The Recognition of the AOSP Digraphs

Hsin-Hung Chou and De-Ron Liang

October 6, 1998

Abstract

A computation task running in distributed systems can be represented as a directed graph, called as a *task graph*. In such graph, vertices represent modules and arcs represent precendence constraints. And the arguments are passing among the modules. In the past, people study such subjects on edge series-parallel(ESP) digraph. In an ESP digraph, the precedence relation among the arcs joining to a vertex defaults to be OR-relation. In this paper, we extend the precedence relation to being a factorable formulas. Such digraphs are called and-or series-parallel(AOSP) digraph. And we present a polynomial time algorithm to recognize the class of the AOSP digraphs.

1 Basic concepts

In this section, we resume some definitions and notations we shall employ.

1.1 Graph-theoretic definitions [1]

A graph $G = (V, E)$ consists of a finite set of vertices V and a finite set of edges E. Each edge is a pair (v, w) where v and w are distinct vertices. If the edges of G are unordered pairs, then G is an **undirected** graph; if the edges are ordered pairs, then G is a **directed graph**(abbreviated **digraph**). A graph is **connected** if for each pair of vertices,v and w , there is a path from v to w. The **connected components** of a graph G are the maximal connected subgraphs of G. An **acyclic** graph is one that contains no cycles. A **tree** is a connected acyclic graph.

1.2 Boolean logic definitions [2]

A **Boolean variable** is denoted by x_i to represent a Boolean value **true** or **false** but not both. The Boolean variables and the negations of variables will be spoken of collectively as **literals.** If x_1 and x_2 are Boolean variables, the **conjunction** of x_1 and x_2, $x_1 \wedge x_2$, is true if both x_1 and x_2 are true. If either x_1 or x_2 is false, or if both are false, $x_1 \wedge x_2$ is false. Symmetrically, the **disjunction** of x_1 and x_2, $x_1 \vee x_2$, is true if at least one of the Boolean variables x_1 or x_2 is true, and is false only if both x_1 and x_2 are false. A **formula** is made up of literals, conjunctions, and disjunctions. A **positive formula** is a formula made up without negative variables. Two formulae F_1 and F_2 are said to be **logically equivalent**, denoted by $F_1 \Leftrightarrow F_2$, provided that the formula F_1 is true(respectively,false) if and only if the formula F_2 is true (respectively,false).

A conjunction of literals such that no variable appears in it twice will be called a **fundamental conjunctive formula.** Any disjunction of fundamental conjunctive formulae will be called a **disjunctive normal formula** or a formula in **disjunctive normal form.** The fundamental conjunctive formulae in a disjunctive normal formula F will be called the **clauses** in F. A disjunctive normal formula with minimum number of clauses is regarded as **irreducible.**

The same considerations as in the set theory, a clause C_1 is said to be a **subclause** of a clause C_2 provided that the set of the literals in the clause C_1 is a subset of the set of the literals in the clause C_2. In this case, we write $C_1 \subseteq C_2$ and we say that C_1 **is included** in C_2. Two clauses C_1 and C_2 are said to be **distinct** if $C_1 \not\subseteq C_2$ and $C_2 \not\subseteq C_1$. In the contrary, two normal formulae, F_1 and F_2, are said to be **isomorphic**, denoted by $F_1 \cong F_2$, since $C_1 \subseteq C_2$ and $C_2 \subseteq C_1$. When the literal sets of the two formulae have no common element, we say that they are **disjoint.**

A **factoring** on a Boolean formula F is an operation that divides F into two disjoint subformulae, F_1 and F_2 which are called the **factors** of F, in the Boolean equation : $F = F_1 \odot F_2$, where $\odot$ is a Boolean relation, $\wedge$ or $\vee$. Obviously, not all the Boolean formulae can be factored. A formula that can be factored recursively such that factors are factored into subfactors until all the factors or subfactors are single literals, is called a **factorable formula.**
In the other way, we use a **factoring tree** to indicate a sequence of the recursive factoring operations, where the factoring tree $T = (V_T, E_T)$ is a binary tree in which each node in V_T has a **sort** in $\{\wedge, \vee\} \cup \mathcal{L}$, where $\mathcal{L}$ is the set of literals. To a factoring on a Boolean formula F,$F = F_1 \odot F_2$, the corresponding factoring tree has the structure that the root of the tree

