MDG Algorithms (I)

F. Corella† Z. Zhou‡ X. Song‡ M. Langevin§ E. Cerny‡

† Hewlett-Packard Company, USA, corella@hprpcd.rose.ibm.com
‡ Dépt. d'I. R. O., Univ. de Montréal, Canada, {zhouz,song,cerny}@iro.umontreal.ca
§ GMD–SET, Germany, langevin@borneo.gmd.de

Abstract

This document provides detailed description of basic algorithms for MDG (Multi-way Decision Graph) operators, including disjunction, relational product and pruning-by-subsumption. There is no difficulty in verifying the soundness of the algorithms, except perhaps for the pruning algorithm, for which a proof of soundness is given in Section 5. The document is not self-containd. For basic concepts and informal descriptions of the algorithms, please refer to [1].

1 Technical preliminaries

If an MDG G represents a formula P, we shall use P to refer both to the formula and to the graph when there is no risk of confusion. We shall also sometimes identify nodes and edges with their labels.

We shall refer to the edges that issue from the root of an MDG as the *root edges*, and to the subgraphs that they lead to as the *immediate subgraphs*.

A *full node* is an internal node N labeled by a variable or cross-term of concrete sort α, such that every individual constant in the enumeration of α appears as a label of an edge issuing from N. (A redundant node is a special case of a full node, where all the edges issuing from the node lead to the same subgraph.)

A *segment* σ of an MDG G is a maximal sequence of consecutive node-edge pairs where all the node labels have same top symbol; we shall refer to this common top symbol as the *top symbol of the segment*. A segment σ can thus consist of a single node N, labeled by a variable, and an edge issuing from N, or of a sequence of node-edge pairs where the nodes are labeled by cross-terms with the same cross-operator. If $[(A_i, B_i)]_{1\leq i\leq n}$ is the sequence of the node-edge pairs in σ, we shall identify σ with the expression $\bigwedge_{1\leq i\leq n}(A_i = B_i)$. Note that σ could also be identified with the graph obtained by adding a leaf node labeled T at the end of the sequence of consecutive nodes and edges. Such a graph G_σ is a well-formed MDG; we shall say that σ is of type $X \leadsto Y$ to mean that G_σ is of type $X \leadsto Y$.

We call *root segments* those that begin at the root.

A *substitution* θ with a set of variables Y as its domain is, as usual, a function that maps every $y \in Y$ to a term of the same sort as y. We shall write $A \cdot \theta$ for the result of *applying* the substitution θ to A, where A can be a term, a formula, a variable, a set of variables, or so on; for example, if x is a variable *not* in the domain of θ, $\theta(x)$ is undefined, but $x \cdot \theta$ is x itself.

A *renaming substitution* is one that maps variables to variables in a one-to-one manner. We say that a renaming substitution η *preserves an ordering relation* $\prec$ over a set of symbols S iff the function $f : x \in S \mapsto x \cdot \eta$ is an ordering isomorphism from $(S, \prec)$ onto $(f(S), \prec)$. We shall use this notion in the case where $\prec$ is the custom symbol ordering, S is a set of variables and cross-operators, and the domain and range of η consist of variables that participate in the symbol ordering; in that case η preserves the symbol ordering iff

$$(\forall x, x' \in S)(x \prec x' \Rightarrow x \cdot \eta \prec x' \cdot \eta). \tag{1}$$

Besides renaming substitutions we shall only consider substitutions θ that map abstract variables to concretely-reduced terms. That is, every $y \in dom(\theta)$ is abstract, and every $\theta(y)$ is a concretely-reduced term. Such a substitution can be identified with a linear MDG G_θ

with node-edge pairs $[(y_i, \theta(y_i)]_{1 \leq i \leq n}$, where y_i, $1 \leq i \leq n$, are the variables of Y in custom order, and with leaf node T. It can also be identified with the formula $\bigwedge_{1 \leq i \leq n}(y_i = \theta(y_i))$. The MDG G_θ is not always well-formed. For G_θ to be well-formed every variable x that occurs in some $\theta(y)$, $y \in Y$, must come before the corresponding y in custom symbol order, and the set X of such x must be disjoint from Y. When this is the case, we say that θ itself is well-formed; and we say that θ is of type $X' \rightarrow Y'$ iff G_θ is of type $X' \rightarrow Y'$: this means that $X \subseteq X'$, $Y \subseteq Y'$, and $Y_{\text{abs}} = (Y')_{\text{abs}}$.

