Semantic Minimization of 3-Valued Propositional Formulae*

Thomas Reps† Alexey Loginov† Mooly Sagiv‡

Abstract

This paper presents an algorithm for a non-standard logic-minimization problem that arises in 3-valued propositional logic. The problem is motivated by the potential for obtaining better answers in applications that use 3-valued logic. An answer of 0 or 1 provides precise (definite) information; an answer of 1/2 provides imprecise (indefinite) information. By replacing a formula φ with a "better" formula ψ, we may improve the precision of the answers obtained. In this paper, we give an algorithm that always produces a formula that is "best" (in a certain well-defined sense).

1. Introduction

A number of recent approaches to verifying properties of hardware and software systems have used 3-valued logics of one sort or another. For instance, 3-valued propositional logic is used in Symbolic Trajectory Evaluation (STE) [23, 12, 7] for verifying properties of hardware systems. In program analysis, dataflow analyses that simultaneously track "may" and "must" information (e.g., [20, 4]) can also be viewed as working in 3-valued propositional logic. In 3-valued logic, a third truth value ("1/2") is introduced to denote uncertainty (or, as it is sometimes expressed, to denote a "truth-value gap" [24, 3]).

This paper concerns a non-standard logic-minimization problem that arises in 3-valued propositional logic. To illustrate the issue, consider the following trivial example (where we use $[\![\varphi]\!](a)$ to denote the value of a formula φ with respect to an assignment a of truth values to propositional variables): In 2-valued logic, the formula $p \vee \neg p$ is equivalent to the formula $\mathbf{1}$; that is, in 2-valued logic, $[\![p \vee \neg p]\!](a) = [\![\mathbf{1}]\!](a) = 1$ for all assignments a that assign some truth value to p. In contrast, $p \vee \neg p$ and $\mathbf{1}$ are not equivalent in 3-valued logic, as can be seen by considering their values under various (3-valued) assignments:

$$\begin{array}{lcl} [\![\mathbf{1}]\!]([p \mapsto 0]) & = 1 & = [\![p \vee \neg p]\!]([p \mapsto 0]) \\ [\![\mathbf{1}]\!]([p \mapsto 1/2]) & = 1 \neq 1/2 & = [\![p \vee \neg p]\!]([p \mapsto 1/2]) \\ [\![\mathbf{1}]\!]([p \mapsto 1]) & = 1 & = [\![p \vee \neg p]\!]([p \mapsto 1]) \end{array}$$

In particular, for $[p \mapsto 1/2]$, the formula $\mathbf{1}$ provides a definite answer (i.e., 1), but $p \vee \neg p$ provides an indefinite answer (i.e., 1/2).

As this example demonstrates, in 3-valued logic there is a notion of one formula being "better" than another: among the formulae that are equivalent in 2-valued logic, some may evaluate to a definite value on more 3-valued assignments. Our interest in this phenomenon is motivated by the possibility of exploiting it to obtain better answers in applications that use 3-valued logic. An answer of 0 or 1 provides precise (definite) information; an answer of $1/2$ provides imprecise (indefinite) information. By replacing a formula φ with a "better" formula ψ, we may improve the precision of the answers obtained.

One might approach the problem of "improving" φ by simplifying φ's subterms using rewriting rules, such as

$$\gamma \vee \neg\gamma \longrightarrow \mathbf{1} \qquad \gamma \wedge \neg\gamma \longrightarrow \mathbf{0}.$$

However, one is left with the question of whether such an approach always produces a formula that is as good as possible.

In this paper, we give an algorithm that uses a different approach; the algorithm always produces a formula that is "best" (in a certain well-defined sense). The paper makes the following contributions:

- We provide a formalization of the "semantic-minimization" problem: Given a formula φ, the goal is to find a best formula ψ. (See Sect. 3.)
- We show that one can always find a best formula.
- We present several methods for creating a best formula.

The remainder of the paper is organized as follows: Sect. 2 introduces some terminology and notation. Sect. 3 defines the problem of semantic minimization for 3-valued propositional logic. Sect. 4 presents a couple of different methods for performing semantic minimization. Sect. 5 defines a semantic-minimization algorithm that, for efficiency, uses Binary Decision Diagrams in certain stages. Sect. 6 discusses related work. (Several proofs appear in App. A.)

2. Terminology and Notation

In this section, we define a standard 2-valued propositional logic, together with a related 3-valued propositional logic with a semantics due to Kleene [13].

2.1. 2-Valued Propositional Logic

We write propositional formulae over a set of propositional variables $\mathcal{V}$ using the propositional constants $\mathbf{0}$ and $\mathbf{1}$, the

*This work was supported in part by NSF under grant CCR-9619219, by ONR under contracts N00014-01-1-0796 and N00014-01-1-0708, and by the Alexander von Humboldt Foundation.

†Comp. Sci. Dept.; Univ. of Wisconsin; 1210 W. Dayton St.; Madison, WI 53706; USA. E-mail: {reps, alexey}@cs.wisc.edu.

‡School of Comp. Sci.; Tel-Aviv Univ.; Tel-Aviv 69978; Israel. E-mail: sagiv@math.tau.ac.il.

unary connective $\neg$, and the binary connectives $\wedge$ and $\vee$. We also make use of conditional expressions, for which we adopt a C-like syntax: $\varphi_1 \ ? \ \varphi_2 : \varphi_3$.[1]

For brevity, we will sometimes use juxtaposition in place of $\wedge$, and use an overbar to denote negation; e.g., $(\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge \neg z)$ may also be written as $\overline{x}yz \vee x\overline{y}\,\overline{z}$.

Propositional variables and negations of propositional variables will be referred to collectively as **literals**.

The (2-valued truth-functional) semantics for propositional logic is defined in the standard way:

Definition 2.1 *An* **assignment** a *is a (finite) function in* $\mathcal{V} \to \{0,1\}$. *Given a formula* φ *over the propositional variables* $x_1, \ldots, x_n$ *and an assignment* a *that is defined on (at least)* $x_1, \ldots, x_n$, *the* **2-valued truth-functional meaning** *of* φ *with respect to* a, *denoted by* $[\![\varphi]\!](a)$, *is the truth value in* $\{0,1\}$ *defined inductively as follows:*

$$\begin{array}{rcl@{\qquad}rcl}
[\![\mathbf{0}]\!](a) & = & 0 & [\![x_i]\!](a) & = & a(x_i) \\
[\![\mathbf{1}]\!](a) & = & 1 & [\![\neg\varphi]\!](a) & = & 1 - [\![\varphi]\!](a) \\
\multicolumn{3}{r@{\qquad}}{[\![\varphi_1 \wedge \varphi_2]\!](a) = \min([\![\varphi_1]\!](a), [\![\varphi_2]\!](a))} & & & \\
\multicolumn{3}{r@{\qquad}}{[\![\varphi_1 \vee \varphi_2]\!](a) = \max([\![\varphi_1]\!](a), [\![\varphi_2]\!](a))} & & & \\
\multicolumn{3}{r@{\qquad}}{[\![\varphi_1 \ ? \ \varphi_2 : \varphi_3]\!](a) = [\![(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3)]\!](a)} & & &
\end{array}$$

We say that a **satisfies** φ, *denoted by* $a \models \varphi$, *iff* $[\![\varphi]\!](a) = 1$. □

Later on, it will be useful to be able to indicate that various semantically equivalent formulae are syntactically related in certain ways. We use $\equiv$ to denote syntactic equality between formulae, up to rearrangements of conjuncts and disjuncts; we use $\equiv_{\text{DM}}$ to denote $\equiv$, extended with applications of De Morgan's laws and introductions/cancellations of double negations. For instance,

$$\begin{array}{l}
\neg((x \wedge \neg y) \vee (\neg x \wedge z)) \equiv \neg((z \wedge \neg x) \vee (\neg y \wedge x)) \\
\neg((x \wedge \neg y) \vee (\neg x \wedge z)) \not\equiv (\neg z \vee x) \wedge (y \wedge \neg x) \\
\neg((x \wedge \neg y) \vee (\neg x \wedge z)) \equiv_{\text{DM}} (\neg z \vee x) \wedge (y \wedge \neg x) \\
\neg((x \wedge \neg y) \vee (\neg x \wedge z)) \not\equiv_{\text{DM}} \begin{array}{l} (\neg z \wedge y) \vee (\neg z \wedge \neg x) \\ \vee \ (x \wedge y) \vee (x \wedge \neg x) \end{array}
\end{array}$$

All four of the formulae used above have the same 2-valued truth-functional meaning. We reserve "=" for semantic equality (e.g., $[\![\neg((x \wedge \neg y) \vee (\neg x \wedge z))]\!] = [\![(\neg z \wedge y) \vee (\neg z \wedge \neg x) \vee (x \wedge y) \vee (x \wedge \neg x)]\!]$).

2.2. 3-Valued Propositional Logic

Moving now to 3-valued logic, the language of formulae that we work with is identical to that defined in Sect. 2.1, except that there is one additional propositional constant, $\mathbf{1/2}$. At the semantic level, a third truth value—$1/2$—is introduced to denote uncertainty. We say that the values 0 and 1 are **definite values** and that $1/2$ is an **indefinite value**, and define a partial order $\sqsubseteq$ on truth values to reflect their degree of definiteness (or **information content**): $l_1 \sqsubseteq l_2$ denotes that l_1 is at least as definite as l_2:

[1] For now, one can think of $\varphi_1 \ ? \ \varphi_2 : \varphi_3$ as a shorthand for $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3)$ (see Defn. 2.1). Later on, for technical reasons, we will consider it to be a shorthand for $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3) \vee (\varphi_2 \wedge \varphi_3)$ (see Defn. 2.3 and Ex. 5.9).