has the sort $\odot$, the left subtree is the factoring tree of F_1, and the right subtree is the factoring tree of F_2. Besides, the sorts of the internal nodes in a factoring tree are the Boolean operations, $\wedge$ or $\vee$, and the sorts of the external nodes are single literals. It is easy to see that the factoring tree of a factoring formula is not unique. In the other words, there are not only one way to factor a formula. Inversely, as we know, a factoring tree represents a unique factoring formula, we call the formula being **expanded** by the factoring tree. Two factoring trees T_1 and T_2 are said to be **logically equivalent**, denoted by $T_1 \equiv T_2$, if the normal formulae expanded by these two trees are isomorphic.

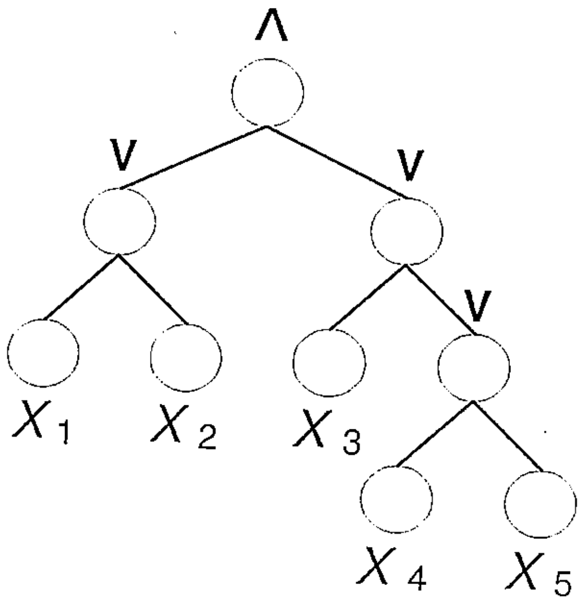

Figure 1: An example for the factoring tree.

1.3 And-Or Series-Parallel (AOSP) digraphs

AOSP digraphs are the extension of ESP digraphs. We introduce the class of ESP digraphs recursively as follows [3]:

Definition 1 *The class of* **ESP** *(***Edge Series-Parallel***) digraphs.*

1. *A digraph consisting of two vertices joined by a single edge is ESP.*

2. *If G_1 and G_2 are ESP digraphs, so are the digraphs constructed by each of the following operations:*

   (a) **Series composition**: *Identify the sink of G_1 with the source of G_2.*

   (b) **Parallel composition**: *Identify the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2.*

The definition of the class of the AOSP digraph is introduced as follows:

Definition 2 *The class of* **AOSP***(***And-Or Series-Parallel***) digraphs : A digraph $G = (V, E, B)$ is an AOSP digraph if and only if the digraph constructed by (V,E) is an ESP digraph and all the elements in B are factorable formulae.*

graph such that the vertices are one-to-one literals between any two clauses.

2 The Recognizing Algorithms for AOSP Digraphs

Since the class of the AOSP digraphs is derived from the task digraphs, it is appropriate to assume that the input formulae are the disjunction normal formulae with positive literals. There are two major steps to recognize the AOSP digraphs: the recognition of the ESP digraph and the recognition of the factoring Boolean formulae. Since the recognition of the ESP digraphs was provided by Jacobo Valdes, Robert E. Tarjan and Eugene L. Lawler [3], all we have to do is to recognize the factorable Boolean formulae.

