \to O(p)$ and $T(r) \to T(p)$.*
   (e) *for $p0$, $O_c(p) = 0$. If $O(p) < O_c(p)$ let $0 \to O(p)$ and $T(p) = \{0\}$.*
   (f) *for $p1$, $O_c(p) = 1$. If $O(p) < O_c(p)$ let $1 \to O(p)$ and $T(p) = \{\}$.*
   (g) *If $p' = S(L)$ (i.e. p' is the source for the order L) then let $T(p) - \{L\} \to T(p)$. If $T(p) = \{\}$ then let $O(p) = 1$. Otherwise let $O(p) = \min(T(p))$.*

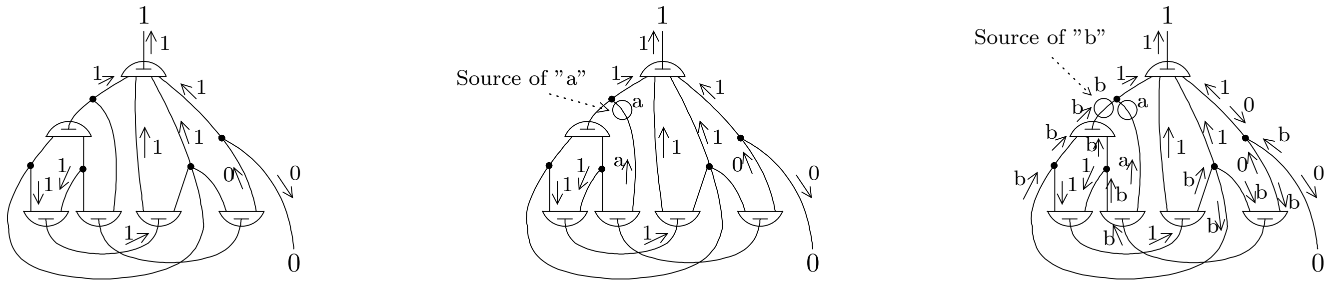
For example, for $p \bullet (r, s)$ let $O(r) = a$, $O(s) = b$ and $O(p) = \flat$ where $\flat < a < b$. Then $O(p)$ would be a and $T(p) = T(r) \cup T(s)$. For $p \dashv (r, s)$ let $O(r) = a$, $O(s) = b$ and $O(p) = \flat$ where $\flat < a < b$. Then $O(p)$ would be b and $T(p) = T(s)$.

Note that for an order L, the algorithm guarantees that if $0 < L < 1$ then there is only one source of L. It also guarantees that if p is a source of L where $0 < L < 1$ then $O(p) \neq O(p')$.

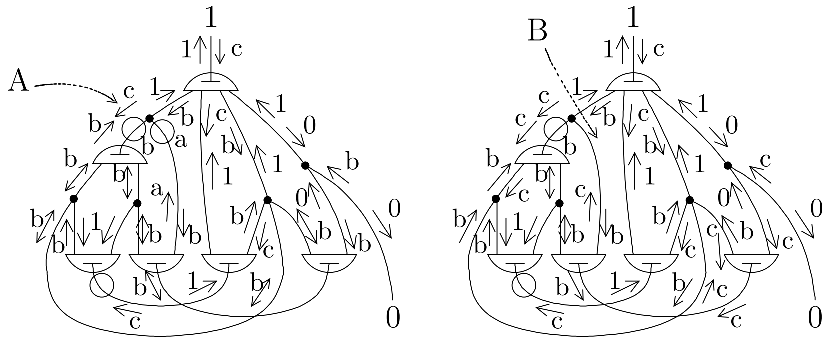


(a) (b) (c)

Figure 91: Markings. If there is no mark it is $\flat$



(a) (b) (c)



(d) A: the reverse arc of the source of "b" becomes "c" because $T() = \{b, c\}$ at that arc

(e) B: the source is erased because the calculated order "c" is greater than "a"