$\neg$	
0	1
1/2	1/2
1	0

$\wedge$	0	1/2	1
0	0	0	0
1/2	0	1/2	1/2
1	0	1/2	1

$\vee$	0	1/2	1
0	0	1/2	1
1/2	1/2	1/2	1
1	1	1	1

$v_1 \ ? \ v_2 : v_3$	v_2 \ v_3	0	1/2	1
$v_1 = 0$	0	0	1/2	1
	1/2	0	1/2	1
	1	0	1/2	1
$v_1 = 1/2$	0	0	1/2	1/2
	1/2	1/2	1/2	1/2
	1	1/2	1/2	1
$v_1 = 1$	0	0	0	0
	1/2	1/2	1/2	1/2
	1	1	1	1

Table 1. The 3-valued truth tables for the propositional operators.

Definition 2.2 [Information Order]. *For* $l_1, l_2 \in \{0, 1/2, 1\}$, *we define the* **information order** *on truth values as follows:* $l_1 \sqsubseteq l_2$ *iff* $l_1 = l_2$ *or* $l_2 = 1/2$. *We use* $l_1 \sqsubset l_2$ *when* $l_1 \sqsubseteq l_2$ *and* $l_1 \neq l_2$. *The symbol* $\sqcup$ *denotes the least-upper-bound operation with respect to* $\sqsubseteq$*:*

$\sqcup$	0	1/2	1
0	0	1/2	1/2
1/2	1/2	1/2	1/2
1	1/2	1/2	1

□

We now generalize Defn. 2.1 to define the meaning of a formula with respect to a 3-valued assignment A. (Our convention will be to use lower-case letters for 2-valued assignments, and upper-case letters for 3-valued assignments.)

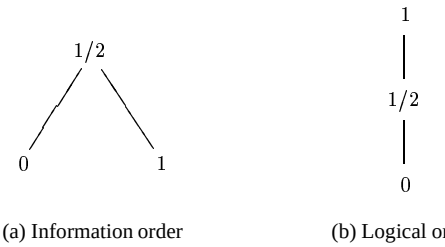
Definition 2.3 *A* **3-valued assignment** A *is a (finite) function in* $\mathcal{V} \to \{0, 1, 1/2\}$. *Given a formula* φ *over the propositional variables* $x_1, \ldots, x_n$ *and an assignment* A *that is defined on (at least)* $x_1, \ldots, x_n$, *the* **3-valued truth-functional meaning** *of* φ *with respect to* A, *denoted by* $[\![\varphi]\!](A)$, *yields a truth value in* $\{0, 1, 1/2\}$. *The meaning of* φ *is defined inductively as in Defn. 2.1, with the following changes:*

$$\begin{array}{rcl}
[\![\mathbf{1/2}]\!](A) & = & 1/2 \\
[\![\varphi_1 \ ? \ \varphi_2 : \varphi_3]\!](A) & = & [\![(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3) \vee (\varphi_2 \wedge \varphi_3)]\!](A)
\end{array}$$

We say that A **potentially satisfies** φ, *denoted by* $A \models \varphi$, *iff* $[\![\varphi]\!](A) \sqsupseteq 1$ *(i.e.,* $[\![\varphi]\!](A) = 1/2$ *or* $[\![\varphi]\!](A) = 1$*).* □

The 3-valued truth tables for the propositional operators are shown in Tab. 1.

In 2-valued logic, $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3)$ and $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3) \vee (\varphi_2 \wedge \varphi_3)$ yield equivalent definitions



(a) Information order (b) Logical order

Figure 1. The semi-bilattice of 3-valued logic.

of the 2-valued truth-functional meaning of $\varphi_1 \,?\, \varphi_2 : \varphi_3$. In 3-valued logic, however, $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \varphi_3) \vee (\varphi_2 \wedge \varphi_3)$ yields a more precise semantics. In the truth table for "$v_1 \,?\, v_2 : v_3$", the sub-table for "$v_1 = 1/2$" (which can be obtained by evaluating $(\mathbf{1/2} \wedge v_2) \vee (\neg\mathbf{1/2} \wedge v_3) \vee (v_2 \wedge v_3)$) is identical to the truth table for "$v_2 \sqcup v_3$" (cf. Defn. 2.2). Consequently, $\mathbf{1/2} \,?\, v_2 : v_3$ yields a definite value when v_2 and v_3 are either both 0 or both 1. In particular, for the assignment $[v_2 \mapsto 1, v_3 \mapsto 1]$, $(\mathbf{1/2} \wedge v_2) \vee (\neg\mathbf{1/2} \wedge v_3)$ yields the indefinite value $1/2$, whereas $(\mathbf{1/2} \wedge v_2) \vee (\neg\mathbf{1/2} \wedge v_3) \vee (v_2 \wedge v_3)$ yields 1. (We will look at this from another vantage point later, in Ex. 5.9.)

We will use $\sqcup$ as a binary connective for constructing formulae: $\varphi_1 \sqcup \varphi_2$ is a shorthand for $\mathbf{1/2} \,?\, \varphi_1 : \varphi_2$.

It should be noted that throughout the remainder of the paper, the symbol $\models$ means the potential-satisfaction relation of Defn. 2.3, even when we are talking about a 2-valued assignment. For instance, $[p \mapsto 1] \models p \wedge \mathbf{1/2}$ because $[\![p \wedge \mathbf{1/2}]\!]([p \mapsto 1]) = 1/2$.

As shown in Fig. 1, the values 0, 1, and $1/2$ form a mathematical structure known as a semi-bilattice (see [11]). A semi-bilattice has two orderings: the **information order** and the **logical order**:

- The information order is the one defined in Defn. 2.2, which captures "(un)certainty".
- The logical order is the one used in Tab. 1: that is, $\wedge$ and $\vee$ are meet and join in the logical order (e.g., $1 \wedge 1/2 = 1/2$, $1 \vee 1/2 = 1$, $1/2 \wedge 0 = 0$, $1/2 \vee 0 = 1/2$, etc.).

A value that is "far enough up" in the logical order indicates "potential truth", and is called a **designated value**. We take $1/2$ and 1 as the designated values; thus, an assignment A potentially satisfies a formula when the formula's truth-functional meaning with respect to A is one of the designated values.

The information ordering on values is extended pointwise to an information ordering on assignments (also denoted by $\sqsubseteq$); e.g., $[p \mapsto 1, q \mapsto 0] \sqsubseteq [p \mapsto 1, q \mapsto 1/2]$, $[p \mapsto 1, q \mapsto 0] \sqsubseteq [p \mapsto 1/2, q \mapsto 0]$, $[p \mapsto 1, q \mapsto 0] \sqsubseteq [p \mapsto 1/2, q \mapsto 1/2]$, etc. When A does not contain any bindings of a propositional variable to the value 1/2, we say that A is **definite** (and usually write it with a lower-case a).

Kleene's 3-valued semantics is monotonic in the information order (cf. Tab. 1 and Defn. 2.3):

Lemma 2.4 *Let φ be a formula, and let A and A' be two assignments such that $A \sqsubseteq A'$. Then $[\![\varphi]\!](A) \sqsubseteq [\![\varphi]\!](A')$.* $\square$

Kleene's semantics retains a number of properties that are familiar from 2-valued logic, including De Morgan's laws and the ability to introduce/cancel double negations. For this reason, $\equiv$ and $\equiv_{\text{DM}}$ relate formulae that are semantically equivalent in 3-valued logic.

Lem. 2.4 provides a way to relate the 2-valued and 3-valued truth-functional meanings of formulae: the value obtained by evaluating any formula φ with respect to a 3-valued assignment A is always safe (i.e., greater than or equal to in the information order) compared to the value obtained by evaluating φ with respect to any 2-valued assignment $a \sqsubseteq A$. In particular,

- If $[\![\varphi]\!](A)$ yields a definite value, then $[\![\varphi]\!](a)$ must yield the same definite value.
- If $[\![\varphi]\!](A)$ yields $1/2$, then $[\![\varphi]\!](a)$ can be either 0 or 1.

We say that a 3-valued assignment A **represents** all 2-valued assignments $a \sqsubseteq A$. Another outlook on the way 2-valued and 3-valued assignments are related stems from the following corollary (which follows immediately from Lem. 2.4):

Corollary 2.5 *Suppose that A represents a. If $a \models \varphi$, then $A \models \varphi$.* $\square$

Thus, if we think of a propositional formula φ of 2-valued logic as a device for accepting a set S of 2-valued assignments, then when φ is considered as a formula of 3-valued logic, the *potential*-satisfaction relation corresponds to an implicit condition for accepting/rejecting an entire set of 2-valued assignments—those that are represented by a 3-valued assignment. Moreover, acceptance via the potential-satisfaction relation is *safe* with respect to the actual set of 2-valued assignments accepted by φ:

$$\{a \in \text{def. assignments} \mid a \models \varphi\} \subseteq \{a \text{ rep. by } A \mid A \models \varphi\}$$

This point of view is useful when 3-valued assignments are used as the space of abstract values in an abstract interpretation (e.g., see Chou's account of STE in abstract-interpretation terms [7]). We will also adopt this viewpoint in Sect. 3.2 in order to justify our definition of the semantic-minimization problem.

3. The Semantic Minimization Problem

In Sect. 1, we observed that although the formula $\mathbf{1}$ is *equivalent* to $p \vee \neg p$ in 2-valued logic, in 3-valued logic, $\mathbf{1}$ is *better than* $p \vee \neg p$. This raises the question, "For any given φ, is there always a best formula?", which, in turn, raises the question, "What properties must a 'best' formula possess?"

3.1. Definition of the Minimization Problem

The concept of a "best formula" is formalized using the concept of a formula's "supervaluational meaning" [24]:

Definition 3.1 *Given a formula φ and assignment A, the* **3-valued supervaluational meaning** *of φ with respect to A, denoted by $\langle\!\langle\varphi\rangle\!\rangle(A)$, is the truth value in $\{0, 1, 1/2\}$ defined by*

$$\langle\!\langle\varphi\rangle\!\rangle(A) = \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a).$$

□

Definition 3.2 *Given a propositional formula φ, we say that the formula ψ is a* **semantically minimal variant** *of φ iff, for all 3-valued assignments A, $[\![\psi]\!](A) = \langle\!\langle\varphi\rangle\!\rangle(A)$. The* **semantic-minimization problem** *for propositional logic is as follows:*

> *Given a propositional formula φ, find a formula ψ that is a semantically minimal variant of φ.*

□

For instance, $\mathbf{1}$ is a semantically minimal variant of $p \vee \neg p$; in particular,

$$\begin{aligned}\langle\!\langle\varphi\rangle\!\rangle([p \mapsto 1/2]) &= \bigsqcup_{a \in \{[p\mapsto 0],[p\mapsto 1]\}} [\![\varphi]\!](a) \\ &= 1 \sqcup 1 \\ &= [\![\mathbf{1}]\!]([p \mapsto 1/2]).\end{aligned}$$

Similarly, $\mathbf{0}$ is a semantically minimal variant of $p \wedge \neg p$.

