On Rewriting in Three-Valued Propositional Logic*

Olaf Owe
Department of Informatics
University of Oslo, 0316 Oslo, Norway
(email: olaf.owe@ifi.uio.no)

Abstract

Classical logic is based on two truth values. However, in computer science applications it is often desirable, or even necessary, to consider a third truth value, representing non-terminating computations, or computational errors. It is therefore of interest to investigate whether reasoning methods for standard propositional logic can be generalized to the case of three truth values ("three-valued logic"). In this paper we will discuss how rewrite systems for standard propositional logic can be adapted to the case of three-valued logic, considering various extensions of the standard logical connectives.

We argue that (inductive) completeness is more essential for practical application in the case of three truth values than in the standard case. For three-valued propositional logic with strict operators, we present a convergent and (inductively) complete system, modulo associativity and commutativity axioms, with a normal form similar to that of the Hsiang system. Unfortunately, convergent and complete rewriting in three-valued logic with Kleene operators requires more complex normal forms. By adding an operator we give a convergent and (inductively) complete system using Hsiang normal form of subterms.

1 Introduction

Propositional logic is an essential theory in computer science since it contains very fundamental concepts used in various computer applications. Propositional

*This research was supported in part by the Norwegian Research Council (Norges forskningsråd).

reasoning is also of practical interest, for instance within automated deduction or program reasoning, and several reasoning methods exist, including several decision procedures formulated as rewrite systems [11, 5, 9].

Standard propositional logic is based on two truth values. However, in computer science applications it is often desirable, or even necessary, to consider a third truth value, representing non-terminating computations, or computational errors [3, 6, 4]. Even if the source of such errors is due to non-logical types, they may propagate to the logical level (as in $1/0 > 1$ or in $\mathit{IsEmpty}(\mathit{pop}(\mathit{EmptyStack}))$). This gives rise to three-valued logic [10, 2, 3, 17, 12]. It is therefore of interest to investigate whether reasoning methods for standard propositional logic can be generalized to the case of three truth values. In this paper we discuss if and how rewrite systems for standard propositional calculus can be adapted to the case of three truth values, considering various extensions of the standard logical connectives.

The following extensions of the Boolean operators (given by truth tables) are considered interesting in computer applications:

- strict operators, as they reflect traditional call-by-value semantics

$\Rightarrow$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**T**	**T**	**U**
U	**U**	**U**	**U**

$\wedge$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**F**	**F**	**U**
U	**U**	**U**	**U**

$\vee$	**T**	**F**	**U**
T	**T**	**T**	**U**
F	**T**	**F**	**U**
U	**U**	**U**	**U**

- left-strict operators, as they are used in many programming languages

$\Rightarrow$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**T**	**T**	**T**
U	**U**	**U**	**U**

$\wedge$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**F**	**F**	**F**
U	**U**	**U**	**U**

$\vee$	**T**	**F**	**U**
T	**T**	**T**	**T**
F	**T**	**F**	**U**
U	**U**	**U**	**U**

(then left-strict **if** can be defined by $\textbf{if } x \textbf{ then } y \textbf{ else } z = (x \Rightarrow y) \wedge (x \vee z)$
)

- Kleene operators, which are useful for (program) specification and reasoning purposes

$\Rightarrow$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**T**	**T**	**T**
U	**T**	**U**	**U**

$\wedge$	**T**	**F**	**U**
T	**T**	**F**	**U**
F	**F**	**F**	**F**
U	**U**	**F**	**U**

$\vee$	**T**	**F**	**U**
T	**T**	**T**	**T**
F	**T**	**F**	**U**
U	**T**	**U**	**U**

where the standard truth values are denoted **T** and **F**, and **U** denotes the third truth value, "the undefined". All operators above have standard semantics when restricted to standard truth values; and are monotonic with respect to the flat ordering with **U** as the smallest element. Non-monotonic operators may be of interest for us as well, provided they can express the operators mentioned.