The MDG algorithms use a common *graph reduction table and a common term reduction table,* as well as separate *result tables*, all implemented as hash tables. Each term has a unique identifier (ID). The ID of an individual constant or an abstract variable is the constant or the variable itself. For a compound term, its ID corresponds to an entry in the term reduction table, which maps the functor of the term, the sequence of identifiers of sub-terms to the term. In the following, when we say label ID, we mean the term identifier of the label. Each MDG has a unique identifier (ID). The graph reduction table has an entry for each MDG that maps the label ID of the root, the sequence of label IDs of the root edges, and the sequence of identifiers of the immediate subgraphs to the root node of the graph. In the descriptions of the algorithms, when we say that we construct a graph with given root label, root edge labels, and immediate subgraphs, it should always be understood that the reduction table is looked up; if an entry is found, the graph is returned; otherwise, a new graph is assembled using a new ID, and the reduction table is updated. (The same thing is true for the construction of a term.) On the other hand, using result tables, the MDG algorithms exploit the structure sharing of MDGs in the same way as ordinary ROBDD algorithms.

We define a total ordering on the set of MDGs other than T and F, the *graph ordering*, as follows: G comes before G' iff the root label of G comes before the root label of G' in node-label ordering, or G and G' have same root label and the ID of G is less than the ID of G' (IDs are natural numbers). When sorting lists of MDGs, the ordering used will always be graph order.

2 Disjunction

$R = \mathsf{Disj}(\{P_i\}_{1\le i\le n})$

- Argument: A set $S = \{P_i\}_{1\le i\le n}$ of MDGs P_i. Each P_i other than T or F is an MDG of type $U_i \to Y$. (Note that all the P_i have the same set $Y_{\mathbf{abs}}$ of primary abstract variables.)

- Result: an MDG R such that

$$\models R \Leftrightarrow (\bigvee_{1\le i\le n} P_i) \tag{2}$$

  R can be F, T, or an MDG of type $V \to Y$, where V is the union of the sets of variables U_i.

If any of the P_i is T, return $R = \mathsf{T}$. If $n = 0$ or all the P_i are F, return $R = \mathsf{F}$. Otherwise sort the P_i other than F and remove duplicates, thus obtaining a list $[P'_i]_{1\le i\le n'}$.

Check if the disjunction of the P'_i, $1 \le i \le n'$, has already been computed. The result table for disjunction maps a list of graph IDs (extracted from the list $[P'_i]_{1\le i\le n'}$) to a graph.

If the disjunction has not been computed previously, let A be the smallest of the root labels of the P'_i, $1 \le i \le n'$, in node-label order. Since the P'_i are sorted in graph order, those whose root label is A appear at the front of the list. If A is an abstract variable, then it is the root label of *all* the P'_i.

For each i, $1 \le i \le n'$, let $L_i = [(B_{ij}, Q_{ij})]_{1\le j\le m_i}$ be a list of pairs defined as follows:

1. If the root of P'_i is A, then the B_{ij}, $1 \le j \le m_i$ are the root edges of P'_i (in standard term order, as they appear in P'_i), and the Q_{ij}, $1 \le j \le m_i$ are the immediate subgraphs that those edges lead to.

2. If the root of P'_i comes after A in node-label order, it must be the case that A is a concrete variable or a cross-term. Let α be the sort of A. Then the B_{ij}, $1 \le j \le m_j$ are the individual constants in the enumeration of α, in standard term order, and $Q_{ij} = P'_i$ for all j.

Merge the L_i, $1 \le i \le n'$, to obtain a list of pairs $[(B'_k, S'_k)]_{1\le k\le l}$, where the B'_k are the B_{ij}, $1 \le i \le n'$, $1 \le j \le m_i$, arranged in standard term order without duplicates, and S'_k is the set of P'_{ij} for which the corresponding B_{ij} is B'_k. For each k, $1 \le k \le l$, recursively compute $Q'_k = \mathsf{Disj}(S'_k)$. If A is of concrete sort α, if all the Q'_k are the same, and if the B'_k comprise all the individual constants in the enumeration of α, then return $R = Q'_1 = \ldots = Q'_l$. Otherwise construct an MDG R with root A, root edges $[B'_k]_{1\le k\le l}$ and immediate subgraphs $[Q'_k]_{1\le k\le l}$, update the result table and return R as the result.

3 Relational product

$R = \mathsf{RelP}(\{P_i\}_{1\le i\le n}, V, \eta)$

- Arguments: A (possibly empty) set $S = \{P_i\}_{1\le i\le n}$, $n \ge 0$, of MDGs P_i, each being either T, or F, or of type $X_i \to Y_i$, a set of variables V, and a renaming substitution η. Let $X = \bigcup_{1\le i\le n} X_i$ and $Y = \bigcup_{1\le i\le n} Y_i$. Conditions:

  1. The $(Y_i)_{\mathrm{abs}}$ are pairwise disjoint.
  2. $V \subseteq Y$.
  3. The domain of η is a subset of $Y \setminus V$, and η preserves the custom ordering over the union of $(Y \setminus V) \cup (X \setminus Y)$ and the set of cross-operators.