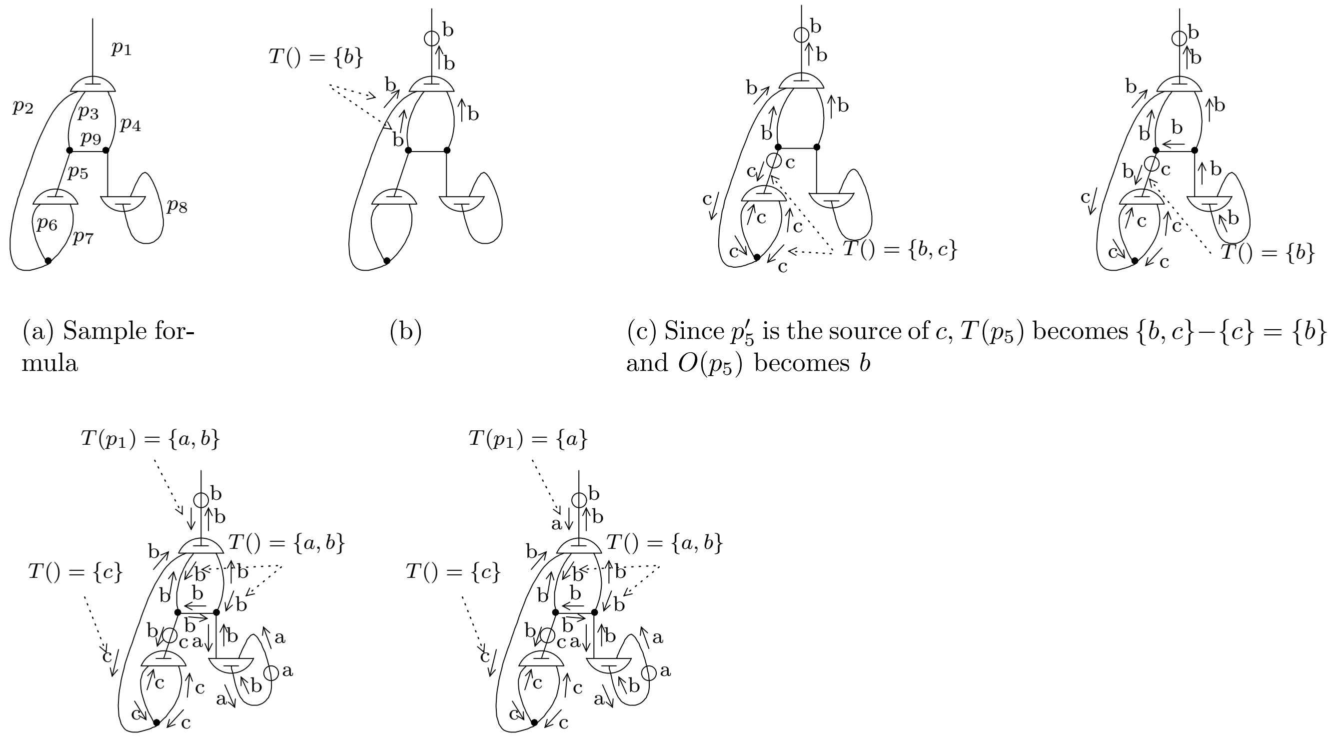
Figure 92: Markings where $1 > c > b > a > 0 > \flat$

Complexity The EXPAND algorithm is finite and its complexity is about $O(n^3)$. To see this consider the fact that the orders on arches always remains same or increases. Also the number of different orders is about n where n is the number of arches. So an arc is marked at most n times. n different labels can be coded with $\log n$ bits. So the comparison takes $\log n$ times. However, copying $T()$ for n different orders takes $O(n)$ time. As a result, the time complexity for n arches would be $\leq O(n^3)$.

In the main algorithm, EXPAND is applied for at most n sources. So the total time complexity is $\leq (n^4)$.

Validity

Definition 21 *Let consider a MEP scan where its every path has a source on it. Now form a partial*



(a) Sample formula

(b)

(c) Since p'_5 is the source of c, $T(p_5)$ becomes $\{b, c\} - \{c\} = \{b\}$ and $O(p_5)$ becomes b

(d) Since p'_1 is the source of b, $T(p_1)$ becomes $\{a, b\} - \{b\} = \{a\}$ and $O(p_1)$ becomes a

Figure 93: Another demonstration where $\flat < 0 < c < b < a < 1$

e-product from this such that every terminal of it is either a literal(0 *or* 1*) or a source of an order* L *where* $0 < L < 1$*. The literals* 0 *and* 1 *are the sources of the orders* 0 *and* 1 *respectively. Then this is a MEP scan whose paths terminate at sources. Let call this MEPST(Maximal E-Product with Source Terminals).*

The following algorithm finds a MEPST scan from an arc.