- INPUT: A digraph $G =< V, E, B >$ where V is a finite set of vertices. E is a finite set of edges and B is a finite set of boolean formulae attached to each vertex in V.
- OUTPUT: If G is an $AOSP$ digraph, then output "YES" else output "NO".
- ALGORITHM:

  RECOGNITION(V,E,B)

```
1 Checking whether the digraph $G' =< V, E >$ is an $ESP$ digraph
  or not by Valdes' algorithm. If the answer is "NO", then output
  "NO" and STOP.
2 For each boolean formula $F$ in $B$:
      Call FACTORING(F)
3 Return

FACTORING(F)

1 If F is a single literal then Return("YES").
2 Reduce $F$ to be the irreducible disjunction normal formula, $DF$.
3 Construct the clause connected graph,$IG$, from $DF$.
4 Find the connected components, $IG_i$, of $IG$. And let $DF_i$ be the
  subformula in disjunction normal form corresponding to $IG_i$.
5 For each subformula $DF_i$:
  5.1 If F is a single literal then Return("YES").
  5.2 Construct the literal connected graph $LG_i$ from $DF_i$.
  5.3 Find the connected components, $LG_{ij}$, of $LG_i$. And decom-
      pose $DF_i$ into the conjunction of the subformulas $DF_{ij}$ which
      are the formulae composed by the literals in $LG_{ij}$ respec-
      tively.
  5.4 If the number of the subformulas is equal to 1 and the number
      of the literals in $DF_i$ is more than 1, then output "NO" and
      STOP.
  5.4 For each subformula $DF_{ij}$:
          Call FACTORING($DF_{ij}$).
5 Return("YES").
```

3 The Correctness of The Algorithms

Theorem 1 *Given any positive disjunction normal formula F,if I is a redundant implicant in F, then there must exist another implicant I' in F such that $I' \subseteq I$.*

Proof. Let $F = I_1 \vee I_2 \vee \ldots \vee I_m \vee I$ and $DF = I_1 \vee I_2 \vee \ldots \vee I_m$. Since I is a redundant implicant in F, by definition, we have that $F \Leftrightarrow DF$.

And suppose there doesn't exist any implicant I' in F such that $I' \subseteq I$. This assumption implies that for each $I_j, 1 \leq j \leq m$, there exists at least a literal, denoted by x_j, being in I_j but not in I. Let's consider the truth assignment such that $x_j = false$ for $1 \leq j \leq m$ and the others are assigned to be *true*. Since the implicants, I_js, are the fundamental conjunction formulas, the truth assignment, $x_j = false$ for $1 \leq j \leq m$, would make $I_j = false$ for $1 \leq j \leq m$. And it implies that $DF = false$. Nevertheless, there is no x_j in I, the assignment would make I to be *true*. And it implies that $F = true$. It is a contradiction to the assumption, $F \Leftrightarrow DF$.
Therefore, there exists at least one implicant I' in F such that $I' \subseteq I$. □

Corollary 1 *Given any positive disjunction normal formula F, then F is irreducible if and only if each two implicants in F are distinct.*

Lemma 1 *Given any two irreducible positive disjunction normal formulas F_1 and F_2 such that $F_1 \Leftrightarrow F_2$, then $F_1 \sim F_2$.*

Proof. Let $F_1 = I_{11} \vee I_{12} \vee \ldots I_{1m}$ and $F_2 = I_{21} \vee I_{22} \vee \ldots \vee I_{2n}$, where I_{ij} are implicants for F_1 and F_2 respectively. Since F_1 and F_2 are irreducible, then each two implicants in F_1 are distinct and the same in F_2. Suppose $n > m$ then there exists at least one implicant I_{2i}in F_2 being distinct to all the implicants in F_1. Let's consider the formula $F = C_{2i} \vee C_{11} \vee C_{12} \vee \ldots \vee C_{1m}$. If C_{2i} is distinct to all the clauses in F_1, then there exists an assignment A such that all the boolean variables in C_{2i} are assigned to be true and others are false, and it implies that $F_1(A) = false$ and $C_{2i}(A) = true = F_2(A)$. Obviously, $F_1 \not\Leftrightarrow F_2$. It is a contradiction to the assumption that $F_1 \Leftrightarrow F_2$. Therefore we have the conclusion that $m = n$.
Since $m = n$, if $F_1 \not\sim F_2$, then there exists at least one clause C_{2i} in F_2 different from all the clauses in F_1. And we can prove that it is a contradiction in the same way.
Therefore we can say that F_1 and F_2 must be similar. □