- Result: an MDG R that can be F, T, or an MDG of type $(X \setminus Y) \to ((Y \setminus V) \cdot \eta)$ such that

$$\models R \cdot \eta^{-1} \Leftrightarrow (\exists V)(\bigwedge_{1\le i\le n} P_i). \tag{3}$$

If any of the P_i is F, return $R = \mathsf{F}$. If $n = 0$ or all the P_i are T, return $R = \mathsf{T}$. Otherwise sort the P_i other than T into a list L, then compute and return

$$R = \mathsf{RelP'}(L, V, \eta, \emptyset). \tag{4}$$

Auxiliary Recursive Algorithm RelP'

$R = \mathsf{RelP'}([P_i]_{1\le i\le n}, V, \eta, \theta)$.

- Arguments: A list $L = [P_i]_{1\le i\le n}$, $n \ge 0$, of MDGs $P_i \ne \mathsf{T}, \mathsf{F}$ of type $X_i \to Y_i$, a set of variables V, a renaming substitution η, and a well-formed substitution θ of type $X' \to Y'$. Let $X = \bigcup_{1\le i\le n} X_i$, $Y = \bigcup_{1\le i\le n} Y_i$, $X'' = X \cup X'$, and $Y'' = Y \cup Y'$. Conditions:

  1. The $(Y_i)_{\mathrm{abs}}$ are pairwise disjoint.
  2. Every variable in Y comes after every variable in Y' in custom symbol order.
  3. $V \subseteq Y''$.
  4. The domain of η is a subset of $Y'' \setminus V$, and η preserves the custom symbol ordering over the union of $(Y'' \setminus V) \cup (X'' \setminus Y'')$ and the set of cross-operators.

- Result: an MDG R that can be F, T, or an MDG of type $(X'' \setminus Y'') \to ((Y'' \setminus V) \cdot \eta)$ such that

$$\models \theta \Rightarrow (R \cdot \eta^{-1} \Leftrightarrow (\exists V)(\bigwedge_{1 \le i \le n} P_i)). \tag{5}$$

If $n = 0$, return $R = \mathsf{T}$. Otherwise, check the result table to see if the result has already been computed. The result table for relational product maps a list of graph IDs (extracted from L), V, η, and θ, to a graph. If no entry is found, proceed as follows.

Let m be the number of graphs P_i at the front of the list L that have same top symbol. (The top symbol of an MDG is the top symbol of the label of its root; we shall refer to the top symbol of the MDGs at the front of the list as the *front top symbol*.) Consider the following cases:

1. The front top symbol is a variable y of concrete type α. Then for $1 \le i \le m$, P_i has y as root label, and individual constants in the enumeration of α as labels of the root edges. Let $[b_j]_{1 \le j \le l}$ be the list of constants that appear as root edge labels in all the P_i, arranged in standard order. If the list is empty ($l = 0$), return $R = \mathsf{F}$. Otherwise, for $1 \le j \le l$ and $1 \le i \le m$, let Q_{ij} be the immediate subgraph of P_i corresponding to the edge labeled b_j.

   For each j, construct a (possibly empty) sorted list L'_j, starting with the (possibly empty) tail $[P_k]_{m < k \le n}$ of L and inserting in turn each $Q_{ij} \ne \mathsf{T}$ (if any) where it belongs. Compute the restriction θ'_j of θ to the set of variables that have secondary occurrences in the graphs that comprise L'_j. And recursively compute:

$$Q'_j = \mathsf{RelP'}(L'_j, V, \eta, \theta'_j). \tag{6}$$

   Then compute the result R as follows:

   (a) If $Q'_j = \mathsf{F}$ for $1 \le j \le l$, let $R = \mathsf{F}$.

   (b) Otherwise, if $y \notin V$, then:

      i. If the Q'_j are all the same MDG for $1 \le j \le l$, and if the b_j comprise all the individual constants in the enumeration of α, let R be that MDG.

      ii. Otherwise let R be the MDG with root labeled $y \cdot \eta$ and one root edge labeled b_j leading to Q'_j for each j such that $Q'_j \ne \mathsf{F}$.

   (c) Otherwise, use algorithm Disj to compute the disjunction of the $Q'_j \ne \mathsf{F}$, and let the result be R.

2. The front top symbol is an abstract variable y. Then, since the $(Y_i)_{\mathsf{abs}}$ are pairwise disjoint, it must be that $m = 1$. P_1 has root label y, and root edges labeled by terms B_j, $1 \leq j \leq l$. Compute $B_j \cdot \theta$ for each j. Form the list $[B_j \cdot \theta]_{1 \leq j \leq l}$, then sort it and remove duplicates to obtain a list $[B'_k]_{1 \leq k \leq h}$ $(h \leq l)$.

   For each k, using algorithm Disj, compute the disjunction Q_{1k} of the immediate subgraphs of P_1 corresponding to the B_j such that $B_j \cdot \theta = B'_k$; if $Q_{1k} = \mathsf{T}$, let L'_k be the (possibly empty) tail $[P_i]_{2 \leq i \leq n}$ of L, otherwise let L'_k be the sorted list obtained by inserting Q_{1k} at its proper position in $[P_i]_{2 \leq i \leq n}$. Form the substitution $\theta'_k = \theta \cup \{(y, B'_k)\}$, then compute the restriction θ''_k of θ'_k to the set of variables that have secondary occurrences in the graphs that comprise L'_k. And recursively compute

$$Q'_k = \mathsf{RelP'}(L'_k, V, \eta, \theta''_k). \tag{7}$$

   Then compute the result R as follows:

   (a) If $Q'_k = \mathsf{F}$ for $1 \leq k \leq h$, let $R = \mathsf{F}$.

