Range Allocation for Equivalence Logic

Amir Pnueli[1], Yoav Rodeh[1,2], and Ofer Shtrichman[1,2]

[1] Weizmann Institute of Science, Rehovot, Israel
[2] IBM Haifa Research Laboratory
{ amir | yrodeh | ofers }@wisdom.weizmann.ac.il

Abstract. The *range allocation* problem was recently introduced as part of an efficient decision procedure for deciding satisfiability of equivalence logic formulas with or without uninterpreted functions. These type of formulas are mainly used when proving equivalence or refinement between systems (hardware designs, compiler's translation, etc). The problem is to find in polynomial time a small finite domain for each of the variables in an equality formula φ, such that φ is valid if and only if it is valid over this small domain. The heuristic that was presented for finding small domains was *static*, i.e. it finds a small set of integer constants for each variable. In this paper we show new, more flexible range allocation methods. We also show the limitations of these and other related approaches by proving a lower bound on the size of the state space generated by such procedures. To prove this lower bound we reduce the question to a graph theoretic counting question, which we believe to be of independent interest.

1 Introduction

The range allocation problem was introduced in [PRSS98] as part of an efficient decision procedure for equivalence logic formulas with or without uninterpreted functions. These type of formulas are mainly used when proving equivalence or refinement (abstraction) between systems. Deciding satisfiability (and validity) of formulas with uninterpreted functions is of major importance due to their broad use in abstraction. We refer the reader to [BD94] and [PSS98], where these type of formulas are used for proving equivalence between hardware designs (former) and for translation validation, a process in which the correctness of a compiler's translation is proven by checking the equivalence of the source and target codes (latter).

In the past few years several different BDD-base procedures for checking satisfiability of such formulas have been suggested. (in contrast to earlier decision procedures that are based on computing congruence closure [BDL96] in combination with case splitting). Typically the first step of these procedures is the translation of the original formula φ to a function-free formula in equivalence logic ψ such that ψ is satisfiable iff φ is. Then, a procedure for checking satisfiability of E-formulas is used for deciding ψ. This second procedure is the focus of this paper.

Goel et al. suggest in [GSZAS98] to replace all comparisons in ψ with new Boolean variables, and thus create a new Boolean formula ψ'. The BDD of ψ' is calculated ignoring the transitivity constraints of comparisons. They then traverse the BDD, searching for a satisfying assignment that will also satisfy these constraints. Bryant et al. at [BV00] suggested to avoid this potentially exponential traversing algorithm by explicitly computing a small set of constraints that are sufficient for preserving the transitivity constraints of equality. By checking ψ' conjuncted with these constraints using a regular BDD package they were able to verify larger designs.

The method which we will present here, extends the method first presented in [PRSS98], where ψ's satisfiability is decided by allocating a small domain for each variable, such that ψ is satisfiable if and only if it is satisfiable over this small domain. To find this domain, the equalities in the formula are represented as a graph, where the nodes are the variables and the edges are the equalities and disequalities (*disequality* standing for $\neq$) in ψ. Given this graph, a heuristic called *range allocation* is used in order to compute a small set of values for each variable. To complete the process, a standard BDD based tool is used to check satisfiability of the formula over the computed domain.

In [RS01] we elaborate on this by generating a smaller graph than [PRSS98]. This is achieved by examining the original formula with uninterpreted function φ, instead of its translated version ψ.

In this paper, we extend the second part of [PRSS98], by suggesting a more general method of allocating finite domains to variables. Using the information in the graph generated by [PRSS98] or [RS01], we suggest and analyze different procedures for generating a small state space that is adequate for checking ψ. One of our main results is a general lower bound on the size of the state space generated by any method using only the information in this graph. The suggests the need for a more in depth investigation of the formula at hand, rather than only examining which atomic equalities appear in it.

2 Equivalence logic formulas

An equivalence logic formula (called an E-formula) has the following syntax:

$$\langle \mathit{Formula} \rangle \longleftarrow \langle \mathit{Term} \rangle = \langle \mathit{Term} \rangle | \neg \langle \mathit{Formula} \rangle \mid \langle \mathit{Formula} \rangle \vee \langle \mathit{Formula} \rangle$$

$$\langle \mathit{Term} \rangle \longleftarrow \langle \mathit{Variable} \rangle \mid \mathbf{ITE}(\langle \mathit{Formula} \rangle, \langle \mathit{Term} \rangle, \langle \mathit{Term} \rangle)$$

$\mathbf{ITE}(f,t,e)$ stands for **if** f **then** t **else** e. The E-formula φ is said to be satisfiable if there is some assignment of values to φ's variables that satisfies φ. Therefore, an E-formula φ with variables V is a function $\varphi : \mathbb{N}^V \rightarrow \{0,1\}$. However, not all such functions can be realized as E-formulas, for example $\varphi(a,b) = a > b$. Therefore, we will try to make a more accurate definition.

Definition 1. (partition): *Given a set V, we say that $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ is a* partition *of V if:*

1. $V = \alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_k$
2. *for all $i \neq j$ we have that $\alpha_i \cap \alpha_j = \emptyset$*

Given a partition $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ we denote $u \sim_\alpha v$ if there is $\alpha_i \in \alpha$ such that $u, v \in \alpha_i$. In other words, a partition α gives us an equivalence relation $\sim_\alpha$. In fact, there is a one-to-one correspondence between the set of equivalence relations on V and its set of partitions.

Given an assignment to the variables of V, $a : V \rightarrow \mathbb{N}$, we denote by *partition*(a) the partition of V that satisfies $a(v) = a(u) \iff v \sim_{partition(a)} u$. The following is immediate from the fact that E-formulas only query comparisons between their input variables:

Claim. For E-formula φ, if assignments a, b satisfy *partition*(a) = *partition*(b) then $\varphi(a) = \varphi(b)$.

This means that E-formula φ on variables V can be viewed as a function of the partitions of V instead of as a function of assignments to V. Denote by $P(V)$ the set of partitions of the set V. φ can therefore be viewed as a function $\varphi : P(V) \rightarrow \{0,1\}$. In fact, any function from $P(V)$ to $\{0,1\}$ can be realized as an E-formula with variable set V.

It is easy to verify that letting each variable in an equivalence formula range over $\{1, \ldots, |V|\}$ suffices for checking the formulas satisfiability. This shows that deciding satisfiability of equivalence formulas is in NP, and therefore is clearly NP-complete (proved via a trivial reduction from satisfiability of boolean formulas).