3.2. Justification of the Problem Definition

It is worthwhile to spend a few moments to consider why Defn. 3.2 is the appropriate definition of the semantic-minimization problem. Let us contrast Defn. 3.2 with a possible alternative definition:

Strawman Definition 3.3 *Given a propositional formula φ, we say that ψ is a* **semantically minimal variant** *of φ iff for all 3-valued assignments A, $[\![\psi]\!](A) \sqsubseteq [\![\varphi]\!](A)$.* □

The motivation behind this definition is that, by using ψ in place of φ, (i) we could sometimes obtain answers that are strictly more definite, and (ii) we could never obtain answers that are strictly less definite. The question we must ask, however, is whether we would obtain *acceptable* answers with Strawman Defn. 3.3.

Note that with Defn. 3.2, neither $\mathbf{0}$ nor $\mathbf{1}$ is a semantically minimal variant of $\mathbf{1/2}$. In contrast, with Strawman Defn. 3.3, the formulae $\mathbf{0}$ and $\mathbf{1}$ would both be semantically minimal variants of $\mathbf{1/2}$. The latter situation would get us into trouble because their meanings, $[\![\mathbf{0}]\!] = \lambda a.0$ and $[\![\mathbf{1}]\!] = \lambda a.1$, are in conflict; that is, with Strawman Defn. 3.3, the admissible ψ's are not all semantically equivalent in 2-valued logic. In contrast, with Defn. 3.2, the admissible ψ's are all semantically equivalent in 2-valued logic; by definition, they all have the meaning $\langle\!\langle\varphi\rangle\!\rangle$—the supervaluational meaning of φ.

An even better way to see that Strawman Defn. 3.3 is unsatisfactory is by considering how the two concepts of "semantically minimal variant" relate to the view of a formula as a device for accepting a set of assignments (cf. the discussion following Cor. 2.5).

Desideratum 3.4 [Better Acceptance Device I]. *When we view a formula as a device for accepting a set of assignments, we would like for ψ to correspond to a* **better** *acceptance device than φ. That is, when applied to an assignment A, either 2-valued or 3-valued, ψ may yield a more precise acceptance condition than φ: in circumstances in which φ waffles (i.e., $[\![\varphi]\!](A) = 1/2$), ψ can either*

- *waffle itself (i.e., $[\![\psi]\!](A) = 1/2$)*
- *accept A (i.e., $[\![\psi]\!](A) = 1$)*
- *reject A (i.e., $[\![\psi]\!](A) = 0$)*

However, ψ must always be safe with respect to the 2-valued assignments that φ accepts:

$$\{a \in \text{def. assignments} \mid a \models \varphi\} \subseteq \{a \text{ rep. by } A \mid A \models \psi\}. \quad (1)$$

Similarly, $\neg\psi$ must always be safe with respect to the 2-valued assignments that $\neg\varphi$ accepts:

$$\{a \in \text{def. assignments} \mid a \models \neg\varphi\} \subseteq \{a \text{ rep. by } A \mid A \models \neg\psi\}. \quad (2)$$

□

For instance, suppose that φ is the formula $\mathbf{1/2}$. Under Strawman Defn. 3.3, the formula $\mathbf{0}$ is an admissible ψ; however, Eqn. (1) does not hold:

$$\begin{aligned}\{a \in \text{def. assignments} \mid a \models \mathbf{1/2}\} &= \{a \in \text{def. assignments}\} \\ &\not\subseteq \emptyset \\ &= \{a \text{ rep. by } A \mid A \models \mathbf{0}\}\end{aligned}$$

Similarly, under Strawman Defn. 3.3, the formula $\mathbf{1}$ is also an admissible ψ; however, Eqn. (2) does not hold:

$$\begin{aligned}\{a \in \text{def. assignments} \mid a \models \neg\mathbf{1/2}\} &= \{a \in \text{def. assignments}\} \\ &\not\subseteq \emptyset \\ &= \{a \text{ rep. by } A \mid A \models \neg\mathbf{1}\}\end{aligned}$$

Under Defn. 3.2, $\mathbf{1/2}$ is an admissible ψ, but $\mathbf{0}$ and $\mathbf{1}$ are not; clearly, with $\varphi = \psi = \mathbf{1/2}$, Eqns. (1) and (2) both hold.

The notion of a "better acceptance device" can also be expressed in a different way, which nicely parallels the statement of Cor. 2.5, but with ψ in the consequent:[2]

[2]The two properties in Desideratum 3.5 are parallel to (i) the property stated in Cor. 2.5, and (ii) the property stated in Cor. 2.5 with φ replaced by $\neg\varphi$.

Desideratum 3.5 [Better Acceptance Device II]. *Let ψ be a semantically minimal variant of φ. Then, for every 3-valued assignment A and 2-valued assignment a such that A represents a, both of the following must hold:*

1. *If $a \models \varphi$, then $A \models \psi$.*
2. *If $a \models \neg\varphi$, then $A \models \neg\psi$.*

□

In contrast to the unsatisfactory results obtained with Strawman Defn. 3.3, we have the following:

Lemma 3.6 *If "semantically minimal variant" means the concept defined in Defn. 3.2, then Desideratum 3.5 holds.*
Proof: The desired properties can be restated as follows:

1. If $[\![\varphi]\!](a) \sqsupseteq 1$, then $[\![\psi]\!](A) \sqsupseteq 1$.
2. If $[\![\neg\varphi]\!](a) \sqsupseteq 1$, then $[\![\neg\psi]\!](A) \sqsupseteq 1$.

These are proved, respectively, as follows:

$$\begin{aligned}
1.\quad [\![\psi]\!](A) &= \langle\!\langle\varphi\rangle\!\rangle(A) \\
&= \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a) \\
&\sqsupseteq 1 \\
2.\quad [\![\neg\psi]\!](A) &= 1 - [\![\psi]\!](A) \\
&= 1 - \langle\!\langle\varphi\rangle\!\rangle(A) \\
&= 1 - \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a) \\
&= \bigsqcup_{a \text{ rep. by } A} (1 - [\![\varphi]\!](a)) \\
&= \bigsqcup_{a \text{ rep. by } A} [\![\neg\varphi]\!](a) \\
&\sqsupseteq 1
\end{aligned}$$

□

Henceforth, the term "semantically minimal variant" means the concept defined in Defn. 3.2.

Strawman Defn. 3.3 was motivated by the desire to obtain more precise answers by using ψ in place of φ. In fact, with Defn. 3.2, we do have such a property:

Lemma 3.7 *If ψ is a semantically minimal variant of φ, then for all 3-valued assignments A, $[\![\psi]\!](A) \sqsubseteq [\![\varphi]\!](A)$.*
Proof: For every semantically minimal variant ψ, we have, by the monotonicity of $[\![\varphi]\!]$ (i.e., Lem. 2.4),

$$\begin{aligned}
[\![\psi]\!](A) &= \langle\!\langle\varphi\rangle\!\rangle(A) \\
&= \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a) \\
&\sqsubseteq [\![\varphi]\!](A).
\end{aligned}$$

□

That the term "semantically minimal variant" is appropriate can be seen from:

Lemma 3.8 *If ψ is a semantically minimal variant of φ, and φ' is any formula that agrees with φ on all 2-valued assignments, then for all 3-valued assignments A, $[\![\psi]\!](A) \sqsubseteq [\![\varphi']\!](A)$.*
Proof: For every semantically minimal variant ψ, we have, by the monotonicity of $[\![\varphi']\!]$ (i.e., Lem. 2.4),

$$\begin{aligned}
[\![\psi]\!](A) &= \langle\!\langle\varphi\rangle\!\rangle(A) \\
&= \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a) \\
&= \bigsqcup_{a \text{ rep. by } A} [\![\varphi']\!](a) \\
&\sqsubseteq [\![\varphi']\!](A).
\end{aligned}$$

□

4. An Algorithm for Semantic Minimization

We now return to the question "For any given φ, is there always a best formula?", and answer it in the affirmative. (More precisely, for each formula φ, there is an equivalence class of best formulae, which may or may not contain φ itself.) Our solution relies on a result, due to Blamey [2, 3, 1], that relates Boolean functions and 3-valued propositional formulae. The result can be stated in a couple of different forms; these yield different methods for creating a best formula. (In Sect. 5, we will focus on a special case of Blamey's result; the latter variant will allow us to define a more efficient minimization algorithm.)

4.1. Realization of Monotonic Boolean Functions Via Formulae

In this section, we review a theorem, due to Blamey, that relates monotonic Boolean functions and 3-valued propositional formulae. We say that a formula φ **realizes** a function f iff $[\![\varphi]\!] = f$. Blamey's theorem states that, for every 3-valued function $f : \{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$ that is monotonic in the information order, there is a formula—built from $\mathbf{0}$, $\mathbf{1}$, $\neg$, $\wedge$, $\vee$, $\sqcup$, and the propositional variables $x_1, \ldots, x_n$—that realizes f.[3] Blamey's proof of the result provides an explicit method for constructing a formula that realizes a given f.

[3] In a slight abuse of notation, we will refer to a Boolean function f as being a member of $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$, but will make use of applications such as $f(A)$, where A is a 3-valued assignment in $\mathcal{V} \to \{0, 1, 1/2\}$. No confusion should result if one thinks of f as a function over the formal parameters $x_1, \ldots, x_n$, and assignment A as supplying values for $x_1, \ldots, x_n$. Under this convention, statements such as $[\![\varphi]\!] = f$ are sensible, even though $[\![\varphi]\!]$ is really of type $(\mathcal{V} \to \{0, 1, 1/2\}) \to \{0, 1, 1/2\}$.