Property 1 *Given any two disjoint irreducible positive normal formulas F_1 and F_2. Suppose $F = F_1 \vee F_2$, then we can see that each clause in F is composed by either the variables in F_1 or in F_2 and not both. F is also an irreducible positive normal formula and the number of the clauses in F is equal to the summation of the numbers of the clauses in F_1 and F_2. And the clause connected graph constructed by F is not a connected graph with just one connected component.*

Property 2 *Given any two disjoint irreducible positive normal formulas F_1 and F_2. Suppose $F = F_1 \wedge F_2$, then we can see that there is at least one clause in F containing both x_i and y_i for each boolean variable x_i in F_1 and y_i in F_2. F is also an irreducible positive normal formula and the number of the clauses in F is equal to the muliplication of the numbers of the clauses in F_1 and F_2. And the clause connected graph constructed by F is a connected graph.*

Property 3 *Given any two irreducible positive normal formulas F_1 and F_2. If $F_1 \sim F_2$ then the two clause connected graphs constructed by F_1 and F_2 are homogenuous.*

Property 4 *Given any two irreducible positive normal formulas F_1 and F_2. If $F_1 \sim F_2$ then the two literal connected graphs constructed by F_1 and F_2 are homogenuous.*

Corollary 2 *Given any boolean positive formula F, then the irreducible positive normal formulas reduced from F are all similar.*

Proof. Suppose there are two distinct irreducible positive normal formulas reduced from F, MF_1 and MF_2, such that $MF_1 \Leftrightarrow F$ and $MF_2 \Leftrightarrow F$. Thus, we have $MF_1 \Leftrightarrow MF_2$. And from Lemma 1, we have that $MF_1 \sim MF_2$. It is a contradiction to the assumption that MF_1 and MF_2 are two distinct formulas. Therefore, we have the conclusion that such formulas are all similar. □

Lemma 2 *Given any boolean binary decomposition tree T, then the positive normal formula expanded from T is irreducible.*

Proof. In a boolean binary decomposition tree, the leaves are distinct literals and the internal nodes are the boolean opeations $\wedge$ or $\vee$. Suppose T is a boolean binary decomposition tree with root node $\wedge$ and two subtrees T_1 and T_2. By induction, it is trivial that a formula with single literal is irreducible. Assume that the formulas expanded from T_1 and T_2 are respectively F_1 and F_2 which are irreducible. Let $F_1 = C_{11} \vee C_{12} \vee \ldots \vee C_{1m}$, $F_2 = C_{21} \vee C_{22} \vee \ldots \vee C_{2n}$.
Then $F = F_1 \wedge F_2 = (C_{11} \vee C_{12} \vee \ldots C_{1m}) \wedge (C_{21} \vee C_{22} \vee \ldots \vee C_{2n})$
$= (C_{11}C_{21} \vee C_{11}C_{22} \vee \ldots \vee C_{11}C_{2n}) \vee \ldots \vee (C_{1m}C_{21} \vee C_{1m}C_{22} \vee \ldots \vee C_{1m}C_{2n})$.
From the above assumption, we can easily see that there are no common literals between C_{1i} and C_{2j} for $1 \le i \le m$ and $1 \le j \le n$. In F_1, there is

no clause C_{1p} with all the literals in another clause C_{1q} for $1 \le p, q \le m$. and the same in F_2.
Thus, there is no clause $C_{1i}C_{2j}$ with all the literals in another clause $C_{1p}C_{2q}$ for $1 \le i, p \le m$ and $1 \le j, q \le n$. Then we have the conclusion that F is irreducible. □

Corollary 3 *Given any irreducible positive normal formula F and suppose F can be decomposed into the boolean binary decomposition tree T. Then the formula expanded from T is similar to F.*

Theorem 2 *Given any boolean positive formula F and suppose the irreducible positive normal formula reduced from F is DF. Then F is a BSP formula if and only if DF is decomposible.*

Proof. If F is a BSP formula then there exists a boolean binary decomposition tree T constructed by F' such that $F' \Leftrightarrow F$. Suppose $F"$ is the irreducible positive normal formula expanded from T and DF is the irreducible positive normal formula reduced from F. Obviously, $F" \Leftrightarrow F$ and $DF \Leftrightarrow F$. Implies $F" \Leftrightarrow DF$. And from Lemma 1, we know that $F" \sim DF$. Thus DF is decomposible.
Now turn to prove the sufficient condition. Since DF is reduced from F, then it is trivial that $DF \Leftrightarrow F$. And from the definition of a BSP formula, since DF is a BSP formula and DF is equivalent to F, then F is also a BSP formula. □