3 Equivalence Graphs

Definition 2. (E-graph): *An* E-graph *$\mathcal{G}$ is a triple $\mathcal{G} = \langle V, EQ, DQ \rangle$ where V is the set of vertices, and EQ (Equality edges) and DQ (disequality edges) are sets of unordered pairs from V.*

We will use $\mathcal{G}$ and $\mathcal{H}$ to denote E-graphs, and G and H for standard graphs. For E-graph $\mathcal{G} = \langle V, EQ, DQ \rangle$ we denote $V(\mathcal{G}) = V$, $EQ(\mathcal{G}) = EQ$ and $DQ(\mathcal{G}) = DQ$. We denote by $\mathcal{G}_=$ the graph on vertices $V(\mathcal{G})$ and edges $EQ(\mathcal{G})$. We use $\leq$ to denote the subgraph relation: $\mathcal{H} \leq \mathcal{G}$ if $EQ(\mathcal{H}) \subseteq EQ(\mathcal{G})$ and $DQ(\mathcal{H}) \subseteq DQ(\mathcal{G})$.

We say that a partition α satisfies equality edge (u, v) if $u \sim_\alpha v$, and that it satisfies inequality edge (u, v) if $u \not\sim_\alpha v$. We say that a partition α satisfies E-graph $\mathcal{G}$ (denoted $\alpha \models \mathcal{G}$) if it satisfies all of $\mathcal{G}$'s edges. An E-graph is said to be satisfiable if there exists a partition that satisfies it.

Lemma 1. *An E-graph $\mathcal{G}$ is satisfiable iff for every $(u, v) \in DQ(\mathcal{G})$, u and v are not connected in $\mathcal{G}_=$.*

The algorithms of [PRSS98,RS01] construct for a given E-formula φ, an E-graph $\mathcal{G}$, satisfying the following property:

Definition 3. (adequacy of E-graphs for E-formulas): *The E-graph $\mathcal{G}$ is* adequate *for the E-formula φ if either φ is not satisfiable, or there exists a satisfiable $\mathcal{H} \leq \mathcal{G}$ such that any partition $\alpha \models \mathcal{H}$ satisfies $\varphi(\alpha) = 1$.*

Hence, if we want to check whether φ is satisfiable, we only need to check φ on a relatively small set of partitions:

Definition 4. (adequacy of partition sets for E-graphs): *For an E-graph $\mathcal{G}$ and a set of partitions $R \subseteq P(V(\mathcal{G}))$, we say that R is* adequate *for $\mathcal{G}$ iff for every satisfiable $\mathcal{H} \leq \mathcal{G}$, there is $\alpha \in R$ such that $\alpha \models \mathcal{H}$.*

This leads to the following claim:

Claim. If the partition set R is adequate for the E-graph $\mathcal{G}$ and $\mathcal{G}$ is adequate for the E-formula φ, then φ is satisfiable iff there is some $\alpha \in R$ such that $\varphi(\alpha) = 1$.

To use this claim, we need to devise a procedure that, given an E-graph $\mathcal{G}$, will find a small partition set R which is adequate for $\mathcal{G}$.

4 Static Range Allocation

The method of [PRSS98] generates for an E-formula φ, an E-graph $\mathcal{G}$ which is adequate for φ, and then uses it to find for each variable v of φ, a small range of natural numbers $D(v)$. It is then proved that φ is satisfiable iff it is satisfiable where each variable v is allowed to range only over $D(v)$ (as opposed to $\mathbb{N}$).

Therefore, we check φ over the assignment set $A_D = \{a \mid \forall v, a(v) \in D(v)\}$. The corresponding partition set is $P_D = \{\mathit{partition}(a) \mid a \in A_D\}$, and the construction of [PRSS98] guarantees that P_D is adequate for $\mathcal{G}$. In general, we will say that assignment set A is adequate for an E-graph $\mathcal{G}$ if its corresponding partition set is adequate for $\mathcal{G}$.

All methods proposed so far (including this paper), generate a new Boolean formula from the original E-formula φ. For example, the static range allocation method replaces every variable of φ by a variable ranging over a finite domain. This resulting formula is then translated to a Boolean formula using standard methods.

We will therefore measure the complexity of our proposed methods in terms of the number of Boolean variables in the newly generated Boolean formula (or equivalently, the size of the state space checked , which is 2 to the power of the number of Boolean variables). For example, in static range allocation, the state space size will be $\prod_v |D(v)|$, i.e., the number of boolean variables needed is $\sum_v \log(|D(v)|)$. This complexity measure is not always related to the true complexity of checking the resulting formula, but it is usually a good indicator for it.

5 Dynamic Range Allocation

We propose the new method of *dynamic range allocation* which improves upon the static range allocation method. Both experimental (Section 9) and theoretical (Section A) results show the advantages of dynamic over static range allocation.

Definition 5. (dynamic assignment): *The mapping $x : V \to \mathbb{N} \cup V$ is a* dynamic assignment *if it is acyclic; i.e., there exists an ordering of V, $v_1, \ldots, v_n$ such that, $x(v_i) = v_j$ implies $j > i$.*

Definition 6. (induced assignment): *For the dynamic assignment x, we define the* induced *(static) assignment $\overline{x} : V \to \mathbb{N}$:*

1. *If $x(v) \in \mathbb{N}$ then $\overline{x}(v) = x(v)$*
2. *If $x(v) = u \in V$ then $\overline{x}(v) = \overline{x}(u)$.*

Note that this is a recursive definition, and that it is well defined since we required that dynamic assignments be acyclic.

Example 1. The dynamic assignment: $x(v_1) = v_2, x(v_2) = v_3$ and $x(v_3) = 2$, induces the static assignment $\overline{x}$: $\overline{x}(v_1) = 2, \overline{x}(v_2) = 2, \overline{x}(v_3) = 2$.

For E-graph $\mathcal{G}$ and dynamic assignment x we denote $x \models \mathcal{G}$ if $\overline{x} \models \mathcal{G}$.