Definition 4.1 *[2, 3, 1]. Let $f(x_1, \ldots, x_n)$ be any monotonic function in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$; let A be a 3-valued assignment that is defined on (at least) $x_1, \ldots, x_n$. We define* One(f, A, i) *and* Zero(f, A, i)*, for $1 \leq i \leq n$, as well as* One$[f]$, Zero$[f]$, *and* Formula$[f]$, *as follows:*

$$\text{One}(f, A, i) \stackrel{\text{def}}{=} \begin{cases} x_i & \text{if } f(A) = 1 \text{ and } A(x_i) = 1 \\ \neg x_i & \text{if } f(A) = 1 \text{ and } A(x_i) = 0 \\ \mathbf{1} & \text{if } f(A) = 1 \text{ and } A(x_i) = 1/2 \\ \mathbf{0} & \text{if } f(A) \sqsupseteq 0 \end{cases} \tag{3}$$

$$\text{Zero}(f, A, i) \stackrel{\text{def}}{=} \begin{cases} \neg x_i & \text{if } f(A) = 0 \text{ and } A(x_i) = 1 \\ x_i & \text{if } f(A) = 0 \text{ and } A(x_i) = 0 \\ \mathbf{0} & \text{if } f(A) = 0 \text{ and } A(x_i) = 1/2 \\ \mathbf{1} & \text{if } f(A) \sqsupseteq 1 \end{cases} \tag{4}$$

$$\text{One}[f] \stackrel{\text{def}}{=} \bigvee_{A \in \{0,1,1/2\}^n} \bigwedge_{1 \leq i \leq n} \text{One}(f, A, i) \tag{5}$$

$$\text{Zero}[f] \stackrel{\text{def}}{=} \bigwedge_{A \in \{0,1,1/2\}^n} \bigvee_{1 \leq i \leq n} \text{Zero}(f, A, i) \tag{6}$$

$$\text{Formula}[f] \stackrel{\text{def}}{=} \text{One}[f] \sqcup \text{Zero}[f] \tag{7}$$

□

Theorem 4.2 [Realization Theorem]. *[2, 3, 1]. For any monotonic function $f(x_1, \ldots, x_n) : \{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$,* Formula$[f]$ *realizes f, i.e.,* $[\![\text{Formula}[f]]\!] = f$.

Proof: See [2]. □

Example 4.3 Consider the formula $\varphi \stackrel{\text{def}}{=} x\overline{y} \vee \overline{x}\,\overline{z} \vee yz$. Its truth-functional semantics, $[\![\varphi]\!]$, is shown in the following truth table:

		z		
	y	0	1/2	1
$x = 0$	0	1	1/2	0
	1/2	1	1/2	1/2
	1	1	1/2	1
$x = 1/2$	0	1/2	1/2	1/2
	1/2	1/2	1/2	1/2
	1	1/2	1/2	1
$x = 1$	0	1	1	1
	1/2	1/2	1/2	1/2
	1	0	1/2	1

Of the twenty-seven disjuncts of One$[[\varphi]]$, all but nine are $\mathbf{000}$: $\overline{x}\,\overline{y}\,\overline{z}$, $\overline{x}\mathbf{1}\overline{z}$, $\overline{x}y\overline{z}$, $\overline{x}yz$, $\mathbf{1}yz$, $x\overline{y}\,\overline{z}$, $x\overline{y}\mathbf{1}$, $x\overline{y}z$, xyz. Of the twenty-seven conjuncts of Zero$[[\varphi]]$, all but two are $\mathbf{1} \vee \mathbf{1} \vee \mathbf{1}$: $x \vee y \vee \overline{z}$ and $\overline{x} \vee \overline{y} \vee z$. The formula that would be created by Eqn. (7) is

$$\text{Formula}[[\![\varphi]\!]] \equiv \begin{pmatrix} \overline{x}\,\overline{y}\,\overline{z} \vee \overline{x}\mathbf{1}\overline{z} \vee \overline{x}y\overline{z} \vee \overline{x}yz \vee \mathbf{1}yz \\ \vee\ x\overline{y}\,\overline{z} \vee x\overline{y}\mathbf{1} \vee x\overline{y}z \vee xyz \end{pmatrix} \sqcup (x \vee y \vee \overline{z})(\overline{x} \vee \overline{y} \vee z). \tag{8}$$

The reader can verify that Eqn. (8) realizes the truth table. □

Defn. 4.1 is somewhat subtle: for instance, an assignment A on which f evaluates to $1/2$ contributes the conjunction $\bigwedge_{1 \leq i \leq n} \text{One}(f, A, i) = \mathbf{00}\ldots\mathbf{0}$ to formula One$[f]$; however, even though $[\![\mathbf{00}\ldots\mathbf{0}]\!](A)$ will necessarily be 0, the overall value of $[\![\text{One}[f]]\!](A)$ is not necessarily 0—due to the contributions from other terms that capture how f behaves on other assignments, i.e., $[\![\bigwedge_{1 \leq i \leq n} \text{One}(f, A', i)]\!](A)$. Despite such effects, Blamey has shown that the Realization Theorem holds [2].

By applying De Morgan's laws, we can derive several variants of Eqn. (7). We have

$$\begin{aligned} \text{Zero}(f, A, i) &\equiv_{\text{DM}} \neg\text{One}(\neg f, A, i) \\ \text{One}(f, A, i) &\equiv_{\text{DM}} \neg\text{Zero}(\neg f, A, i), \end{aligned}$$

which lead to the following variant forms of Eqn. (7):

$$\text{Formula}[f] \stackrel{\text{def}}{=} \text{One}[f] \sqcup \neg\text{One}[\neg f] \tag{9}$$

$$\text{Formula}[f] \stackrel{\text{def}}{=} \neg\text{Zero}[\neg f] \sqcup \text{Zero}[f] \tag{10}$$

$$\text{Formula}[f] \stackrel{\text{def}}{=} \neg\text{Zero}[\neg f] \sqcup \neg\text{One}[\neg f]. \tag{11}$$

With Eqn. (7), the formula constructed has the form "sum-of-products ⊔ product-of-sums"; with Eqn. (9), it has the form "sum-of-products ⊔ ¬sum-of-products"; etc. For instance, using Eqn. (9) in place of Eqn. (7), Eqn. (8) becomes

$$\text{Formula}[[\![\varphi]\!]] \equiv \begin{pmatrix} \overline{x}\,\overline{y}\,\overline{z} \vee \overline{x}\mathbf{1}\overline{z} \vee \overline{x}y\overline{z} \vee \overline{x}yz \vee \mathbf{1}yz \\ \vee\ x\overline{y}\,\overline{z} \vee x\overline{y}\mathbf{1} \vee x\overline{y}z \vee xyz \end{pmatrix} \sqcup \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}). \tag{12}$$

4.2. Creating a Semantically Minimal Variant

Defn. 3.1 and Eqns. (7), (9), (10), or (11) give us the tools needed to construct a semantically minimal variant of a formula φ:

Theorem 4.4 [Minimization Theorem]. *Let ψ be* Formula$[\langle\!\langle\varphi\rangle\!\rangle]$. *Then ψ is a semantically minimal variant of φ.*

Proof: It follows immediately from Defn. 3.1 that $\langle\!\langle\varphi\rangle\!\rangle$, the supervaluational semantics of φ, is a monotonic function in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$. Thus, $\langle\!\langle\varphi\rangle\!\rangle$ meets the conditions of Thm. 4.2:

$$\begin{aligned} [\![\psi]\!] &= [\![\text{Formula}[\langle\!\langle\varphi\rangle\!\rangle]]\!] \\ &= \langle\!\langle\varphi\rangle\!\rangle \qquad \text{(by Thm. 4.2),} \end{aligned}$$

and hence ψ is a semantically minimal variant of φ. $\square$

Example 4.5 Consider again the formula $\varphi \stackrel{\text{def}}{=} x\overline{y} \vee \overline{x}\,\overline{z} \vee yz$. The following table shows the three 3-valued assignments A for which $\langle\!\langle\varphi\rangle\!\rangle(A) \sqsubset [\![\varphi]\!](A)$:

A	$\langle\!\langle\varphi\rangle\!\rangle(A)$	$[\![\varphi]\!](A)$
$[\ x \mapsto 1/2,\ y \mapsto 0,\ z \mapsto 0\ ]$	1	1/2
$[\ x \mapsto 0,\ y \mapsto 1,\ z \mapsto 1/2\ ]$	1	1/2
$[\ x \mapsto 1,\ y \mapsto 1/2,\ z \mapsto 1\ ]$	1	1/2

Thus, the supervaluational semantics, $\langle\!\langle\varphi\rangle\!\rangle$, is as follows:

		z		
	y	0	1/2	1
$x = 0$	0	1	1/2	0
	1/2	1	1/2	1/2
	1	1	1	1
$x = 1/2$	0	1	1/2	1/2
	1/2	1/2	1/2	1/2
	1	1/2	1/2	1
$x = 1$	0	1	1	1
	1/2	1/2	1/2	1
	1	0	1/2	1

The formula that would be created by the semantic-minimization algorithm (using Eqn. (9)) is

$$\text{Formula}[\langle\!\langle\varphi\rangle\!\rangle] \equiv \begin{pmatrix} \overline{x}\,\overline{y}\,\overline{z} \vee \overline{x}\mathbf{1}\overline{z} \vee \overline{x}y\overline{z} \vee \overline{x}y\mathbf{1} \\ \vee\ \overline{x}yz \vee \mathbf{1}\overline{y}\,\overline{z} \vee \mathbf{1}yz \vee x\overline{y}\,\overline{z} \\ \vee\ x\overline{y}\mathbf{1} \vee x\overline{y}z \vee x\mathbf{1}z \vee xyz \end{pmatrix} \sqcup\ \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}). \tag{13}$$

Comparing with Eqn. (12), note the three additional disjuncts in the first part of Eqn. (13): $\mathbf{1}\overline{y}\,\overline{z}$, $\overline{x}y\mathbf{1}$, and $x\mathbf{1}z$. These correspond to the three entries that have value $1/2$ in the truth table for $[\![\varphi]\!]$, but have value 1 in the truth table for $\langle\!\langle\varphi\rangle\!\rangle$. $\square$

4.3. An Improved Construction for Formula[f]

Blamey's thesis contains an improved construction for Formula$[f]$, which often results in a formula that has fewer, and less complicated, constituents.