   (b) Otherwise, if $y \notin V$, let R be the MDG with root labeled $y \cdot \eta$ and one root edge labeled B'_k leading to Q'_k for each k such that $Q'_k \neq \mathsf{F}$.

   (c) Otherwise use algorithm Disj to compute the disjunction of the $Q'_k \neq \mathsf{F}$, and let the result be R.

3. The front top symbol is a cross-op f. (The root labels of the P_i for $1 \leq i \leq m$ are cross-terms with cross-op f.) In that case compute $P'_i = \mathsf{ST}(P_i, \theta)$ for $1 \leq i \leq m$, using the algorithm ST described below. If any of the P'_i is F, return $R = \mathsf{F}$. Otherwise construct a (possibly empty) sorted list L', starting with the (possibly empty) tail $[P_j]_{m < j \leq n}$ of L and inserting in turn each $P'_i \neq \mathsf{T}$ (if any) where it belongs. Compute the restriction θ' of θ to the set of variables that have secondary occurrences in the graphs that comprise L'. Then use the algorithm $\mathsf{RelP''}$ described below to compute

$$R = \mathsf{RelP''}(L', V, \eta, \theta', f). \tag{8}$$

Return R after entering it in the result table.

Auxiliary Recursive Algorithm RelP”

$R = \mathsf{RelP''}([P_i]_{1 \leq i \leq n}, V, \eta, \theta, f)$.

- Arguments: $L = [P_i]_{1\leq i\leq n}$, V, η, and θ are as in RelP', with P_i of type $X_i \rightarrow Y_i$, $X = \bigcup_{1\leq i\leq n} X_i$, $Y = \bigcup_{1\leq i\leq n} Y_i$, and θ of type $X' \rightarrow Y'$. The fourth argument f is a cross-op with target-type α. Let $X'' = X \cup X'$, $Y'' \cup Y'$. The conditions on the arguments are as follows:

1...4: as in RelP'.

5. When $n > 0$, if s is the front top symbol of L, $f \leq s$ in custom order.
6. Cross-terms with cross-op f in L have no occurrences of variables in Y'.

- Result: as in RelP', an MDG R, which can be F, T, or of type $(X''\backslash Y'') \rightarrow ((Y''\backslash V)\cdot\eta)$ such that

$$\models \theta \Rightarrow (R \cdot \eta^{-1} \Leftrightarrow (\exists V)(\bigwedge_{1\leq i\leq n} P_i)). \tag{9}$$

If $n = 0$, return $R = \mathsf{T}$. Otherwise, if $f < s$, compute and return

$$R = \mathsf{RelP'}(L, V, \eta, \theta). \tag{10}$$

Otherwise, check if the result has already been computed using the same key and result table as in RelP'. If no entry is found, proceed as follows.

Let m be the number of P_i with same root label A at the front of L; A is a cross-term with cross-op f. (Note that there may be MDGs in L after the first m ones whose root label is a also a cross-term with cross-op f, a cross-term that comes after A in node-label order.) Let $[b_j]_{1\leq j\leq l}$ be the list of constants that appear as root edge labels in all the P_i, arranged in standard order. If the list is empty ($l = 0$), return $R = \mathsf{F}$. Otherwise, for $1 \leq j \leq l$ and $1 \leq i \leq m$, let Q_{ij} be the immediate subgraph of P_i corresponding to the edge labeled b_j.

For each j, construct a sorted list L'_j, starting with the (possibly empty) tail $[P_k]_{m<k\leq n}$ of L and inserting each Q_{ij} where it belongs. Compute the restriction θ' of θ to the set of variables that have secondary occurrences in the graphs that comprise L'_j. And recursively compute

$$Q'_j = \mathsf{RelP''}(L'_j, V, \eta, \theta', f). \tag{11}$$

Then compute the result R as follows:

1. If $Q'_j = \mathsf{F}$ for $1 \leq j \leq l$, let $R = \mathsf{F}$.
2. If the Q'_j are all the same MDG for $1 \leq j \leq l$, and if the b_j comprise all the individual constants in the enumeration of α, let R be that MDG.

3. Otherwise let R be the MDG with root labeled A and one root edge labeled b_j leading to Q'_j for each j such that $Q'_j \neq \mathsf{F}$.

Return R after entering it in the result table.