Definition 7. (dynamic range): *A* dynamic range *for vertex set V is a function $D : V \to 2^{\mathbb{N} \cup V}$ that is acyclic; i.e. there exists an ordering of V, $v_1, \ldots, v_k$ such that, $v_j \in x(v_i)$ implies $j > i$.*

A dynamic range D gives rise to $X_D = \{x \mid \forall v \in V, x(v) \in D(v)\}$, a set of dynamic assignments, which in turn induces a set of static assignments $\overline{X_D} = \{\overline{x} \mid x \in X_D\}$. We will therefore say that D is adequate for the E-graph $\mathcal{G}$ if the above assignment set $\overline{X_D}$ is adequate for $\mathcal{G}$.

One advantage of dynamic range allocation is that given an E-formula φ, and an adequate dynamic range D for φ, we can efficiently generate a Boolean formula that is satisfiable iff φ is, with little increase in the size of the formula.

W.l.o.g., assume that the variable ordering satisfying Definition 7 is $v_1, \ldots, v_n$, and therefore, fore every i, $D(v_i) \subseteq \mathbb{N} \cup \{v_i + 1, \ldots, v_n\}$. We encode the range $D(v_i)$ by a variable c_i whose domain is:

$$D(c_i) = (\mathbb{N} \cap D(v_i)) \cup \{-j \mid v_j \in D(v_i)\}$$

That is, we include in $D(c_i)$ all the integers which are in $D(v_i)$ and, for each $v_j \in D(v_i)$, we include $-j$ in $D(c_i)$.

Then, we can derive a formula χ which is satisfiability equivalent to φ but depends only on the variables $c_1, \ldots, c_n$. If these variables are then finite, i.e., all $D(v_i)$ are finite, then χ can be easily translated to a Boolean formula. We let

$$\chi_1 = \textbf{ let } (v_1 = \textbf{ if } (c_1 \geq 0) \textbf{ then } c_1 \textbf{ else } v_{-c_1}) \textbf{ in } \varphi$$

We then derive $\chi_2, \chi_3, \ldots, \chi_n$ successively, where for each $i > 1$:

$$\chi_i = \textbf{ let } (v_i = \textbf{ if } (c_i \geq 0) \textbf{ then } c_i \textbf{ else } v_{-c_i}) \textbf{ in } \chi_{i-1}$$

Finally, we let $\chi = \chi_n$. Note that since the dynamic range is acyclic, the resulting formula (with term-sharing) is also acyclic.

Claim. If D is adequate for $\mathcal{G}$ which is adequate for φ, the resulting formula χ is satisfiable iff φ is.

We now wish to construct a procedure that, given an E-graph $\mathcal{G}$, will construct a small adequate dynamic range for it. Notice that like in static ranges, the size of the state space when using a dynamic range D is $\prod_v |D(v)|$.

We will use the following notation for an E-graph $\mathcal{G}$ and a vertex $v \in V(\mathcal{G})$: $\Gamma_{EQ}(v) = \{u \mid (u,v) \in EQ(\mathcal{G})\}$, and $\Gamma_{DQ}(v) = \{u \mid (u,v) \in DQ(\mathcal{G})\}$.

We now define the E-graph $\mathcal{G}[v]$, which results from removing the vertex v from $\mathcal{G}$ and transforming v's equality and disequality constraints to constraints on its neighboring vertices:

1. The vertex set is $V(\mathcal{G}) \setminus \{v\}$.
2. Initially, $\mathcal{G}[v]$'s edges are all the edges of $\mathcal{G}$ which are not adjacent to v.
3. For $u_1 \neq u_2$ and $u_1, u_2 \in \Gamma_{EQ}(v)$, add an equality edge (u_1, u_2).
4. For $u_1 \neq u_2$, $u_1 \in \Gamma_{DQ}(v)$ and $u_2 \in \Gamma_{EQ}(v)$. add a disequality edge (u_1, u_2).

Example 2. In Figure 1, $\mathcal{G}_1 = \mathcal{G}_0[a]$, $\mathcal{G}_2 = \mathcal{G}_1[b]$, $\mathcal{G}_3 = \mathcal{G}_2[c]$, etc.

The following theorem is our building block for the construction of procedures that calculate an adequate dynamic range for a given E-graph $\mathcal{G}$ (see Appendix B.1 for its proof):

Theorem 1. *For E-graph $\mathcal{G}$ and $u \in V(\mathcal{G})$, if the dynamic range D is adequate for $\mathcal{G}[u]$, then D' is adequate for $\mathcal{G}$, where D' is defined as follows:*

1. $D'(v) = D(v)$ for every $v \neq u$.
2. If $\Gamma_{DQ}(u) = \emptyset$ and $\Gamma_{EQ}(u) \neq \emptyset$ then $D'(u) = \Gamma_{EQ}(u)$. Otherwise, $D'(u) = \Gamma_{EQ}(u) \cup \{\mathbf{unique}\}$.

Where **unique** $\in \mathbb{N}$ *and* **unique** $\notin \cup_v D(v)$.

Based on Theorem 1, the following procedure finds an adequate dynamic assignment set for a given E-graph $\mathcal{G}$:

1. Set *counter* $\leftarrow 0$.
2. Pick some vertex v of $V(\mathcal{G})$.
3. Set:
$$D(v) = \begin{cases} \{counter\} & \Gamma_{EQ}(v) = \emptyset \\ \Gamma_{EQ}(v) \cup \{counter\} & \Gamma_{EQ}(v) \neq \emptyset \text{ and } \Gamma_{DQ}(v) \neq \emptyset \\ \Gamma_{EQ}(v) & \Gamma_{EQ}(v) \neq \emptyset \text{ and } \Gamma_{DQ}(v) = \emptyset \end{cases}$$
4. Set $\mathcal{G} = \mathcal{G}[v]$.
5. Set *counter* $\leftarrow$ *counter* $+ 1$.
6. If $V(\mathcal{G}) \neq \emptyset$ return to Step 2.

Notice that *counter* is only used to generate unique numbers.

Example 3. Using this procedure we can generate an adequate dynamic range for the E-graph $\mathcal{G}_0$ of Figure 1:

1. Set $D(a) = \{c\}$ and calculate $\mathcal{G}_1 = \mathcal{G}_0[a]$.
2. Set $D(b) = \{0, d, e\}$, and $\mathcal{G}_2 = \mathcal{G}_1[b]$.
3. Set $D(c) = \{1, d, e\}$, and $\mathcal{G}_3 = \mathcal{G}_1[c]$.
4. Set $D(d) = \{2, e\}$, and $\mathcal{G}_4 = \mathcal{G}_3[d]$.
5. Set $D(e) = \{3\}$.