Definition 4.6 *[2]. Let $f(x_1, \ldots, x_n)$ be any monotonic function in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$; let A be a 3-valued assignment that is defined on (at least) $x_1, \ldots, x_n$.* Formula$[f]$ *is as defined in Eqns. (7), (9), (10), or (11), but with* One$[f]$ *and* Zero$[f]$ *redefined as follows:*

$$\text{One}[f] \stackrel{\text{def}}{=} \bigvee_{\substack{A\,:\,f(A)=1 \text{ and} \\ \forall A' \sqsupset A.f(A')=1/2}} \ \bigwedge_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{One}(f, A, i) \tag{14}$$

$$\text{Zero}[f] \stackrel{\text{def}}{=} \bigwedge_{\substack{A\,:\,f(A)=0 \text{ and} \\ \forall A' \sqsupset A.f(A')=1/2}} \ \bigvee_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{Zero}(f, A, i) \tag{15}$$

An empty disjunction has the value 0*; an empty conjunction has the value* 1. $\square$

The differences between Eqns. (5) and (6) and Eqns. (14) and (15) are that, in the latter,

- The outer connectives are indexed by "$A : f(A) = 1$ and $\forall A' \sqsupset A.f(A') = 1/2$" and "$A : f(A) = 0$ and $\forall A' \sqsupset A.f(A') = 1/2$", respectively, which leads to fewer terms being generated.

- The indices of the inner connectives only range over values of i for which $A(x_i)$ is a definite value, and hence One(f, A, i) and Zero(f, A, i) generate only literals, leaving out unnecessary occurrences of $\mathbf{1}$ and $\mathbf{0}$.

Example 4.7 Consider again the formula $\varphi \stackrel{\text{def}}{=} x\overline{y} \vee \overline{x}\,\overline{z} \vee yz$ that was discussed in Exs. 4.3 and 4.5. Suppose that Formula$[f]$ is defined as in Eqn. (9), but that One$[f]$ and Zero$[f]$ are defined as in Defn. 4.6. The formula that would be created via Formula$[[\![\varphi]\!]]$ is

$$\text{Formula}[[\![\varphi]\!]] \equiv \begin{array}{ll} & x\overline{y} \vee \overline{x}\,\overline{z} \vee yz \\ \sqcup & \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}) \end{array}$$

(cf. Eqn. (12)). The semantically minimal variant of φ that would be created via Formula$[\langle\!\langle\varphi\rangle\!\rangle]$ is

$$\text{Formula}[\langle\!\langle\varphi\rangle\!\rangle] \equiv \begin{array}{ll} & \overline{y}\,\overline{z} \vee yz \vee \overline{x}\,\overline{z} \vee \overline{x}y \vee xz \vee x\overline{y} \\ \sqcup & \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}). \end{array} \tag{16}$$

(cf. Eqn. (13)). $\square$

Theorem 4.8 *[2]. Let f be a monotonic function in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$. Let* Formula$[f]$ *be the formula* One$[f] \sqcup$ Zero$[f]$*, where* One$[f]$ *and* Zero$[f]$ *are defined as in Eqns. (14) and (15), respectively. Then* Formula$[f]$ *realizes f (i.e., $[\![\text{Formula}[f]]\!] = f$).*
Proof: See [2]. $\square$

Henceforth, One$[\cdot]$ and Zero$[\cdot]$ mean the operations defined in Defn. 4.6 (Eqns. (14) and (15), respectively).

5. A BDD-Based Minimization Algorithm

This section presents an improved algorithm for semantic minimization. The worst-case running time of the algorithm is exponential in the size of φ; however, all operations can be implemented using BDDs.

The issues that we face in using the material that has been presented in Sects. 4.2 and 4.3 are

1. How do we efficiently represent the function $\langle\!\langle\varphi\rangle\!\rangle : \{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$?

2. To use Eqns. (14) and (15), how do we efficiently implement the indexing operations needed in the outermost connectives:

$$A : f(A) = 1 \text{ and } \forall A' \sqsupset A . f(A') = 1/2 \qquad (17)$$
$$A : f(A) = 0 \text{ and } \forall A' \sqsupset A . f(A') = 1/2. \qquad (18)$$

Our approach to issue 1 is to use BDDs [6]. Our approach to issue 2 is based on the following observation:

Observation 5.1 The Realization Theorem provides a way to construct a 3-valued propositional formula that realizes *any* given monotonic Boolean function. However, the realization problem that arises in the semantic-minimization problem does not require this general a method: In the semantic-minimization problem, the monotonic Boolean functions that arise are always ones that are in the range of $\langle\!\langle \cdot \rangle\!\rangle$; that is, we are only concerned with realization problems of the form Formula$[\langle\!\langle \varphi \rangle\!\rangle]$. □

Focusing on this special case of the realization problem allows us to sidestep issue 2 by implementing One$[\cdot]$ and Zero$[\cdot]$ differently from the way they are stated in Eqns. (14) and (15). The approach described takes advantage of the BDD-based representation used to address issue 1.

5.1. Representing the Supervaluational Semantics

Given φ, our goal is to find a semantically minimal variant ψ. The truth-valuational semantics of ψ must be equal to the supervaluational semantics of φ:

$$[\![\psi]\!](A) = \langle\!\langle \varphi \rangle\!\rangle(A) = \bigsqcup_{a \text{ rep. by } A} [\![\varphi]\!](a).$$

Thus, in order to capture $\langle\!\langle \varphi \rangle\!\rangle$, we need only concern ourselves with the truth-functional semantics of φ on definite assignments—i.e., just a portion of the truth-functional semantics of φ. In other words, rather than considering $[\![\varphi]\!]$ and $\langle\!\langle \varphi \rangle\!\rangle$ as functions in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$, we need only consider them as functions in $\{0, 1\}^n \to \{0, 1, 1/2\}$. (Functions of the latter type are called **Boolean functions with don't cares** or **incompletely specified Boolean functions**.)

An incompletely specified Boolean function f can be represented via a pair of **total Boolean functions** (i.e., functions in $\{0, 1\}^n \to \{0, 1\}$), denoted by $(\lfloor f \rfloor, \lceil f \rceil)$, where $\lfloor f \rfloor$ conflates 0 and $1/2$, and $\lceil f \rceil$ conflates 1 and $1/2$:[4]

$$\lfloor f \rfloor(a) = \begin{cases} 1 & \text{if } f(a) = 1 \\ 0 & \text{if } f(a) \sqsupseteq 0 \end{cases} \qquad \lceil f \rceil(a) = \begin{cases} 1 & \text{if } f(a) \sqsupseteq 1 \\ 0 & \text{if } f(a) = 0 \end{cases}$$

[4] Thus, there are three types of Boolean functions that play a role in this paper:

(3-valued) Boolean functions:	$\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$
incompletely specified Boolean functions:	$\{0, 1\}^n \to \{0, 1, 1/2\}$
total Boolean functions:	$\{0, 1\}^n \to \{0, 1\}$

We do not introduce any special notation to distinguish among functions of the three types (although terms of the form $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ always denote total Boolean functions). In particular, occurrences of $[\![\varphi]\!]$ and $\langle\!\langle \varphi \rangle\!\rangle$ sometimes denote incompletely specified Boolean functions (which may be total Boolean functions), but sometimes denote 3-valued Boolean functions. However, it should always be clear from context which use is intended.

$\lfloor f \rfloor$ and $\lceil f \rceil$ can each be represented using ordinary BDDs [6], as proposed by Minato et al. [16].

To capture $\langle\!\langle \varphi \rangle\!\rangle$ (as a function from $\{0, 1\}^n \to \{0, 1, 1/2\}$), it will be represented as the pair $(\lfloor\langle\!\langle \varphi \rangle\!\rangle\rfloor, \lceil\langle\!\langle \varphi \rangle\!\rangle\rceil)$, where $\lfloor\langle\!\langle \varphi \rangle\!\rangle\rfloor$ and $\lceil\langle\!\langle \varphi \rangle\!\rangle\rceil$ are both functions in $\{0, 1\}^n \to \{0, 1\}$. Given the formula φ (over the propositional variables $x_1, \ldots, x_n$), this can be accomplished by traversing φ, applying the following translation rules bottom-up:

$$\begin{aligned}
\mathbf{0} &\longrightarrow (\lambda a.0, \lambda a.0) \\
\mathbf{1} &\longrightarrow (\lambda a.1, \lambda a.1) \\
\mathbf{1/2} &\longrightarrow (\lambda a.0, \lambda a.1) \\
x_i &\longrightarrow (\lambda a.a(x_i), \lambda a.a(x_i)) \\
\neg(\lfloor f \rfloor, \lceil f \rceil) &\longrightarrow (\neg\lceil f \rceil, \neg\lfloor f \rfloor) \\
(\lfloor f_1 \rfloor, \lceil f_1 \rceil) \wedge (\lfloor f_2 \rfloor, \lceil f_2 \rceil) &\longrightarrow (\lfloor f_1 \rfloor \wedge \lfloor f_2 \rfloor, \lceil f_1 \rceil \wedge \lceil f_2 \rceil) \\
(\lfloor f_1 \rfloor, \lceil f_1 \rceil) \vee (\lfloor f_2 \rfloor, \lceil f_2 \rceil) &\longrightarrow (\lfloor f_1 \rfloor \vee \lfloor f_2 \rfloor, \lceil f_1 \rceil \vee \lceil f_2 \rceil) \\
(\lfloor f_1 \rfloor, \lceil f_1 \rceil) \sqcup (\lfloor f_2 \rfloor, \lceil f_2 \rceil) &\longrightarrow (\lfloor f_1 \rfloor \wedge \lfloor f_2 \rfloor, \lceil f_1 \rceil \vee \lceil f_2 \rceil) \\
(\lfloor f_1 \rfloor, \lceil f_1 \rceil) \,?\, (\lfloor f_2 \rfloor, \lceil f_2 \rceil) : (\lfloor f_3 \rfloor, \lceil f_3 \rceil) &\longrightarrow \\
&\left(\begin{array}{c} (\lfloor f_1 \rfloor \,?\, \lfloor f_2 \rfloor : \lfloor f_3 \rfloor) \wedge (\lceil f_1 \rceil \,?\, \lfloor f_2 \rfloor : \lfloor f_3 \rfloor), \\ (\lceil f_1 \rceil \,?\, \lceil f_2 \rceil : \lceil f_3 \rceil) \wedge (\lfloor f_1 \rfloor \,?\, \lceil f_2 \rceil : \lceil f_3 \rceil) \end{array} \right)
\end{aligned} \qquad (19)$$

All of the operations on total Boolean functions that are required on the right-hand sides of the above rules are ones from the standard repertoire of BDD operations: the creation of BDDs for $\lambda a.0$, $\lambda a.1$, and $\lambda a.a(x_i)$ (for $1 \le i \le n$), and the application of the following Boolean operations to existing BDDs: $\neg$, $\wedge$, $\vee$, and $(\cdot \,?\, \cdot : \cdot)$ (also known as ITE [5]).