Algorithm 7.6 *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) If p is a source then do nothing.
  (c) For $p \dashv (r_1, .., r_n)$*, choose* r_i *such that* $O(r_i) = \max(O(r_1), .., O(r_n))$ *and put to BUF.*
  (d) For $p \vdash r$*, choose* r *and put to BUF.*
  (e) For $p \bullet (r_1, r_2)$*, choose* r_1 *and* r_2 *and put to BUF.*
  (f) For $p1$ *or* $p0$ *do nothing.*

3. The e-product scanned is a MEPST.

For example in the following figure a MEPST scan from the arc p is shown.

In fact, in the given algorithm, $T(p)$ represents the terminal set of a MEPST scan starting from p.

Theorem 29 *Let consider any MEPST scan from p. Let* $\{t_1, .., t_m\}$ *is the set of all the terminals of it. Then* $O_c(p) = \min(O(t_1), .., O(t_n))$*.*

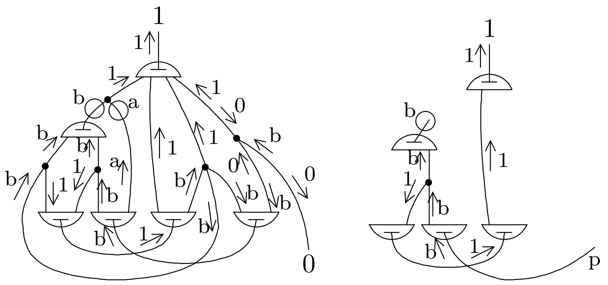


Figure 94: A MEPST scan from the arc p for the markings given. $T(p)$ represents the terminals of this scan, hence $T(p) = \{b\}$

Proof 29 *By algorithm, in the OR-nodes always the base outgoing arc having the maximal order is chosen.*

Theorem 30 *If an arc has an order L where $L \neq \flat$ then there is a MEPST scan from it.*

Proof 30 *We will first assume that no source is erased once generated.*

In the algorithm, if an arc is a source then obviously there is a MEPST scan from that arc. For the other arches, an arc p is marked iff all the outgoing arches of the node to which p is connected do not have order $\flat$. So if the sources are not overwritten, then every path starting from the arces marked with an order different than $\flat$ has a source on it. This means that there is a MEPST scan from such an arc.

Now consider any arc p whose one of the MEPST scan has terminals $s_1, .., s_m$. Assume there is a MEPST scan from one of these sources, for example s_1 and $t_1, .., t_n$ are the terminals of this scan. Assume for an $i = 1, .., n$, $t_i = s_1$. Then $O(s_1) \geq \min(O(t_1), .., O(t_n))$ and thus, since $O_c(s_1) = \min(O(t_1), .., O(t_n))$, $O(s_1) \geq O_c(s_1)$. So there would be no change on s_1 and s_1 remains as source. On the other hand, assume, for any $i = 1, .., n$, $t_i \neq s_1$. Then if s_1 is overwritten and is no longer a source, then the MEPST scan starting from p changes and its terminals becomes $t_1, .., t_n, s_2, .., s_m$ where all of these terminals are still sources.