The resulting state space in our example was of size 18. If we would have taken a different order on the vertices when using our procedure: a, d, b, c, e, the resulting state space would be of size 12. In our implementation we use a simple greedy heuristic that in Step 2 chooses the vertex which will be allocated the smallest domain in Step 3. This heuristic generates a state space of 12 on the above example. In Section 9 we compare this procedure with the static range allocation procedure of [PRSS98].

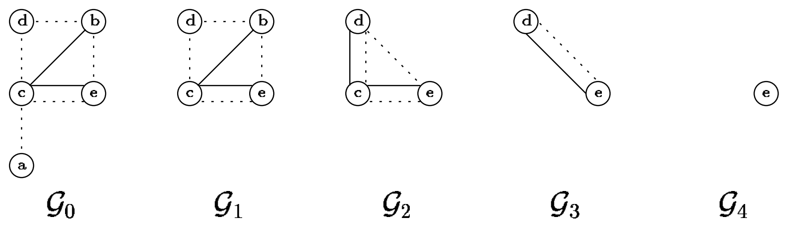

Fig. 1. Dynamic range allocation is based on an incremental process in which vertices are removed one by one, and their constraints are reallocated to their neighbors. The dashed lines in graphs $\mathcal{G}_0, \ldots, \mathcal{G}_4$ represent equality edges while solid lines represent disequality edges.

The sequence of E-graphs generated by our procedure is almost the same as that generated by the procedure suggested by [BV00], except they don't distinguish between equality and disequality edges, and therefore the vertices in our graphs should generally have a smaller degree. In their procedure, when a vertex is removed from the graph, the number of Boolean variables added to the formula is equal to its degree, and therefore the number of Boolean variables appearing in their resulting formula is $\sum_v degree(v)$, where $degree(v)$ is the degree of v in the graph at the time of its removal. In contrast, in our procedure, when a vertex v_i is removed, a variable c_i ranging over $degree(v) + 1$ values is added to the formula, and so we get $\sum_v \log(degree(v) + 1)$ Boolean variables. Therefore, we need much less Boolean variables than their method. However, we note that the Boolean formula they generate is very different than ours, and in spite of the increased number of boolean variables, may actually be easier to check.

6 One-Orientable Assignment Sets

In this section we present an alternative method for finding a partition set that is adequate for a given E-graph. This method generates a partition set of a different kind, with a more complex representation. Although the resulting Boolean formula has a relatively small number of Boolean variables in it, this method is not practical since this Boolean formula is much larger and more complex than the original one. This renders this method impractical, yet still interesting since we can show that on a large set of E-graphs the number of Boolean variables is slightly more than twice the minimal number needed.

Definition 8. (partition of a graph): *A graph G defines a partition:*

$$\alpha_G = \{W \subseteq V(G) \mid W \textit{ is a connected component of } G\}$$

Definition 9. (one-orientable): *A graph is* one-orientable *if its edges can be directed in such a way that the out-degree of every vertex* ≤ 1.

Definition 10. (one-orientable partition set): *The* one-orientable partition set *of an E-graph* $\mathcal{G}$ *is* $One(\mathcal{G}) = \{\alpha_H \mid$ *graph H is one-orientable and* $H \leq \mathcal{G}_= \}$

Proposition 1. *$One(\mathcal{G})$ is adequate for $\mathcal{G}$.*

Example 4. Figure 2 presents the graph $\mathcal{G}_=$ of some E-graph $\mathcal{G}$. The sub-graphs $1, \ldots, 5$ describe some of its one-orientable sub-graphs, and therefore their corresponding partitions are in $One(\mathcal{G})$; i.e., $\{\{a, b, c, d\}\} \in One(\mathcal{G})$ because of sub-graph 1, $\{\{a\}, \{b\}, \{c\}, \{d\}\}$ because of 2, and $\{\{a, b\}, \{c, d\}\}$ because of 5. Note that the only sub-graph of this graph that is not one-orientable is the graph itself.

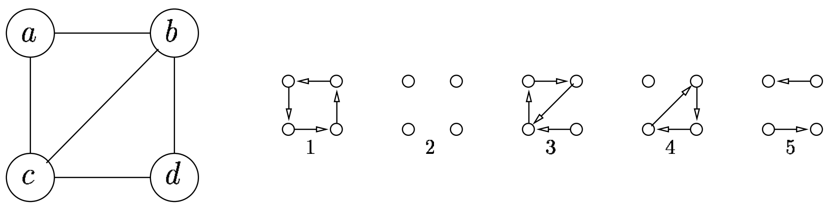

Fig. 2. The small graphs are one-orientable sub-graphs of the graph on the left. The direction on the edges was added for demonstrating the fact that the sub-graphs are one-orientable.

As a result, if we are given an E-formula φ together with an adequate E-graph $\mathcal{G}$ for it, we can check if φ is satisfiable by checking if there is some partition $\alpha \in One(\mathcal{G})$ such that $\varphi(\alpha) = 1$. This is implemented as follows:

Represent φ as a Boolean formula with atoms $(u = v)$. This can be done by flattening all the **ITE** terms in the formula. This flattening procedure increases the size of the formula only polynomially. Now construct the following formula C:

1. For each $v \in V(\mathcal{G})$, C has an input variable I_v ranging over $\{u \mid (u,v) \in EQ(\mathcal{G})\} \cup \{\star\}$.
2. For each $u, v \in V(\mathcal{G})$, C contains an internal variable $e_{(u,v)} := (I_u = v) \vee (I_v = u)$.
3. C contains a circuit for calculating the transitive closure of a graph with vertices $V(\mathcal{G})$. for every $u, v \in V(\mathcal{G})$, it has input variable $e_{(u,v)}$, and output variable $t_{(u,v)}$. There are known constructions of this circuit that are of polynomial size (e.g., using successive $\log(V(\mathcal{G}))$ boolean matrix multiplications).
4. Replace every atom $(u = v)$ in φ by $t_{(u,v)}$.

Proposition 2. *C is satisfiable iff φ is satisfiable.*

The general idea of the construction (see Appendix B.2 for the proof) is that the variables $e_{u,v}$ represent a one-orientable sub-graph of $\mathcal{G}_=$ that results from undirecting all the edges (u, I_u). Then $t_{u,v}$ represents the partition resulting from this sub-graph, that is used as input to φ.