Auxiliary recursive algorithm ST ("Substitution at the Top segment")

$R = \mathsf{ST}(P, Xop, \theta)$

- Arguments: An MDG P, of type $X \to Y$, having a cross-op f as top symbol, a cross-op Xop which is supposed to be the top symbol of the segment to be processed, and a well-formed substitution θ, of type $X' \to Y'$, such that every variable in Y comes after every variable in Y' in custom symbol order.
- Result: An MDG R such that
  - $\models \theta \Rightarrow (P \Leftrightarrow R)$.
  - R is F, T, or an MDG of type $(X \cup X') \to Y$, with f as top symbol and no occurrences of Y' variables in the root segments.

If There are no secondary variables in P to be substituted, i.e., not in $dom(\theta)$, then return $R = P$. If P is out of the processing segment, i.e., $Xop \neq f$, then return $R = P$. Otherwise, check the result table to see if the result has already been computed. The result table for ST maps the graph ID of P, cross-op Xop and θ to the graph ID of R. If no entry is found, proceed as follows.

Let term A be the root of P, $[b_j]_{1 \leq j \leq l}$ be the labels of root edges and $[P_j]_{1 \leq j \leq l}$ be the immediate subgraphs. Compute the new root label $A' = A \cdot \theta$.

- If A' can be rewritten into an individual constant c by applying unconditional rewrite rules, then we distinguish two cases:
  - If there exists a root edge $b_j (1 \leq j \leq l)$ such that $b_j = c$, then compute $R = \mathsf{ST}(P_j, Xop, \theta)$. Return R after entering it into the result table.
  - Otherwise let $R = \mathsf{F}$.
- Otherwise, for each immediate subgraph $P_j (1 \leq j \leq l)$, recursively call

$$Q_j = \mathsf{ST}(P_j, Xop, \theta)$$

$$P'_j = \mathsf{ST1}(Q_j, (A', b_j))$$

where $\mathsf{ST1}$ is an auxiliary algorithm to canonize (i.e., to insert A' into the proper position) of $Q_j (1 \leq j \leq l)$. Disjunct P'_j for $1 \leq j \leq l$ and let the result be R. Return R after entering it into the result table.

Auxiliary recursive algorithm ST1

$R = \mathsf{ST1}(P, (C, c))$

- Arguments: An MDG P, of type $X \rightarrow Y$, and a root-edge pair (C, c).
- Result: An MDG R such that $\models R \Leftrightarrow P \wedge Q$ where Q is an MDG with root segment $[C, c]$ and immediate subgraph T.

Let term A be the root of P, $[b_j]_{1 \leq j \leq l}$ be the labels of root edges and $[P_j]_{1 \leq j \leq l}$ be the immediate subgraphs. We distinguish the following cases:

- If $P = \mathsf{F}$, then $R = \mathsf{F}$.
- If $P = \mathsf{T}$, then R is a graph with root C, root edge c and immediate subgraph T.
- If $A = C$ (in this case, it is necessary to check for duplication or contradiction), then if there exits one root edge of P which equals c, then let R be the immediate subgraph of P which c leads to, otherwise let $R = \mathsf{F}$.
- If A comes after C in term order (i.e., the cross-terms are already in order), then assemble graph R with root labeled C, one root edge labeled c and an immediate subgraph P.
- Otherwise, recursively insert root-edge pair (C, c) into the proper position of P. For each immediate subgraph $[P_j]_{1 \leq j \leq l}$ of P, recursively call

  $$P'_j = \mathsf{ST1}(P_j, (C, c)).$$

  If all the $P'_j = \mathsf{F}$ for $1 \leq j \leq l$, then return $R = \mathsf{F}$, otherwise let R be the MDG with root labeled A, root edge labeled b_j leading to P'_j for each j such that $P'_j \neq \mathsf{F}$.

4 Pruning-by-subsumption

$P' = \mathsf{PbyS}(P, Q)$

- Arguments: Two MDGs P and Q of type $V \rightsquigarrow Y$, where V contains only abstract variables that do not participate in the custom symbol ordering; P and Q can both be T or F.

- Result: An MDG P', derivable from P by pruning, such that:

$$\models P \vee (\exists V)Q \Leftrightarrow P' \vee (\exists V)Q \tag{12}$$

  Since P' is derivable from P by pruning, it is, like P, of type $V \rightsquigarrow Y$. Moreover, if P is of type $V \rightarrow Y_1$, $Y_1 \subseteq Y$, then P' is also of type $V \rightarrow Y_1$ when it is not F.

If $Q = \mathsf{F}$ return $P' = P$. If $Q = \mathsf{T}$ or $P = \mathsf{F}$ return $P' = \mathsf{F}$. Otherwise compute and return

$$P' = \mathsf{PbyS'}(P, [(Q, \emptyset)]). \tag{13}$$

Auxiliary algorithm PbyS'

$P' = \mathsf{PbyS'}(P, [(Q_i, \theta_i)]_{1 \leq i \leq n})$

- Arguments: An MDG P of type $V \rightsquigarrow Y$ other than F, and a list of graph-substitutions pairs (Q_i, θ_i), $1 \leq i \leq n$, possibly empty ($n \geq 0$), where each Q_i is an MDG of type $V \rightsquigarrow Y$ other than T or F, and each θ_i is a substitution that maps some of the variables in V to concretely-reduced terms containing only variables that are elements of V. The list of graph-substitution pairs is sorted and without duplicates. The sorting criterion does not matter: the purpose of keeping the list sorted is only to increase the hit rate in the result table.