5.2. Realization for Semantic Minimization

Definition 5.2 *(Cf. [17, 8].) A superscripted propositional variable x^b, where $b \in \{0, 1\}$, stands for a literal:*

$$x^b \stackrel{\text{def}}{=} \begin{cases} x & \text{if } b = 1 \\ \neg x & \text{if } b = 0 \end{cases}$$

Let f be a total Boolean function over the propositional variables $x_1, \ldots, x_n$. A conjunction of literals in which each propositional variable appears at most once—either negated or unnegated—i.e.,

$$\bigwedge_{i \in S \subseteq \{1, \ldots, n\}} x_i^{b_i} \qquad (20)$$

is an **implicant** *of f if, for every 2-valued assignment a such that $a(x_i) = b_i$ for all $i \in S$, we have $f(a) = 1$.*

Each conjunction of literals of the form shown in (20) can be thought of as the set of literals $\{x_i^{b_i} \mid i \in S\}$. An implicant is a **prime implicant** *if none of its proper subsets is an implicant (i.e., corresponds to a conjunction of literals that is an implicant).* □

Example 5.3 xy, xz, and xyz are all implicants of $xz \vee y\overline{z}$. x, y, and z are not implicants of $xz \vee y\overline{z}$; hence, xy and xz are prime implicants. □

Our concern is with realization problems of the form Formula$[f]$, where $f = \langle\!\langle \varphi \rangle\!\rangle$. We now show that for this case, realization can be can be implemented as follows:

$$\text{Formula}[f] \equiv \text{Primes}[\lfloor f \rfloor] \sqcup \neg\text{Primes}[\neg\lceil f \rceil],$$

where Primes$[g]$ is the operation that, given a total Boolean function g, creates the disjunction of g's prime implicants:

$$\text{Primes}[g] \stackrel{\text{def}}{=} \bigvee_{\pi \text{ a prime implicant of } g} \pi.$$

Lemma 5.4 *Let A be a 3-valued assignment such that (i) $\langle\!\langle \varphi \rangle\!\rangle(A) = 1$, and (ii) for all $A' \sqsupset A$, $\langle\!\langle \varphi \rangle\!\rangle(A') = 1/2$. Then the formula*

$$\pi \stackrel{\text{def}}{=} \bigwedge_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{One}(\langle\!\langle \varphi \rangle\!\rangle, A, i)$$

is a prime implicant of $\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor$.
Proof: See App. A. □

Lemma 5.5 *Let π by a prime implicant of $\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor$. Then there is a 3-valued assignment A such that*

(i) $\langle\!\langle \varphi \rangle\!\rangle(A) = 1$

(ii) For all $A' \sqsupset A$, $\langle\!\langle \varphi \rangle\!\rangle(A') = 1/2$

(iii) $\pi \equiv \bigwedge_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{One}(\langle\!\langle \varphi \rangle\!\rangle, A, i)$

Proof: See App. A. □

These results yield the following procedure for semantic minimization:

Theorem 5.6 *Let ψ be* Primes$[\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor] \sqcup \neg\text{Primes}[\neg\lceil \langle\!\langle \varphi \rangle\!\rangle \rceil]$. *Then ψ is a semantically minimal variant of φ.*
Proof: From Defn. 4.6 (i.e., Eqn. (14)), and Lemmas 5.4 and 5.5, it follows that One$[\langle\!\langle \varphi \rangle\!\rangle] \equiv$ Primes$[\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor]$. $\neg\langle\!\langle \varphi \rangle\!\rangle$ is represented by both $(\lfloor \neg\langle\!\langle \varphi \rangle\!\rangle \rfloor, \lceil \neg\langle\!\langle \varphi \rangle\!\rangle \rceil)$ and $\neg(\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor, \lceil \langle\!\langle \varphi \rangle\!\rangle \rceil)$; by translation method (19), the latter equals $(\neg\lceil \langle\!\langle \varphi \rangle\!\rangle \rceil, \neg\lfloor \langle\!\langle \varphi \rangle\!\rangle \rfloor)$. This implies that $\lfloor \neg\langle\!\langle \varphi \rangle\!\rangle \rfloor = \neg\lceil \langle\!\langle \varphi \rangle\!\rangle \rceil$, and hence

$$\begin{aligned} \text{Zero}[\langle\!\langle \varphi \rangle\!\rangle] &\equiv_{\text{DM}} \neg\text{One}[\neg\langle\!\langle \varphi \rangle\!\rangle] \\ &\equiv \neg\text{Primes}[\lfloor \neg\langle\!\langle \varphi \rangle\!\rangle \rfloor] \\ &\equiv \neg\text{Primes}[\neg\lceil \langle\!\langle \varphi \rangle\!\rangle \rceil]. \end{aligned}$$

The claim now follows from Thms. 4.8 and 4.4. □

```
[1]  formula MinimizeFormula(formula φ) {
[2]    Transform φ to (⌊⟨⟨φ⟩⟩⌋, ⌈⟨⟨φ⟩⟩⌉) using
                 translation method (19)
[3]    return Primes[⌊⟨⟨φ⟩⟩⌋] ⊔ ¬Primes[¬⌈⟨⟨φ⟩⟩⌉]
[4]  }
```

Figure 2. A minimization algorithm.

Thm. 5.6 provides the justification for the function MinimizeFormula, shown in Fig. 2; given a propositional formula φ as input, MinimizeFormula creates and returns a semantically minimal variant of φ. MinimizeFormula uses the auxiliary procedure Primes$[f]$, which creates a sum-of-prime-implicants formula for a given formula f. Any of several known methods for efficiently generating prime implicants can be used for this step [9, 8, 21]. (These methods all start from the BDD representation of f; thus, when line [2] is implemented with BDDs, exactly the right kind of input structure is at hand.)

Let us return again to the formula $\varphi \stackrel{\text{def}}{=} x\overline{y} \vee \overline{x}\,\overline{z} \vee yz$ (cf. Exs. 4.3, 4.5, and 4.7). MinimizeFormula would create the formula that we saw in Eqn. (16) of Ex. 4.7—although MinimizeFormula would arrive at the answer by a different, and more efficient, method:

$$\text{Formula}[\langle\!\langle \varphi \rangle\!\rangle] \equiv \begin{array}{l} \overline{y}\,\overline{z} \vee yz \vee \overline{x}\,\overline{z} \vee \overline{x}y \vee xz \vee x\overline{y} \\ \sqcup\ \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}). \end{array} \quad (16)$$

A drawback of MinimizeFormula is the need to generate all prime implicants. One might try to substitute other sum-of-products expressions that can be used to represent a given function (in 2-valued logic), such as an irredundant prime cover [15, 10]. This approach is not tenable, however; for instance, for the formula $\varphi \stackrel{\text{def}}{=} x\overline{y} \vee \overline{x}\,\overline{z} \vee yz$, if we substitute the irredundant-prime-cover algorithm from [10] for the two calls on Primes$[\cdot]$ in line [3] of MinimizeFormula, we would get the following formula:

$$\overline{y}\,\overline{z} \vee yz \vee \overline{x}\,\overline{z} \vee xz \sqcup \neg(\overline{x}\,\overline{y}z \vee xy\overline{z}). \quad (21)$$

However, formula (21) is not a semantically minimal variant of φ: for the assignment $[x \mapsto 0, y \mapsto 1, z \mapsto 1/2]$, Eqn. (16) evaluates to 1, whereas formula (21) evaluates to $1/2$. Moreover, formula (21) is actually *worse* than φ itself (and Eqn. (16)) for the assignment $[x \mapsto 1, y \mapsto 0, z \mapsto 1/2]$: φ and Eqn. (16) evaluate to 1, whereas formula (21) evaluates to $1/2$.