The size of the resulting state space is $\prod_v |I_v| = \prod_{v \in V}(degree(v) + 1)$, where $degree(v)$ is the degree of v in $\mathcal{G}_=$.

7 Lower Bound On The Size of Partition Sets

In Section 6, we constructed an adequate partition set (the one-orientable partition set) for a given E-graph $\mathcal{G}$. This set was of size at most $\prod_{v \in V}(degree(v) + 1)$, where $degree(v)$ is the degree of v in $\mathcal{G}_=$. In this section we show every partition set that is adequate for $\mathcal{G}$ is at least of size:

$$\sqrt{\prod_{v \in V} (\frac{1}{2} degree'(v) + 1)}$$

where $degree'(v)$ is the degree of v in $\langle V(\mathcal{G}), EQ(\mathcal{G}) \cap DQ(\mathcal{G})\rangle$. Therefore on any E-graph where $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$, $One(\mathcal{G})$ is close to optimal in terms of the number of Boolean variables that is needed to represent such a state space. We get that any adequate assignment set will need at least $\frac{1}{2}\sum(\log(degree(v) + 2) - 1)$ Boolean variables, and that for representing $One(\mathcal{G})$ we need only slightly more than twice this number: $\sum \log(degree(v) + 1)$.

It is important to notice that a lower bound on the size of an adequate partition set for the E-graph $\mathcal{G}$ is in fact a lower bound on the size of the state space generated by any method that uses only the information in $\mathcal{G}$. This means that this lower bound applies to any method that examines only the set of atomic equalities (and their polarity) that appear in the E-formula. To break this lower bound barrier, a more careful analysis of the formula will be needed.

We start with the following definition:

Definition 11. (maximal satisfiable sub-graph): *An E-graph $\mathcal{H}$ is a* maximal satisfiable sub-graph *of $\mathcal{G}$ (denoted $\mathcal{H} \lll \mathcal{G}$), if it is a satisfiable sub-graph of $\mathcal{G}$, and there is no $\mathcal{H}_1$ that is a satisfiable sub-graph of $\mathcal{G}$ such that $\mathcal{H}$ is a proper subgraph of $\mathcal{H}_1$.*

Lemma 2. *If $\mathcal{H}_1 \lll \mathcal{G}$, $\mathcal{H}_2 \lll \mathcal{G}$ and $\mathcal{H}_1 \neq \mathcal{H}_2$ then there is no partition α such that $\alpha \models \mathcal{H}_1$ and $\alpha \models \mathcal{H}_2$.*

Proof. Assume to the contrary: $\alpha \models \mathcal{H}_1$ and $\alpha \models \mathcal{H}_2$. Define E-graph $\mathcal{H} = \langle V(\mathcal{G}), EQ(\mathcal{H}_1) \cup EQ(\mathcal{H}_2), DQ(\mathcal{H}_1) \cup DQ(\mathcal{H}_2)\rangle$. Clearly, $\alpha \models \mathcal{H}$. Also, $\mathcal{H} \leq G$, and thereby $\mathcal{H} \lll \mathcal{G}$. Since $\mathcal{H}_1, \mathcal{H}_2 \leq \mathcal{H}$, $\mathcal{H} = \mathcal{H}_1 = \mathcal{H}_2$. □

Lemma 1 directly implies:

Corollary 1. *If partition set R is adequate for the E-graph $\mathcal{G}$, then $|R| \geq |\{\mathcal{H} \mid \mathcal{H} \lll \mathcal{G}\}|$.*

Consider an E-graph $\mathcal{G}$ where $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$. For this type of E-graphs we can bound their set of maximal satisfiable sub-graphs. We say a partition α is connected in a graph G if every set $\alpha_i \in \alpha$ is connected in G restricted to the vertices of α_i. We denote by $CP(G)$ the set of connected partitions of G.

Example 5. In the graph of Figure 2, $\{\{a,b\},\{c\},\{d\}\}$ and $\{\{a,c,d\},\{b\}\}$ are connected partitions, yet $\{\{c,b\},\{a,d\}\}$ is not.

Proposition 3. *If $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$, then $|CP(\mathcal{G}_=)| \leq |\{\mathcal{H} \mid \mathcal{H} \lll \mathcal{G}\}|$*

Proof. We define the following mapping:

$$\psi : \{\mathcal{H} \mid \mathcal{H} \lll \mathcal{G}\} \rightarrow CP(\mathcal{G}_=)$$

Where $\psi(\mathcal{H}) = \alpha_{\mathcal{H}_=}$ (see Definition 8).

Clearly, for every $\mathcal{H}$, $\psi(\mathcal{H}) \in CP(\mathcal{G}_=)$ since for every $\alpha_i \in \alpha_{\mathcal{H}_=}$, α_i is a connected component of $\mathcal{H}_= \leq \mathcal{G}_=$, and so α_i is connected in $\mathcal{G}_=$.

To prove the proposition we show that ψ is onto. For $\alpha \in CP(\mathcal{G}_=)$, define $\mathcal{H} \leq \mathcal{G}$ to be:

1. If $u \sim_\alpha v$ and $(u,v) \in EQ(\mathcal{G})$ then $(u,v) \in EQ(\mathcal{H})$.
2. If $u \not\sim_\alpha v$ and $(u,v) \in DQ(\mathcal{G})$ then $(u,v) \in DQ(\mathcal{H})$.

We claim that $\mathcal{H}$ is maximal and that $\psi(\mathcal{H}) = \alpha$ (clearly $\mathcal{H}$ is satisfiable). To show that $\psi(\mathcal{H}) = \alpha$, we need to show that the connected components of $\mathcal{H}_=$ coincide with the sets of α:

1. Each connected component of $\mathcal{H}_=$ is a subset of some $\alpha_i \in \alpha$, since every edge $(u,v) \in \mathcal{H}_=$ satisfies $u \sim_\alpha v$.
2. Each α_i is a subset of some connected component of $\mathcal{H}_=$, since all edges of $\mathcal{G}_=$ between vertices of α_i are in $\mathcal{H}_=$, and α_i is connected in $\mathcal{G}_=$.