- Result: An MDG P' of type $V \rightsquigarrow Y$, derivable from P by pruning, such that

$$\models P \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \Leftrightarrow P' \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \tag{14}$$

  where a 0-ary conjunction ($n = 0$) means T by definition, and the notation $Q * \theta$ is defined as follows, for an arbitrary MDG Q of type $V \rightsquigarrow Y$ and an arbitrary substitution θ with domain $V_1 \subseteq V$: let $V_2 = V \setminus V_1$; let W be a set of abstract variables disjoint from $V \cup Y$; let $\eta : V_2 \mapsto W$ be a renaming substitution; and let $\theta' = \theta \cup \eta$; then $Q * \theta$ stands for

$$(\exists W)(Q \cdot \theta') \tag{15}$$

(defined up to renaming of bound variables).

If $P = \mathsf{T}$ then: if $n = 0$, return $P' = \mathsf{F}$, otherwise return $P' = \mathsf{T}$. If $P \neq \mathsf{T}$, check if the result has already been computed. The result table for PbyS maps a list of graph IDs (extracted from P and the Q_i), and the substitutions θ_i, to a graph. If no entry is found, proceed as follows.

For each segment σ issuing from N, R being the subgraph that σ leads to, compute

$$R' = \mathsf{PbyS"}(\sigma, R, [(Q_i, \theta_i)]_{1 \leq i \leq n}) \tag{16}$$

and replace R with R'. ¿From the resulting graph, obtain P' by eliminating F if necessary (i.e. if $R' = \mathsf{F}$) and removing redundant nodes. Return P' after entering it in the result table.

Auxiliary algorithm PbyS"

$R' = \mathsf{PbyS"}(\sigma, R, [(Q_i, \theta_i)]_{1 \leq i \leq n})$

- Arguments: σ is a segment of type $V \rightsquigarrow Y$, R is an MDG of type $V \rightsquigarrow Y$ other than F, the list $[(Q_i, \theta_i)]_{1 \leq i \leq n}$ is as before, with $n > 0$.

- Result: a $V \rightsquigarrow Y$ graph R', derivable from R by pruning, such that

$$\models (\sigma \wedge R) \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \Leftrightarrow (\sigma \wedge R') \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right). \tag{17}$$

Compute

$$\{L_j\}_{1 \leq j \leq m} = \mathsf{PD}(\sigma, [(Q_i, \theta_i)]_{1 \leq i \leq n}). \tag{18}$$

If $m = 0$, return $R' = R$. Otherwise compute a series of MDGs

$$P_{j+1} = \mathsf{PbyS'}(P_j, L_j), \tag{19}$$

starting with $P_1 = R$ and ending as soon as $P_{j+1} = \mathsf{PbyS'}(P_j, L_j) = \mathsf{F}$, or else ending with P_{m+1}. Return as R' the last MDG in the series.

Auxiliary algorithm PD ("push down")

$S = \mathsf{PD}(\sigma, [(Q_i, \theta_i)]_{1 \le i \le n})$

- Arguments: σ and $[(Q_i, \theta_i)]_{1 \le i \le n}$ are as before, with $n \ge 0$.
- Result: S is a set (implemented, say, as a list without duplicates) each element of which is a sorted list without duplicates $[(R_j, \theta'_j)]_{1 \le j \le m}$, $m \ge 0$, such that
  - Each R_j is an MDG of type $V \rightsquigarrow Y$ other than T or F, whose top symbol comes after the top symbol of σ
  - Each θ'_j is a substitution mapping variables in V to concretely-reduced terms constructed from variables in V
  - $\models \left(\sigma \wedge \left(\bigwedge_{1 \le j \le m}(R_j * \theta'_j)\right)\right) \Rightarrow \left(\bigwedge_{1 \le i \le n}(Q_i * \theta_i)\right)$.

Given two sets S and S' whose elements are sorted lists without duplicates, the *cross-union* of S and S', $S \otimes S'$, is the set of sorted lists without duplicates obtained by merging L and L' for every $(L, L') \in S \times S'$.