Theorem 31 *Let p' be the source of L. If $T(p) = \{L\}$ then p is valid.*

Proof 31 *...*

Theorem 32 *Let $O(p) = 1$. Then the formula starting from p is valid.*

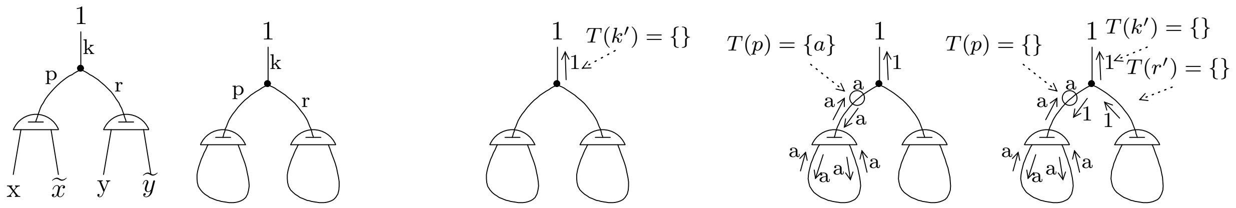
Proof 32 *If $O(p) = 1$ then there is a MEPTS scan whose all terminals are sources of 1. There are only two sources of 1 in the algorithm. The first one is the literal 1. The other is an arc r where $O(r) = L$ and $r' = S(L)$. In either case the paths terminate with valid arces and therefore p is valid.*

Invalidity

Theorem 33 *Let for an arc p, $O(p) = L$ and $O(p') \geq O(p)$. Assume r is the source of L. Then there is a min-max path in the form $p\underline{A}r$ where $O(p\underline{A}r) = L$.*

Proof 33 *From the algorithm.*

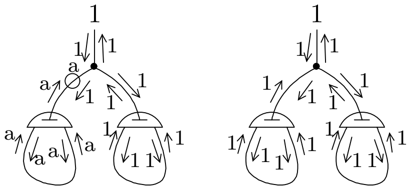
Theorem 34 *Let for an arc p, $O(p') > O(p)$ and $O(p) \neq \flat$. Then there is always a unidirectional min-max path $p..r$ where r is a source.*



(a) The sample formula and totally reduced form

(b) Applying EXPAND algorithm initially

(c) Here first $T(p)$ is found as $\{a\}$. However, since $p' = S(a)$, it becomes $\{\}$



(d) The source of "a" is overwritten. $O(k) = 1$, hence the formula is valid

Figure 95: A valid formula

Proof 34 *Since $O(p) \neq \flat$, any min-max path starting from p has a source on it. Let $p\underline{A}r$ be a min-max path where r is a source.Since p is a REG arc, this path is MDREG. Assume it is returning. Now this path can not include r' because r is a source and thus $O(r') \neq O(r)$. So it can not be in the form $p..r'..r$ since this is a MDREG path and in a MDREG path this is possible only if $O(r') = O(r)$. Similarly it can not be in the form $p..p'..r$ because $O(p') \neq O(p)$. Assume it is in the form $p\underline{A}r = p..s..s'..r$ where $s \neq r$ and $s \neq p$. Then apply the following rules.*

1. *The path is in the form $p\underline{B}ş \vdash s..s' \dashv ş'..r$. Then this means that the path is in the form $p..ş..ş'..r$. Apply the same logic to this path along for $ş$.*

2. *The path is in the form $p\underline{B}ş \vdash s..s' \dashv t..r$. Since the path is MDREG, $O(ş') \geq O(ş)$ and $O(ş) \geq O(t)$. This means that $O(ş') \geq O(t)$. On the other hand this node can be written as $s' \dashv (ş', t, ..)$ and since this is a min-max path $O(t) = \max(O(ş'), O(t), ..)$ must be true. From here we deduce that $O(ş') = \max(O(ş'), O(t), ..)$ and thus $O(ş') = O(t)$. Since $O(s) \geq O(t)$, it is found that $O(ş') = O(ş)$. There is a min-max and MDREG path $ş'\underline{C}t$ where t is a source. Now form a new path $p\underline{B}ş \vdash s..s' \dashv ş'\underline{C}t$. Clearly this is a min-max path and in the form $p..ş..ş'..t$. Apply the same logic to this path along for $ş$.*

3. *The path is in the form $p\underline{B}ş \bullet s..s' \bullet ş'..r$. Then this means that the path is in the form $p..ş..ş'..r$. Apply the same logic to this path along for $ş$.*

4. *The path is in the form $p\underline{B}ş \bullet s..s' \bullet t..r$ where the AND node is $ş \bullet (s, t, ..)$. Since this is a min-max path $O(s) = \min(O(s), O(t), ..)$. On the other hand $O(t) = \min(O(ş'), O(t), ..)$. Therefore $O(s) = O(t)$. Then clearly the path $p\underline{B}ş \bullet t..r$ is a min-max path. If this is a unidirectional path then the theorem is proved. Otherwise apply the same logic to this path along for the returning arces.*

At last we get a path where the above rules can not be applied. Clearly this is a unidirectional, min-max and MDREG path.