5.3. Other Semantically Minimal Formulae

In this section, we derive some other forms, different from the one that was the subject of Thm. 5.6, in which one can express a semantically minimal formula. The proof of the following theorem can be found in App. A:

Theorem 5.7 *If f is a total Boolean function, then* $[\![\text{Primes}[f]]\!] = [\![\neg\text{Primes}[\neg f]]\!]$. □

In the proof of Thm. 5.6, we showed

$$\begin{aligned} \text{One}[\langle\!\langle\varphi\rangle\!\rangle] &\equiv \quad \text{Primes}[\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor] \\ \text{Zero}[\langle\!\langle\varphi\rangle\!\rangle] &\equiv_{\text{DM}} \neg\text{Primes}[\neg\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]. \end{aligned}$$

These imply

$$[\![\text{One}[\langle\!\langle\varphi\rangle\!\rangle]]\!] = [\![\text{Primes}[\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]]\!] \tag{22}$$

$$[\![\text{Zero}[\langle\!\langle\varphi\rangle\!\rangle]]\!] = [\![\neg\text{Primes}[\neg\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]]\!]. \tag{23}$$

Applying Thm. 5.7 results in two new relationships:

$$[\![\text{One}[\langle\!\langle\varphi\rangle\!\rangle]]\!] = [\![\neg\text{Primes}[\neg\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]]\!] \tag{24}$$

$$[\![\text{Zero}[\langle\!\langle\varphi\rangle\!\rangle]]\!] = [\![\text{Primes}[\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]]\!]. \tag{25}$$

Consequently, we can "mix and match" Eqns. (22), (23), (24), and (25) to create expressions that yield semantically minimal formulae different from the one given in Thm. 5.6. That is, the function MinimizeFormula of Fig. 2 creates a semantically minimal variant of φ with any of the following four expressions used in line [3]:

		Zero	
		sum-of-products	¬sum-of-products
One	sum-of-products	$\text{Primes}[\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]$ ⊔ $\text{Primes}[\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]$	$\text{Primes}[\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]$ ⊔ $\neg\text{Primes}[\neg\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]$
	¬sum-of-products	$\neg\text{Primes}[\neg\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]$ ⊔ $\text{Primes}[\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]$	$\neg\text{Primes}[\neg\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor]$ ⊔ $\neg\text{Primes}[\neg\lceil\langle\!\langle\varphi\rangle\!\rangle\rceil]$

A term in ¬sum-of-products form can be put in product-of-sums form (with no blow-up in size) by applying De Morgan's laws. Thus, we may create a semantically minimal formula with any combination of sum-of-products and product-of-sums terms that we desire.

Our final result provides a condition under which we may generate a semantically minimal variant that does not contain an occurrence of ⊔:

Corollary 5.8 *Suppose that φ is a formula such that $\langle\!\langle\varphi\rangle\!\rangle$ is a total Boolean function. Let $\psi_1 \stackrel{\text{def}}{=} \text{Primes}[\langle\!\langle\varphi\rangle\!\rangle]$ and $\psi_2 \stackrel{\text{def}}{=} \neg\text{Primes}[\neg\langle\!\langle\varphi\rangle\!\rangle]$. Then ψ_1 and ψ_2 are both semantically minimal variants of φ.*

Proof: When $\langle\!\langle\varphi\rangle\!\rangle$ is a total Boolean function, $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor = \lceil\langle\!\langle\varphi\rangle\!\rangle\rceil = \langle\!\langle\varphi\rangle\!\rangle$. The result follows from Eqns. (22), (23), (24), and (25)—and the elimination of duplicate terms in the diagonal entries of the table given above. □

In particular, any formula that does not contain an explicit occurrence of $\mathbf{1/2}$ or ⊔ has a semantically minimal variant that does not contain an occurrence of ⊔.

Example 5.9 Consider the formula $\varphi \stackrel{\text{def}}{=} xy \vee \overline{x}z$. In 2-valued logic, φ can be treated as a syntactic shorthand for $x\,?\,y : z$. In Sect. 2.2, we discussed why φ is not a suitable syntactic shorthand for (the extension of) $x\,?\,y : z$ to 3-valued logic, and why $xy \vee \overline{x}z \vee yz$ is a suitable shorthand.

We can now derive this by means of our results on semantic minimization: $xy \vee \overline{x}z$ does not contain an explicit occurrence of $\mathbf{1/2}$ or ⊔, and thus $\langle\!\langle xy \vee \overline{x}z\rangle\!\rangle$ is a total Boolean function. Because $\text{Primes}[\langle\!\langle xy \vee \overline{x}z\rangle\!\rangle] = xy \vee \overline{x}z \vee yz$, $xy \vee \overline{x}z \vee yz$ is a semantically minimal variant of $xy \vee \overline{x}z$. □

6. Related Work

There is a substantial body of work that addresses methods for syntactic minimization of propositional formula. Previous work has addressed finding minimal-size sum-of-products formulae [17], as well as minimal-size formulae for other forms [22]. In contrast, this paper concerns semantic minimization (in 3-valued propositional logic). Because the minimization criterion is a semantic one, rather than a syntactic one, the formula ψ that results is not necessarily smaller than φ.

The realization problem, and the two versions of the Realization Theorem that we have used, are due to Blamey [2, 3, 1]. However, Blamey's work did not address the semantic-minimization problem that we defined in Sect. 3. In Sect. 5, we focused on a special case of the realization problem, which allowed us to define a semantic-minimization algorithm (MinimizeFormula) that is more efficient than what one would have using the general realization constructions given by Blamey.

Our motivation for investigating the semantic-minimization problem for propositional logic was as a heuristic for creating "better" formulae in a 3-valued first-order logic (with a transitive-closure operator). This logic is the basis of a system for "shape analysis" (i.e., the determination of information about the heap-allocated data structures that a program manipulates), and, more broadly, for a variety of static analyses of programs that manipulate heap-allocated data structures [18, 19, 14]. By replacing a formula φ with a formula ψ, we may improve the precision of the answers that the system obtains. We have implemented MinimizeFormula, and have used it as a subroutine in a heuristic method for minimizing first-order formulae; the method works on a formula bottom-up, applying MinimizeFormula to the body of each non-propositional operator (i.e., each quantifier or transitive-closure operator).

Acknowledgments

We thank S. Blamey for providing us with a copy of [2] and with a preliminary version of [1].

References

[1] S. Blamey. Partial logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd. Ed., Volume 5*. Kluwer Acad. In preparation.

[2] S. Blamey. *Partial-Valued Logic*. PhD thesis, University of Oxford, Oxford, England, 1980.

[3] S. Blamey. Partial logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume 3: Alternatives To Classical Logic*, pages 1–70. 1986.

[4] R. Bodik, R. Gupta, and M. L. Soffa. Complete removal of redundant computations. In *SIGPLAN Conf. on Prog. Lang. Design and Impl.*, pages 1–14, 1998.

[5] K. Brace, R. Rudell, and R. Bryant. Effi cient implementation of a BDD package. In *Proc. of the 27th ACM/IEEE Design Automation Conf.*, pages 40–45, 1990.

[6] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, C-35(6):677–691, Aug. 1986.

[7] C.-T. Chou. The mathematical foundation of symbolic trajectory evaluation. In *Proc. Computer-Aided Verif.* Springer-Verlag, July 1999.

[8] O. Coudert. Two-level logic minimization: An overview. *Integration, The VLSI Journal*, 17(2):97–140, Oct. 1994.

[9] O. Coudert and J. Madre. A new graph based prime computation technique. In T. Sasao, editor, *Logic Synthesis and Optimization*, pages 33–57. Kluwer Acad., Norwell, MA, 1993.

[10] O. Coudert, J. Madre, H. Fraisse, and H. Touati. Implicit prime cover computation: An overview. In *Proc. Workshop on Synth. and Syst. Int. of Mixed Tech.*, Oct. 1993.

[11] M. Ginsberg. Multivalued logics: A uniform approach to inference in artifi cial intelligence. *Comp. Intell.*, 4:265–316, 1988.

[12] A. Jain. *Formal Verifi cation by Symbolic Trajectory Evaluation*. PhD thesis, Carnegie Mellon Univ., Pittsburgh, PA, July 1997.

[13] S. Kleene. *Introduction to Metamathematics*. North-Holland, second edition, 1987.

[14] T. Lev-Ami and M. Sagiv. TVLA: A system for implementing static analyses. In *Static Analysis Symp.*, pages 280–301, 2000.

[15] S. Minato. Fast generation of irredundant sum-of-products forms from binary decision diagrams. In *Proc. Workshop on Synth. and Syst. Int. of Mixed Tech.*, pages 64–73, Apr. 1992.

[16] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagram with attributed edges for effi cient boolean function manipulation. In *Design Automation Conf.*, pages 52–57, New York, 1990. ACM.

[17] W. Quine. The problem of simplifying truth functions. *Amer. Math. Monthly*, 59(8):521–531, Oct. 1952.

[18] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. In *Symp. on Princ. of Prog. Lang.*, pages 105–118, New York, NY, Jan. 1999. ACM Press.

[19] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *Trans. on Prog. Lang. and Syst.*, 2002. To appear.

[20] S. Sagiv, N. Francez, M. Rodeh, and R. Wilhelm. A logic-based approach to data fbw analysis problems. *Acta Inf.*, 35(6):457–504, June 1998.

[21] T. Sasao. Ternary decision diagrams and their applications. In Sasao and Fujita [22], pages 269–292.

[22] T. Sasao and M. Fujita, editors. *Representations of Discrete Functions*. Kluwer Acad., Norwell, MA, 1996.

[23] C.-J. Seger and R. Bryant. Formal verifi cation by symbolic evaluation of partially-ordered trajectories. *Formal Methods in System Design*, 6(2):147–189, Mar. 1995.

[24] B. van Fraassen. Singular terms, truth-value gaps, and free logic. *J. Phil.*, 63(17):481–495, Sept. 1966.

A. Proofs

Lemma 5.4 *Let A be a 3-valued assignment such that (i) $\langle\!\langle\varphi\rangle\!\rangle(A) = 1$, and (ii) for all $A' \sqsupset A$, $\langle\!\langle\varphi\rangle\!\rangle(A') = 1/2$. Then the formula*

$$\pi \stackrel{\text{def}}{=} \bigwedge_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{One}(\langle\!\langle\varphi\rangle\!\rangle, A, i)$$

is a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$.

Proof: By Defn. 4.1 (Eqn. (3)), and the presence of the guard "$A(x_i) \sqsubset 1/2$" in the index of the conjunction, π is a conjunction of literals of the form shown in (20), namely the conjunction

$$\pi \equiv \bigwedge_{A(x_i) \sqsubset 1/2} x_i^{A(x_i)}. \tag{26}$$

By assumption (i), $\langle\!\langle\varphi\rangle\!\rangle(A) = 1$. Defn. 3.1 implies that $\langle\!\langle\varphi\rangle\!\rangle$ is a monotonic function in $\{0, 1, 1/2\}^n \to \{0, 1, 1/2\}$. Thus, for all a rep. by A, $\langle\!\langle\varphi\rangle\!\rangle(a) \sqsubseteq \langle\!\langle\varphi\rangle\!\rangle(A) = 1$, which, by the defi nition of $\lfloor\cdot\rfloor$, implies that $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor(a) = 1$. Because $a(x_i) = A(x_i)$ for all x_i for which $A(x_i) \sqsubset 1/2$, π is an implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$.