We now show that $\mathcal{H}$ is maximal. We have to handle two cases:

1. $(u,v) \in EQ(\mathcal{G})$, $(u,v) \notin EQ(\mathcal{H})$. This means that $u \not\sim_\alpha v$, and therefore, since $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$, then $(u,v) \in DQ(\mathcal{H})$, but then if we add (u,v) to $EQ(\mathcal{H})$ it makes $\mathcal{H}$ unsatisfiable.
2. $(u,v) \in DQ(\mathcal{G})$, $(u,v) \notin DQ(\mathcal{H})$. This means that $u \sim_\alpha v$, implying there exists $\alpha_i \in \alpha$ such that $u,v \in \alpha_i$. α_i is connected in $\mathcal{G}_=$, and therefore in $\mathcal{H}_=$. Now, using Lemma 2, we see that adding (u,v) to $DQ(\mathcal{H})$ will make $\mathcal{H}$ unsatisfiable.

□

Corollary 2. *For an E-graph $\mathcal{G}$ such that $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$, every partition set R that is adequate for $\mathcal{G}$ satisfies $|R| \geq |CP(\mathcal{G}_=)|$.*

Theorem 2. *For every graph G, $|CP(G)| \geq \sqrt{\prod_{v \in V(G)}(\frac{1}{2}degree(v) + 1)}$*

We believe this theorem (and its proof) to be of independent interest, as it addresses a natural (yet nontrivial) combinatorial counting question: the number of connected partitions of a given graph.

In fact, using a construction similar to that of one-orientable sets, it is easy to show that for every graph G, $|CP(G)| \leq \prod_{v \in V(G)}(degree(v) + 1)$. Together with Theorem 2 we have a good approximation of $|CP(G)|$.

Combining Theorem 2 with Corollary 2 we get:

Corollary 3. *For an E-graph $\mathcal{G}$ such that $EQ(\mathcal{G}) \subseteq DQ(\mathcal{G})$, every partition set R that is adequate for $\mathcal{G}$ satisfies*

$$|R| \geq \sqrt{\prod_{v \in V(G)} (\frac{1}{2} degree(v) + 1)}$$

Where $degree(v)$ is the degree of v in $\mathcal{G}_=$.

Claim. For any two E-graphs $\mathcal{G}_1 \leq \mathcal{G}_2$, if partition set R is adequate for $\mathcal{G}_2$ then it is also adequate for $\mathcal{G}_1$.

Using this claim with Corollary 3:

Corollary 4. *For an E-graph $\mathcal{G}$, every partition set R that is adequate for $\mathcal{G}$ satisfies*

$$|R| \geq \sqrt{\prod_{v \in V(G)} (\frac{1}{2} degree'(v) + 1)}$$

Where $degree'(v)$ is the degree of v in $\langle V(\mathcal{G}), EQ(\mathcal{G}) \cap DQ(\mathcal{G})\rangle$.

8 Proof of Theorem 2

In order to prove Theorem 2 we need the following lemmas:

Lemma 3. *For a graph G and two non-intersecting sets $S, T \subseteq V(G)$ such that $S \cup T = V(G)$*

$$|CP(G)| \geq \prod_{v \in S} (degree_T(v) + 1)$$

Where $degree_T(v) = |\{u \in T \mid (u, v) \in E\}|$.

Proof. Consider the following procedure that constructs a partition α:

1. First start with $\alpha = \{\{v\} \mid v \in T\}$ (Note that α is not a partition yet).
2. For all vertices $v \in S$ do one of the two:
   (a) Take one vertex $u \in T$ such that $(u, v) \in E$, and add v to u's set in α.
   (b) Add the set $\{v\}$ to α.

Clearly, the final α is a partition, and is connected, since we add a vertex v to a set only if there is a vertex u in that set such that $(u, v) \in E$.

For each vertex $v \in S$, we have to choose between $degree_T(v) + 1$ choices in the procedure. So, if we show that different choices for the vertices always lead to different partitions, then we constructed a set of $\prod_{v \in S}(degree_T(v) + 1)$ different connected partitions of G.

First note that the partition α constructed has the following property: Every $\alpha_i \in \alpha$ satisfies one of:

1. $\alpha_i \cap T = \emptyset$ and $|\alpha_i| = 1$
2. $|\alpha_i \cap T| = 1$.

From this we see that two different runs lead to different partitions. If a vertex v chooses to join some $u \in T$, it will not be in the same set with any other $u' \in T$. So different choices here lead to different partitions. Also, if a vertex chooses not to join any $u \in T$, then it will be a singular set in α, and this cannot happen if it chooses to join some vertex of T. □

Lemma 4. *For a graph G, there are non-intersecting sets S_0, S_1, such that $S_0 \cup S_1 = V(G)$, and for every $v \in S_i$, $degree_{S_{(1-i)}}(v) \geq \frac{1}{2}degree(v)$.*

Proof. Start with two arbitrary sets S_0 and S_1. While possible, pick some $v \in S_i$ such that $degree_{S_{(1-i)}}(v) < \frac{1}{2}degree(v)$. Take v from S_i and put it in S_{1-i}. If there are no such vertices left, then S_0 and S_1 satisfy the lemma.

We claim that the procedure will end. This is because each such move increases the following function:

$$Cut(S_0, S_1) = |\{(u,v) \in E(G) \mid u \in S_0, v \in S_1\}|$$

This is because:

$$Cut(S_0 \cup \{v\}, S_1 \setminus \{v\}) = Cut(S_0, S_1) + degree_{S_1}(v) - degree_{S_0}(v)$$

By the way we pick our vertices, we see that this function increases for each move we make. Since $Cut(S_0, S_1) < |E(G)|$, the procedure will halt after at most $|E(G)|$ moves. □

We comment that the proof of Lemma 4 is in fact a schoolbook construction of a maximum cut in a graph.

Proof. (of Theorem 2): Using Lemma 4 we get two sets non-intersecting sets S_0 and S_1. Now we use Lemma 3, setting $S = S_0$, and $T = S_1$. We get:

$$|CP(G)| \geq \prod_{v \in S_0} (degree_{S_1}(v) + 1) \geq \prod_{v \in S_0} (\frac{1}{2}degree(v) + 1)$$

We use Lemma 3 again, setting $S = S_1$ and $T = S_0$. We get:

$$|CP(G)| \geq \prod_{v \in S_1} (\frac{1}{2}degree(v) + 1)$$

Combining:

$$|CP(G)|^2 \geq \prod_{v \in S_0} (\frac{1}{2}degree(v) + 1) \cdot \prod_{v \in S_1} (\frac{1}{2}degree(v) + 1) = \prod_{v \in V} (\frac{1}{2}degree(v) + 1)$$

Proving the theorem. □

9 Static vs. Dynamic Ranges

It is difficult to estimate the advantages of using dynamic ranges. We cannot show any relation to the minimal partition set possible (as we did for one-orientable ranges), but since dynamic ranges generalize static ranges, they are at least as good. There are E-graphs where dynamic ranges seem to not give any advantage over static ones, but we will show an example where using a dynamic range gives far better results.