Lemma 3 *Given any irreducible positive normal formula F that can be decomposed into the boolean binary decomposition tree T by our algorithm, if there exists another algorithm that can decompose F into the boolean binary decomposition tree T', then T and T' are graphical equivalent.*

Proof At first, let's consider if F is decomposed into the disjunction form. Assume that F is decomposed by our algorithm into $F_1 \vee F_2 \vee \ldots \vee F_m$ where $F_1, F_2, \ldots, F_m$ are disjoint irreducible positive normal formulas and there exists another algorithm decompose F into $F'_1 \vee F'_2 \vee \ldots \vee F'_n$ where $F'_1, F'_2, \ldots, F'_n$ are disjoint irreducible normal formulas. According to the number of the connected components of the clause connected graph constructed by F, it implies that m = n. It is trivial that there exists one formula F'_j similar to F_i for each i.
Furthermore, let's consider if F is decomposed into the conjunction form. Suppose F is decomposed by our algorithm into $H_1 \wedge H_2 \wedge \ldots \wedge H_p$ where

$H_1, H_2, \ldots, H_p$ are disjoint irreducible positive normal formulas and there exists another algorithm decompose F into $H_1' \wedge H_2' \wedge \ldots \wedge H_q'$ where $H_1', H_2', \ldots, H_q'$ are disjoint irreducible normal formulas. According to the number of the connected components of the literal connected graph constructed by F, it implies that p = q. From the distribution of the conjunction operation. There exists one formula H_j' similar to H_i for each i. □

Theorem 3 *Given any irreducible positive normal formula F, if F cannot be decomposed by our algorithm, then F is not a BSP formula.*

Proof. Assume that F cannot be decomposed by our algorithm, but it can be decomposed by other algorithm. At first, let's consider if F is decomposed into the disjunction form, $F = F_1 \vee F_2$ where F_1 and F_2 are disjoint. Since F cannot be decomposed by our algorithm, the clause connected graph constructed by F is a connected graph. But from the decomposition that $F = F_1 \vee F_2$ where F_1 and F_2 are disjoint, the clause connected graphs constructed by F_1 and F_2 are also disjoint and it implies that the clause connected graph constructed by F contains more than one connected components. It is a contradiction.
Furthermore, let's consider if F is decomposed into conjunction form, $F = H_1 \wedge H_2$ where H_1 and H_2 are disjoint. There are two cases that a formula cannot be decomposed into conjunction form by our algorithm.Case 1: The literal connected graph constructed by F is a connected graph. Case 2: $F_1' \wedge F_2' \neq F$ where F_1' and F_2' are disjoint. Since $F = H_1 \wedge H_2$ where H_1 and H_2 are disjoint, the literal connected graph must contains more than one connected components. Obviously, it is contradict to case 1.
We know that the set of the literals in F_1' or F_2' are decided by the literal connected graph constructed by F, and since $H_1 \wedge H_2 = F$, then the literal connected graph constructed by $H_1 \wedge H_2$ is the same as the one constructed by F. From the property of the conjunction operation, we have that either $F_1' \sim H_1$ and $F_2' \sim H_2$ or $F'1 \sim H_2$ and $F_2' \sim H_1$. It is a contradiction to $F_1' \wedge F_2' \neq F$ but $H_1 \wedge H_2 = F$.
Therefore, if F cannot be decomposed by our algorithm, then it cannot be decomposed by others algorithms neither. □

[]

References

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications.*

Macmillan, 1976.

[2] W. V. Quine, "The problem of simplifying truth functions," *American Mathematical Monthly*, vol. 59, pp. 521-531, 1952.

[3] J. Valdes, R. E. Tarjan, and E. L. Lawler, "The recognition of series parallel digraphs," *Siam J. Comput.*, vol. 11, pp. 298 - 313, May 1982.