To see that π is a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$, consider the right-hand side of Eqn. (26) to be the set of literals

$$S_\pi = \{x_i^{A(x_i)} \mid A(x_i) \sqsubset 1/2\}.$$

For any S that is a strict subset of S_π, we would have the set of literals

$$S = \{x_i^{A'(x_i)} \mid A'(x_i) \sqsubset 1/2\}$$

corresponding to an assignment A', where $A' \sqsupset A$. However, by assumption (ii), $\langle\!\langle\varphi\rangle\!\rangle(A') = 1/2$, which means, by Defn. 3.1, that

$$\{\langle\!\langle\varphi\rangle\!\rangle(a) \mid a \text{ rep. by } A'\} = \{0, 1\}.$$

Therefore, there is a 2-valued assignment a_0 such that $a_0(x_i) = A'(x_i)$ for all x_i for which $A'(x_i) \sqsubset 1/2$, and $\langle\!\langle\varphi\rangle\!\rangle(a_0) = 0$, and hence $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor(a_0) = 0$. This means that S, or more precisely,

$$\bigwedge_{A'(x_i) \sqsubset 1/2} x_i^{A'(x_i)}$$

is not an implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$.

Consequently, π is a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$. □

Lemma 5.5 *Let π by a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$. Then there is a 3-valued assignment A such that*

(i) $\langle\!\langle\varphi\rangle\!\rangle(A) = 1$

(ii) For all $A' \sqsupset A$, $\langle\!\langle\varphi\rangle\!\rangle(A') = 1/2$

(iii) $\pi \equiv \bigwedge_{\substack{1 \le i \le n \text{ and} \\ A(x_i) \sqsubset 1/2}} \text{One}(\langle\!\langle\varphi\rangle\!\rangle, A, i)$

Proof: Let π be the product

$$\pi \stackrel{\text{def}}{=} \bigwedge_{i \in S \subseteq \{1,\ldots,n\}} x_i^{b_i},$$

and let A_S be the assignment

$$A_S = \left[x_i \mapsto \begin{cases} b_i & \text{if } i \in S \\ 1/2 & \text{otherwise} \end{cases} \right]$$

(i) Because π is a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$, for all a represented by A_S, $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor(a) = 1$. Therefore,

$$\begin{aligned}\langle\!\langle\varphi\rangle\!\rangle(a) &= 1 \qquad \text{(by the definition of } \lfloor\cdot\rfloor)\\ [\![\varphi]\!](a) &= 1 \qquad \text{(by Defn. 3.1)}\\ \langle\!\langle\varphi\rangle\!\rangle(A_S) &= 1 \qquad \text{(by Defn. 3.1)}\end{aligned}$$

(ii) If $A_S = [x_i \mapsto 1/2 \mid 1 \le i \le n]$ (and hence π is the formula **1**), then property (ii) holds vacuously. Thus, we may assume that A_S binds at least one x_i to a definite value.

Let A' be an assignment such that $A' \sqsupset A_S$. A' and A_S agree on some (possibly empty) set $\{x_i \mid i \in S'\}$, for some $S' \subset S$. Let S'_π be $\{x_i^{b_i} \mid A'(x_i) = b_i, i \in S'\}$. By our assumptions, S'_π is not an implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$. Thus, there is a 2-valued assignment a_0 such that $a_0(x_i) = A'(x_i)$ for all $i \in S'$, and $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor(a_0) = 0$; consequently, by the definition of $\lfloor\cdot\rfloor$, $\langle\!\langle\varphi\rangle\!\rangle(a_0) \sqsupseteq 0$.

Pick any a_1 represented by A_S. Because π is a prime implicant of $\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor$, we have

$$\begin{aligned}\lfloor\langle\!\langle\varphi\rangle\!\rangle\rfloor(a_1) &= 1 \qquad \text{(by Defn. 5.2)}\\ \langle\!\langle\varphi\rangle\!\rangle(a_1) &= 1 \qquad \text{(by the definition of } \lfloor\cdot\rfloor)\\ [\![\varphi]\!](a_1) &= 1 \qquad \text{(by Defn. 3.1)}\end{aligned}$$

Because $A' \sqsupset A_S$, a_1 is also represented by A'. Thus,

$$\begin{aligned}\langle\!\langle\varphi\rangle\!\rangle(A') &= \bigsqcup_{a \text{ rep. by } A'} [\![\varphi]\!](a)\\ &\sqsupseteq [\![\varphi]\!](a_0) \sqcup [\![\varphi]\!](a_1)\\ &\sqsupseteq 0 \sqcup 1\\ &= 1/2\end{aligned}$$

(iii) π can be rewritten as

$$\bigwedge_{\substack{1 \le i \le n \text{ and}\\ A_S(x_i) \sqsubset 1/2}} x_i^{A_S(x_i)}. \tag{27}$$

Because we showed in (i) that $\langle\!\langle\varphi\rangle\!\rangle(A_S) = 1$, formula (27) can, in turn, be expressed as

$$\bigwedge_{\substack{1 \le i \le n \text{ and}\\ A_S(x_i) \sqsubset 1/2}} \text{One}(\langle\!\langle\varphi\rangle\!\rangle, A_S, i).$$

□

Lemma A.1 *If f is a total Boolean function, then* $[\![\text{Primes}[f]]\!] = \langle\!\langle\text{Primes}[f]\rangle\!\rangle$.

Proof: Let A be a 3-valued assignment. By Defn. 3.1 and the monotonicity of $[\![\cdot]\!]$ (Lem. 2.4), if $[\![\text{Primes}[f]]\!](A)$ yields a definite value d, then $\langle\!\langle\text{Primes}[f]\rangle\!\rangle(A)$ must also yield d.

Thus, we must only consider the case in which

$$[\![\text{Primes}[f]]\!](A) = 1/2. \tag{28}$$

To show that $\langle\!\langle\text{Primes}[f]\rangle\!\rangle(A) = 1/2$, we need to show that there exist definite assignments (i) a_1 rep. by A, such that $[\![\text{Primes}[f]]\!](a_1) = 1$, and (ii) a_0 rep. by A, such that $[\![\text{Primes}[f]]\!](a_0) = 0$.

(i) Pick any disjunct of Primes$[f]$ that evaluates to $1/2$ under assignment A, say $\pi_S \stackrel{\text{def}}{=} \bigwedge_{i \in S \subseteq \{1,\ldots,n\}} x_i^{b_i}$. Consider the variables x_j such that $j \in S$ and $A(x_j) = 1/2$. Create a_1 from A by replacing each binding $x_j \mapsto 1/2$ with $x_j \mapsto b_j$. Because $[\![\pi_S]\!](a_1) = 1$, we have $[\![\text{Primes}[f]]\!](a_1) = 1$.

(ii) Suppose for the sake of argument that,

$$\text{for all } a \text{ rep. by } A, [\![\text{Primes}[f]]\!](a) = 1. \tag{29}$$

By Defn. 5.2, this implies that for all a represented by A, $f(a) = 1$, which means that the formula

$$\pi_A \stackrel{\text{def}}{=} \bigwedge_{A(x_i) \sqsubset 1/2} x_i^{A(x_i)}$$

is an implicant of f. Consequently, Primes$[f]$ contains a disjunct π such that π (considered as a set of literals) is a subset of π_A (considered as a set of literals). Therefore, $[\![\pi]\!](A) = 1$, which means that $[\![\text{Primes}[f]]\!](A) = 1$. However, this contradicts assumption (28), and hence our subsequent assumption, assumption (29), must be incorrect.

Because f is a total Boolean function, there cannot be any definite assignment a such that $[\![\text{Primes}[f]]\!](a) = 1/2$. Thus, the fact that assumption (29) is incorrect implies that there must exist an a_0 rep. by A such that $[\![\text{Primes}[f]]\!](a_0) = 0$.

□

Lemma A.2 *If f is a total Boolean function, then for all definite assignments a,* $[\![\text{Primes}[f]]\!](a) = [\![\neg\text{Primes}[\neg f]]\!](a)$.

Proof:

$$\begin{aligned}[\![\text{Primes}[f]]\!](a) &= f(a) && \text{(follows from Defn. 5.2)}\\ &= 1 - (1 - f(a))\\ &= 1 - \overline{f(a)}\\ &= 1 - \overline{f}(a)\\ &= 1 - [\![\text{Primes}[\neg f]]\!](a) && \text{(follows from Defn. 5.2)}\\ &= [\![\neg\text{Primes}[\neg f]]\!](a)\end{aligned}$$

□

Theorem 5.7 *If f is a total Boolean function, then* $[\![\text{Primes}[f]]\!] = [\![\neg\text{Primes}[\neg f]]\!]$.

Proof: Let A be a 3-valued assignment.

$$\begin{aligned}[\![\text{Primes}[f]]\!](A) &= \langle\!\langle\text{Primes}[f]\rangle\!\rangle(A) && \text{(by Lem. A.1)}\\ &= \bigsqcup_{a \text{ rep. by } A} [\![\text{Primes}[f]]\!](a) && \text{(by Defn. 3.1)}\\ &= \bigsqcup_{a \text{ rep. by } A} [\![\neg\text{Primes}[\neg f]]\!](a) && \text{(by Lem. A.2)}\\ &= \bigsqcup_{a \text{ rep. by } A} (1 - [\![\text{Primes}[\neg f]]\!](a))\\ &= 1 - \bigsqcup_{a \text{ rep. by } A} [\![\text{Primes}[\neg f]]\!](a)\\ &= 1 - \langle\!\langle\text{Primes}[\neg f]\rangle\!\rangle(A) && \text{(by Defn. 3.1)}\\ &= 1 - [\![\text{Primes}[\neg f]]\!](A) && \text{(by Lem. A.1)}\\ &= [\![\neg\text{Primes}[\neg f]]\!](A)\end{aligned}$$

□