Consider the following E-graph $\mathcal{G}$:

1. $V = \{v_i \mid i \in \{1, \ldots, n\}\} \cup \left\{u_i^j \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, d\}\right\}$
2. $DQ = \{(v_i, v_j) \mid i, j \in \{1, \ldots, n\}\}$

3. $EQ = \{(v_i, v_j) \mid i, j \in \{1, \ldots, n\}\} \cup \left\{(u_i^j, v_i) \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, d\}\right\}$

In other words, $\mathcal{G}$ is constructed of a clique of equality and disequality edges: $v_1, \ldots, v_n$, where each v_i has d equality leaves connected to it $u_i^1, \ldots, u_i^d$.

Assume D is some static range of minimal size that is adequate for $\mathcal{G}$.

Claim. for all i and j, $D(v_i) \subseteq D(u_i^j)$

Proof. Assume there is some $l \in D(v_i) \setminus D(u_i^j)$. Define range D' to be the same as D, except $D'(v_i) = D(v_i) \setminus \{l\}$. We claim that D' is adequate for $\mathcal{G}$ contradicting the fact that D is minimal.

For a satisfiable $\mathcal{H} \leq \mathcal{G}$, there exists an assignment $a \in D$ s.t. $a \models \mathcal{H}$. We find $a' \in D'$ s.t. $a' \models \mathcal{H}$. We split to two cases:

1. If $(v_i, u_i^j) \in EQ(\mathcal{H})$ then $a(v_i) = a(u_i^j)$, and therefore, since $a(u_i^j) \neq l$, $a(v_i) \neq l$, and then we define $a' = a$, since $a \in D'$.
2. If $(v_i, u_i^j) \notin EQ(\mathcal{H})$ then adding this edge to $\mathcal{H}$ will leave it satisfiable (this is because of $\mathcal{G}$'s structure). Therefore, there is some $a \in D$ that satisfies this new $\mathcal{H}$. This a will also satisfy the original $\mathcal{H}$. Using the same argument as in (1) we see that $a \in D'$.

□

We therefore have that for all i and j, $|D(u_i^j)| \geq |D(v_i)|$, and therefore:

$$\prod_{i,j} |D(u_j^i)| \geq \left(\prod_i |D(v_i)|\right)^d$$

Considering the restriction of $\mathcal{G}$ to $\{v_1, \ldots, v_n\}$, and using corollary 4, we have:

$$\prod_i |D(v_i)| \geq \left(\frac{n}{2}\right)^{\frac{n}{2}}$$

Therefore:

$$\prod_{v \in V(\mathcal{G})} |D(v)| = \prod_i |D(v_i)| \cdot \prod_{i,j} |D(u_i^j)| \geq \left(\prod_i |D(v_i)|\right)^{d+1} \geq \left(\frac{n}{2}\right)^{\frac{(d+1)\cdot n}{2}}$$

If we take $\log_2$ of this number (to get the number of Boolean variables needed), we have $\frac{1}{2} \cdot (d+1) \cdot n \cdot (\log(n) - 1)$. In comparison, the following dynamic range is adequate for $\mathcal{G}$:

1. $v_i = \{1, \ldots, i\}$
2. $u_i^j = \{v_i\}$

Which is of size $n!$, i.e., $O(n \log(n))$ Boolean variables, which does not depend on d at all.

Note that the lower bound on the size of any static range is true for any E-graph containing this E-graph as a sub-graph.

10 Experimental Results

To test our dynamic range allocation procedure and compare our results to the static range allocation procedure of [PRSS98], we generated many random E-graphs. For each E-graph $\mathcal{G}$ we calculated the size of the resulting state space generated $|S|$, and then calculated $|S|^{\frac{1}{|V(\mathcal{G})|}}$ to give the average size of the range for each variable.

We believe dynamic range allocation performs especially well on E-graphs that can be divided into components with a small number of equality edges between them. We performed the following set of tests:

Set $V(\mathcal{G}) = \{1, \ldots, 100\}$. For each $p \in \{.2, .25, \ldots, 1\}$ and $q \in \{.01, .02, \ldots, .2\}$ we generate 10 random E-graphs, by letting each edge (i, j) be chosen with probability p if $i \ mod\ 4 = j \ mod\ 4$, and with probability q otherwise. This way we get 4 components with edge probability p for edges internal to these components, and probability q for edges between the components. We also ran a simpler set of tests, where for each $p \in \{.02, .04, \ldots, 1\}$ we generated 10 random E-graphs on 100 vertices, where each edge is taken with probability p.

In Figure 3 we see the summary of the results, where for each test we averaged the results ($|S|^{\frac{1}{100}}$) of dynamic and static ranges over the 10 graphs, and give the ratio between them. We can see that for all cases the ratio is greater than 1, meaning the dynamic range allocation is at least as good (on the 10 graph average), and that on sparse graphs (either q is small or p is small) we get an improvement of approximately 2, which means a decrease of 2^{100} in the state space size. We have

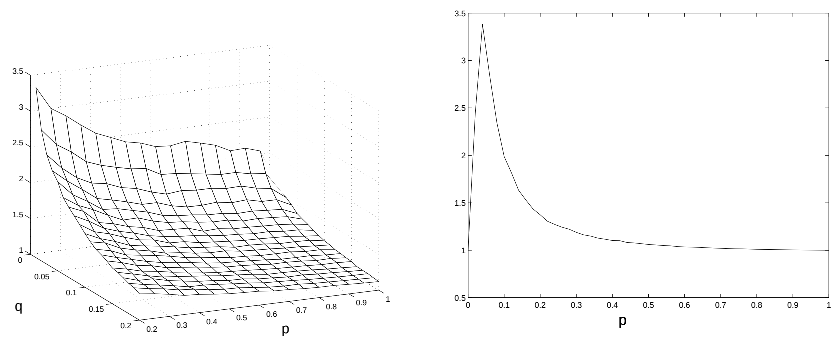

Fig. 3. The ratio between the average domain sizes (of each variable) allocated by the static and dynamic range allocation methods, as computed on graphs with 100 vertices and 4 components on the left graph, where p and q are the probabilities of adding edges in and between the components, respectively. The right graph summarizes the results for graphs with 100 vertices and edge probability p.

also implemented dynamic range allocation as a part of a procedure for checking uninterpreted functions [RS01], and achieved a factor 2 improvement in the range size, similar to the results on random graphs. However, we have not found a case where this improvement led to a significant change in running times. This is especially because our examples are of two types: the run either completes in less than 1 second, or it never completes.