Definition 22 *If a path is not in the form $p..r \vdash s..s' \dashv t..$ then it is a primal path.*

Theorem 35 *Let the order of an arc p is L and $O(p') > O(p)$. Then there is always a complete unidirectional MDREG path starting from p.*

Proof 35 *It has been proved that, for p, if $O(p') > O(p)$ and $O(p) \neq \flat$ then there is always a unidirectional MDREG path $p\underline{A}r$ where r is a source. Since r is a source and REG, $O(r') > O(r)$. Therefore, according to the previous theorem, there is a unidirectional min-max path starting from r. Let this be $r.s\underline{B}t$ where t is a source. Now we will show that $p\underline{A}r.s\underline{C}t$ is a unidirectional min-max path.*

Clearly $p\underline{A}r.s\underline{C}t$ is a min-max path. Now assume it is returning. Since $p\underline{A}r$ and $r.s\underline{B}t$ are non returning paths, it can be only in the form $p..\c{s}..r.s..\c{s}'..t$. If $t \neq r$ then according to the previous theorem, there is a unidirectional path $p..t$. Also $O(t) < O(r)$. Since the number of sources is at most n, at last we get a path in the form $p..\c{s}..r..\c{s}'..r$ where r is a source. Now we will show that there is a unidirectional min-max path in the form $p..r..r$.

If the path $p\underline{A}\mathbf{r}\underline{B}\mathbf{r}$ is not a primal path then apply the following rules to get a primal min-max path.

1. *The path can not be in the form $p..\c{s} \vdash s..s' \dashv t..\mathbf{r}..\mathbf{r}$ because $p\underline{A}\mathbf{r}$ is a unidirectional path.*

2. *The path is in the form $p\underline{A}_1\c{s} \vdash s\underline{A}_2\mathbf{r}\underline{B}_1s' \dashv t\underline{B}_2\mathbf{r}$ where $p\underline{A}_1\c{s} \vdash s\underline{A}_2\mathbf{r}\underline{B}_1s'$ is a primal path. Then there is a path in the form $p\underline{A}_1\c{s} \vdash s\underline{A}_2\mathbf{r}\underline{B}_1s' \dashv \c{s}'\underline{C}\mathbf{r}$. Assume in $\underline{B}_1$ there is a $\c{s}$. Then there is a min-max primal path $p\underline{A}_1\c{s} \vdash s\underline{A}_2\mathbf{r}\underline{B}_{11}\c{s} \vdash s\underline{A}_2\mathbf{r}$. Otherwise apply the same rules to the path $p\underline{A}_1\c{s} \vdash s\underline{A}_2\mathbf{r}\underline{B}_1s' \dashv \c{s}'\underline{C}\mathbf{r}$.*

3. *The path is in the form $p\underline{A}\mathbf{r}B_1\c{s} \vdash s\underline{B}_2s' \dashv t\underline{B}_3r$. Then there is a path in the form $p\underline{A}r\underline{B}_1\c{s} \vdash s\underline{B}_2s' \dashv \c{s}'\underline{C}r$. In $\underline{A}$ there can not be a t' arc, because otherwise the path would be in the form $p\underline{A}_1t' \vdash s\underline{A}_2r\underline{B}_1\c{s} \vdash s\underline{B}_2s' \dashv t\underline{B}_3r$ and from here a min-max primal path $p\underline{A}_1t' \vdash s\underline{A}_2r\underline{B}_1\c{s} \vdash s\underline{A}_2r$ is derived.*

Therefore, we get a primal min-max path in the form $p\underline{A}\mathbf{r}\underline{B}\mathbf{r}$. Now apply the following rules to this path to get a unidirectional path.