If $n = 0$ return $S = \{[\,]\}$. Otherwise let T be the tail of $[(Q_i, \theta_i)]_{1 \le i \le n}$ (i.e. the list whose i-th element is (Q_{i-1}, θ_{i-1}), for $1 \le i \le n-1$), and consider the following cases:

1. The top symbol of Q_1 comes after the top symbol of σ in custom order. Then recursively compute
$$S_{\text{tail}} = \mathsf{PD}(\sigma, T). \tag{20}$$
If $S_{\text{tail}} = \emptyset$, return $S = \emptyset$. Otherwise return
$$S = \{[Q_1, \theta_1]\} \otimes S_{\text{tail}}. \tag{21}$$

2. The top symbol of Q_1 comes before the top symbol of σ. Then:

   (a) If the root of Q_1 is a full node, let Q'_j, $1 \le j \le m$, $m > 0$, be the immediate subgraphs of Q_1 other than T. For each j, let θ'_j be the restriction of θ_1 to the set of variables having secondary occurrences in Q'_j. Obtain a sorted list L by merging T and $[(Q'_j, \theta'_j)]_{1 \le j \le m}$. Then compute and return
$$S = \mathsf{PD}(\sigma, L). \tag{22}$$

   (b) Otherwise return $S = \emptyset$.

3. Q_1 and σ have the same top symbol. Then:

(a) If the common top symbol of Q_1 and σ is a concrete variable y, then σ must consist of a node labeled by y and an edge labeled by an individual constant a. If Q_1 does not have a root edge labeled a, return $S = \emptyset$. If Q_1 has a root edge labeled a, leading to a subgraph Q', recursively compute

$$S_{\text{tail}} = \mathsf{PD}(\sigma, T). \tag{23}$$

If S_{tail} is empty, return $S = \emptyset$. Otherwise S_{tail} will be a singleton, $S_{\text{tail}} = \{L\}$. If $Q' = \mathsf{T}$, return

$$S = S_{\text{tail}} = \{L\}. \tag{24}$$

If $Q' \neq \mathsf{T}$, let θ' be the restriction of θ_1 to the set of variables having secondary occurrences in Q'. Obtain a list L' from L by inserting the pair (Q', θ') at its proper position, if not already present. Return

$$S = \{L'\}. \tag{25}$$

(b) If the common top symbol of Q_1 and σ is an abstract variable y, σ must consist of a node labeled by y and an edge labeled by a term A. Let B_j, $1 \leq j \leq m$, be the labels of the root edges of Q_1 that match A under θ_1, and θ'_j the resulting substitutions: for each j, $1 \leq j \leq m$, θ'_j is an extension of θ_1 such that $B_j \cdot \theta'_j$ coincides with A. For each j, let Q'_j be the subgraph of Q_1 that the edge B_j leads to; if $Q'_j \neq \mathsf{T}$, let θ''_j be the restriction of θ'_j to the set of variables having secondary occurrences in Q'_j and let $L_j = [(Q'_j, \theta''_j)]$; if $Q'_j = \mathsf{T}$ let $L_j = [\,]$.

If $m = 0$ (i.e. if no matches are found), return $S = \emptyset$. Otherwise recursively compute

$$S_{\text{tail}} = \mathsf{PD}(\sigma, T). \tag{26}$$

If $S_{\text{tail}} = \emptyset$, return $S = \emptyset$. Otherwise compute and return

$$S = \{L_j\}_{1 \leq j \leq m} \otimes S_{\text{tail}}. \tag{27}$$

(c) If the common top symbol of Q_1 and σ is a cross-operator f, then (i) σ consists of a series of node-edge pairs $(A_j, a_j)_{1 \leq j \leq m}$, where each A_j is a cross-term with cross-operator f and each a_j is an individual constant, and (ii) Q_1 has a root node labeled by a cross-term B with cross-operator f, and root edges b_k, $1 \leq k \leq l$, labeled by individual constants, leading to subgraphs Q'_k.

For each j, $1 \leq j \leq m$, try to find an extension θ'_j of θ_1 such that $B \cdot \theta'_j$ coincides with A_j, and a k_j such that a_j and b_{k_j} are the same individual constant. If successful and $Q'_{k_j} \neq \mathsf{T}$, let θ''_j be the restriction of θ'_j to the set of variables having secondary occurrences in Q'_{k_j} and recursively compute

$$S'_j = \mathsf{PD}(\sigma, [(Q'_{k_j}, \theta''_j)]). \tag{28}$$

If successful with $Q'_{k_j} = \mathsf{T}$, let $S'_j = \{[\,]\}$. And if not successful, let $S'_j = \emptyset$.
Then, if the root of Q_1 is a full node, let L be the sorted list without duplicates containing the pairs (Q'_k, θ'''_k) for every k in the range $1 \leq k \leq l$ such that $Q'_k \neq \mathsf{T}$, θ'''_k being the restriction of θ_1 to the set of variables having secondary occurrences in Q'_k, and recursively compute

$$S'_0 = \mathsf{PD}(\sigma, L). \tag{29}$$

Otherwise let $S'_0 = \emptyset$.
Having obtained S'_0 and the S'_j, $1 \leq j \leq m$, compute

$$S_{\text{head}} = \bigcup_{0 \leq j \leq m} S'_j. \tag{30}$$

If $S_{\text{head}} = \emptyset$, return $S = \emptyset$. Otherwise recursively compute

$$S_{\text{tail}} = \mathsf{PD}(\sigma, T) \tag{31}$$

and return

$$S = S_{\text{head}} \otimes S_{\text{tail}}. \tag{32}$$

5 Proof of soundness of PbyS

It is straightforward to show that the auxiliary algorithm PD is sound and terminating, i.e. that an invocation

$$\mathsf{PD}(\sigma, [(Q_i, \theta_i)]_{1 \le i \le n}) \tag{33}$$

of the algorithm with appropriate arguments σ and $[(Q_i, \theta_i)]_{1 \le i \le n}$ terminates and returns a set S satisfying the conditions specified above on the result.