References

[BD94] J.R. Burch and D.L. Dill, "Automatic Verification of Microprocessor Control", In *Computer-Aided Verification CAV '94* .

[BDL96] Clark W. Barrett, David L. Dill and Jeremy R. Levitt, "Validity Checking for Combinations of Theories with Equality", In *Formal Methods in Computer Aided Design FMCAD '96* .

[BV00] R. E. Bryant and M. N. Velev, "Boolean satisfiability with transitivity constraints", In *Computer-Aided Verification CAV 2000* .

[GSZAS98] A. Goel, K. Sajid, H. Zhou, A. Aziz and V. Singhal, "BDD Based Procedures for a Theory of Equality with Uninterpreted Functions", In *Computer-Aided Verification CAV '98* .

[PRSS98] A. Pnueli, Y. Rodeh, M. Seigel and O. Shtrichman, "Deciding Equality Formulas by Small Domain Instantiations", In *Computer-Aided Verification CAV '99* .

[PSS98] A. Pnueli, M. Siegel and O. Shtrichman, "Translation Validation for Synchronous Languages", In *International Colloquium on Automata, Languages and Programming ICALP '98* .

[RS01] Y. Rodeh and O. Shtrichman, "Finite Instantiations in Equivalence Logic with Uninterpreted Functions", In *Computer-Aided Verification CAV '01* ..

A Proofs

A.1 Dynamic Range Allocation

Proof. (of Theorem 1): For a satisfiable $\mathcal{H} \leq \mathcal{G}$, we find $x \in D'$ such that $x \models \mathcal{H}$. We first notice that since $\mathcal{H}$ is satisfiable, $\mathcal{H}[v]$ is also satisfiable. Also, $\mathcal{H}[v] \leq \mathcal{G}[v]$. Therefore there is some $x \in D$ such that $x \models \mathcal{H}[v]$. We extend x to v, and therefore only need to show that x satisfies all edges of $\mathcal{H}$ involving v since all other edges are clearly satisfied by x.

1. If there is no $(u, v) \in EQ(\mathcal{H})$, then set $x(v) =$ **unique**. Since all edges involving v are disequality edges, we have that $x \models \mathcal{H}$.
2. If there is some $(u, v) \in EQ(\mathcal{H})$, then set $x(v) = u$:
   (a) For a vertex w such that $(v, w) \in EQ(\mathcal{H})$, if $w = u$ then clearly, $\overline{x}(w) = \overline{x}(v)$. Otherwise, there is an equality edges $(u, w) \in EQ(\mathcal{H}[v])$, and therefore $\overline{x}(w) = \overline{x}(u)$, and then $\overline{x}(w) = \overline{x}(v)$.
   (b) For a vertex w such that $(v, w) \in DQ(\mathcal{H})$, since $\mathcal{H}$ is satisfiable, $w \neq u$. Also, there is a disequality edge $(w, u) \in DQ(\mathcal{H}[v])$, and then $\overline{x}(v) = \overline{x}(u) \neq \overline{x}(w)$.

□

A.2 One-Orientable Assignment Sets

To prove proposition 1, we need the following definitions:

Definition 12. (forest): *A* forest *is an acyclic undirected graph.*

Definition 13. (spanning forest): *A* spanning forest *for a graph G, is a subgraph F of G, such that F is a forest, and the connected components of F and G are the same.*

Claim. Every graph has a spanning forest.

Definition 14. (forest partition set): *The* forest partition set *of the E-graph $\mathcal{G}$ is:*

$$F(\mathcal{G}) = \{\alpha_F \mid F \textit{ is a forest and } F \leq \mathcal{G}_=\}$$

Proposition 4. *$F(\mathcal{G})$ is adequate for $\mathcal{G}$.*

Proof. Given a satisfiable $\mathcal{H} \leq \mathcal{G}$, take the forest F to be a spanning forest of the graph $\mathcal{H}_=$. Clearly, $E(F) \subseteq EQ(\mathcal{H}) \subseteq EQ(\mathcal{G})$. So, $\alpha_F \in R$.

We claim that $\alpha_F \models \mathcal{H}$:

1. If $(u, v) \in EQ(\mathcal{H})$, then u and v are in the same connected component of $\mathcal{H}_=$, and therefore of F. This means that $u \sim_{\alpha_F} v$.
2. If $(u, v) \in DQ(\mathcal{H})$ then u and v are not in the same connected component of $\mathcal{H}_=$ by Lemma 2. This means that they are in different components of F, and therefore $u \not\sim_{\alpha_F} v$.

□

Proof. (of Proposition 1): Since every forest is one-orientable (by rooting each of the trees in the forest, and directing all edges towards the root), we get that $F(\mathcal{G}) \subseteq One(\mathcal{G})$ and therefore $One(\mathcal{G})$ is adequate for $\mathcal{G}$. □

Proof. (of Proposition 2): We first show that the graphs represented by the variables $e_{(u,v)}$ are all the one-orientable sub-graphs of $\mathcal{G}_=$ possible.

Take some one-orientable $H \leq \mathcal{G}_=$. Denote by D the directed graph resulting from directing H's edges in such a way that every vertex has out-degree at most 1 in D. We define assignment a to the input variables of C. For each v:

- If there is exactly one u such that $(v, u) \in D$, set $a(I_v) = u$.
- Otherwise, there are no outgoing edges from v in D. Set $a(I_v) = \star$.

We get that $a(e_{(u,v)}) = 1$ iff $(u, v) \in H$. Therefore, $a(t_{(u,v)}) = 1$ iff u and v are connected in H. In other words $a(t_{(u,v)}) = 1$ iff $u \sim_{\alpha_H} v$.

So, for every $\alpha \in One(\mathcal{G})$, there is some assignment a to C, such that $a(C) = \varphi(\alpha)$, and since $One(\mathcal{G})$ is adequate for φ, we conclude. □