1. *The path is in the form $p\underline{A}_1s\underline{A}_2\mathbf{r}\underline{B}_1s'\underline{B}_2\mathbf{r}$ where $p\underline{A}_1s\underline{A}_2\mathbf{r}\underline{B}_1$ is a unidirectional path.*
   - *(a) The path can not be in the form $p\underline{A}_1\c{s}.s\mathbf{r}\underline{B}_1s'.\c{s}'\underline{B}_2\mathbf{r}$.*
   - *(b) The path can not be in the form $p\underline{A}_1\c{s} \vdash s\mathbf{r}\underline{B}_1s' \dashv t\underline{B}_2\mathbf{r}$ because this is a primal path.*
   - *(c) The path is in the form $p\underline{A}_1\c{s} \bullet s\underline{A}_2\mathbf{r}\underline{B}_1s' \bullet t\underline{B}_2\mathbf{r}$ where the AND node is $\c{s} \bullet (s, t, ..)$. Since this is a min-max path $O(s) = \min(O(s), O(t), ..)$. On the other hand $O(t) = \min(O(\c{s}'), O(t), ..)$. Therefore $O(s) = O(t)$. Then clearly the path $p\underline{A}_1\c{s} \bullet t\underline{B}_2r$ is a min-max path. If this is not a unidirectional path apply the same logic to this path along for the returning arches. Otherwise form a new path $p\underline{A}_1\c{s}\bullet t\underline{B}_2\mathbf{r}\underline{B}_1s'\bullet t\underline{B}_2\mathbf{r}$ which is unidirectional.*

Thus we get a unidirectional min-max path in the form $p\underline{A}\mathbf{r}\underline{B}\mathbf{r}$. Then clearly there is a unidirectional min-max path $p\underline{A}\mathbf{r}(\underline{B}\mathbf{r})^\infty$.

Theorem 36 *Let the order of an arc p is L and $p' \geq p$. Then there is always a MES scan whose all paths are unidirectional.*

Proof 36 *It has been shown that if $O(p') \geq O(p)$ then there is always a unidirectional min-max path starting from p. This path is a member of MES scan.Now consider other paths of the MES scan. Let the min-max path be $p\underline{A}r \dashv s_1..$ where in the node $r \dashv (s_1, ..s_n)$ $O(s_1) = \max(O(s_1), .., O(s_n))$. Now consider the arc s_2. Since this is a MES scan, all of the paths of it MDREG. So obviously $O(s_2') \geq O(s_2)$. This means that there is a unidirectional min-max path from s_2. Let this be $s_2\underline{B}$.*

Now we will show that $p\underline{A}r \dashv s_2\underline{B}$ *is a unidirectional path. Assume it is not. Then it must be in the form* $p..ş..r \dashv s_2..ş'...$ *Since this path is a MD path and* $O(s_2) < O(r)$, *then* $O(ş') < O(ş)$. *However, this is not possible because this is a MDREG path and* $O(ş') \geq O(ş)$ *must be true. So the path* $p\underline{A}r \dashv s_2\underline{B}$ *can not be a returning path.*

Theorem 37 *Let* $O(k) = L$ *where* $\flat < L < 1$ *and* k *is the root arc. Then this is an invalid formula.*

Proof 37 *Since* k' *is terminated with a* 1, $O(k') = 1$. *So this means that* $O(k') > O(k)$. *Then there is a MES scan whose all paths are unidirectional. Since all of these paths are MD, any of these paths can not terminate with* 1. *So any path on it is either infinite or terminate with* 0. *Therefore the formula is invalid.*

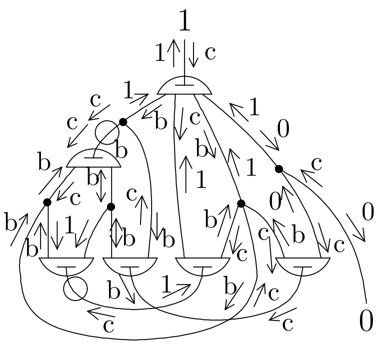


Figure 96: An invalid formula where $1 > c > b > a > 0 > \flat$

References

[1] P. Lammens, L. Claesen, H. De Man, 1989, Correctness Verification of VLSI Modules Supported by a Very Efficient Boolean Prover, Proceedings: IEEE International Conference on Computer Design, October 1989, 266-269

[2] John E. Hopcroft, Jeffrey D. Ullman, "Introduction to Automata Theory, Languages and Computation,p.322,pp328-329, Addison-Wesley, 1979