It is also easy to see that the soundness and termination of PbyS follow from the soundness and termination of PbyS'. (Note that $(\exists V)Q$ is the same as $Q * \theta$ up to renaming of bound variables.) To establish the soundness and termination of PbyS', consider an invocation

$$\mathsf{PbyS'}(P, [(Q_i, \theta_i)]_{1 \le i \le n}) \tag{34}$$

of the algorithm where the arguments P and $[Q_i]_{1 \le i \le n}$ are as specified in the description of the algorithm. We are going to prove by induction on the structure of the first argument P that for every appropriate second argument $[(Q_i, \theta_i)]_{1 \le i \le n}$ the invocation terminates and produces a result P' that satisfies (14).

If $P = \mathsf{T}$ and $n = 0$, PbyS' immediately returns $P' = \mathsf{F}$. In this case (14) reduces to

$$\models \mathsf{T} \vee \mathsf{T} \Leftrightarrow \mathsf{F} \vee \mathsf{T} \tag{35}$$

which obviously holds. If $P = \mathsf{T}$ with $n \neq 0$, PbyS' returns $P' = \mathsf{T}$ and (14) becomes

$$\models \mathsf{T} \vee \left(\bigwedge_{1 \le i \le n} (Q_i * \theta_i) \right) \Leftrightarrow \mathsf{T} \vee \left(\bigwedge_{1 \le i \le n} (Q_i * \theta_i) \right) \tag{36}$$

which again holds. If $P \neq \mathsf{T}$ then, since $P \neq \mathsf{F}$ by hypothesis, the root N of P is not a leaf node. For each segment σ issuing from N and leading to a subgraph R, the algorithm calls PbyS" to compute R',

$$R' = \mathsf{PbyS"}(\sigma, R, [(Q_i, \theta_i)]_{1 \le i \le n}). \tag{37}$$

PbyS" in turn calls PD to compute a set of lists

$$\{L_j\}_{1 \le j \le m} = \mathsf{PD}(\sigma, [(Q_i, \theta_i)]_{1 \le i \le n}). \tag{38}$$

($m \ge 0$). PD terminates, and each L_j is a list $[(R_{jk}, \eta_{jk})]_{1 \le k \le l_j}$. Using L_j as a shorthand for

$$\bigwedge_{1 \le k \le l_j} R_{jk} * \eta_{jk} \tag{39}$$

we have for each j, $1 \leq j \leq m$,

$$\models \sigma \wedge L_j \Rightarrow \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right). \tag{40}$$

If $m = 0$, PbyS" returns $R' = R$, which obviously satisfies (14). Otherwise it computes

$$P_{j+1} = \mathsf{PbyS'}(P_j, L_j), \tag{41}$$

starting from $j = 1$, until $j = m$ or PbyS' returns F; the last value obtained from PbyS' is returned as R'. By induction hypothesis each invocation of PbyS' terminates and

$$\models P_j \vee L_j \Leftrightarrow P_{j+1} \vee L_j. \tag{42}$$

Hence

$$\models \sigma \wedge (P_j \vee L_j) \Leftrightarrow \sigma \wedge (P_{j+1} \vee L_j) \tag{43}$$

and

$$\models (\sigma \wedge P_j) \vee (\sigma \wedge L_j) \Leftrightarrow (\sigma \wedge P_{j+1}) \vee (\sigma \wedge L_j). \tag{44}$$

Then, by (40),

$$\models (\sigma \wedge P_j) \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \Leftrightarrow (\sigma \wedge P_{j+1}) \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \tag{45}$$

and, since this holds at every step j of the stepwise computation of R' from R,

$$\models (\sigma \wedge R) \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \Leftrightarrow (\sigma \wedge R') \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right). \tag{46}$$

Having computed R' for each σ and R, PbyS' concludes by replacing each R with R', eliminating F, and removing redundant nodes. Clearly, P is logically equivalent to the disjunction of the formulas $\sigma \wedge R$, and P' to the disjunction of the formulas $\sigma \wedge R'$. Thus, since we have established (46) for every R and corresponding R',

$$\models P \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right) \Leftrightarrow P' \vee \left(\bigwedge_{1 \leq i \leq n} (Q_i * \theta_i) \right). \tag{47}$$

We have thus proved the termination and soundness of PbyS'.

References

[1] F. Corella, Z. Zhou, X. Song, M. Langevin, and E. Cerny. Multiway decision graphs for automated hardware verification. To appear in the journal *Formal Methods in System Design*. Available as IBM technical report RC19676.