Notice that there is only one monotonic negation operator ($\neg$), only one monotonic equivalence operator ($\Leftrightarrow$), and only one monotonic exclusive or ($|$), all being strict. In all three three-valued semantics, $\neg x$ can be defined as $a \Rightarrow \mathbf{F}$, $x \Leftrightarrow y$ as $(x \Rightarrow y) \wedge (y \Rightarrow x)$, and $x \mid y$ as $\neg(x \Leftrightarrow y)$. Kleene operators are more expressive than left-strict ones[1], which again are more expressive than strict ones; thus a rewrite system for the more expressive operators is preferable!

2 Preliminaries

A rewrite system [11, 5] consists of a set of rules (of form $rhs \to lhs$) (and possibly a set of equations) and defines a congruence relation, denoted $=$, over the set of terms, letting $a = b$ mean that the terms a and b can be rewritten to the exact same term by means of (instantiations of) the rules (applied left to right) and the equations. If such rewriting always terminates the system is said to be *terminating*; and if the result of rewriting a term is unique (modulo the equations), the system is said to be *confluent*. A system with both properties is called *convergent*; if these properties hold for all ground (variable free) terms the system is *ground convergent*. For a convergent system $\mathcal{R}$, we let $\mathcal{R} \vdash p = q$ denote that rewriting of p and q results in the same term (modulo the $\mathcal{R}$-equations). The practical usefulness of reasoning by $\mathcal{R}$ depends on the complexity of decidability of congruence modulo the $\mathcal{R}$-equations. For instance, associative and commutative (AC) operators are often needed; and implementation of congruence over AC equations has been given much attention. Only AC equations will be considered in the systems below.

A rewrite system for standard propositional logic is here said to be *ground complete* if the only ground normal forms are $\mathbf{T}$ and $\mathbf{F}$. And a rewrite system for three-valued propositional logic is said to be *ground complete* if the only ground normal forms are $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{U}$. For instance, by reformulating truth tables as sets of ground rules (and adding rules reducing $\neg$, $\Leftrightarrow$ and $|$), the three above extensions of the boolean operators yield three ground complete and convergent rewrite systems, denoted $\mathcal{S}$, $\mathcal{L}$, and $\mathcal{K}$, defining strict, left-stict and Kleene semantics, respectively. The system $\mathcal{C}$, defining classical semantics, can be seen as the two-valued restriction of (one of) these systems. We will only consider systems that are ground complete and ground convergent, reflecting standard semantics when restricted to the standard truth values; thus a ground equation $p = q$ is valid iff p and q have the same normal form. An equation $p = q$ is *valid* in a system $\mathcal{R}$, denoted $\mathcal{R} \models p = q$, iff $p = q$ is valid in $\mathcal{R}$ for all truth value assignments to the (propositional) variables (including $\mathbf{U}$ in the case of three truth values). Similarly, a rule $p \to q$ is said to be valid iff $p = q$ is valid, and a formula p is said to be valid (or a tautology) iff $p = \mathbf{T}$ is valid. Notice that

[1] For instance, $x \wedge y$ with left-strict 'and' can be expressed as $x \wedge \neg x \vee x \wedge y$ with Kleene operators.

a ground convergent and ground complete system is sound in the sense that all rules and equations are valid, thus $\mathcal{R} \vdash p = q$ implies $\mathcal{R} \models p = q$. A convergent system is said to be (inductively) *complete* iff any two terms, say p and q, are reduced to the same (irreducible) normal form (modulo the AC equations) if the equation $p = q$ is valid, i.e. $\mathcal{R} \models p = q$ implies $\mathcal{R} \vdash p = q$.

In classical logic the validity of the equivalence $p \Leftrightarrow q$ is the same as that of the equation $p = q$; thus completeness for valid formulas suffices for proving such equations (using the standard reduction of equivalence). However, when we turn to three truth values, there are few interesting valid formulas with variables; and we do not in general have that validity of $p \Leftrightarrow q$ is the same as that of $p = q$. Thus in this case (unrestricted) completeness is in general needed to prove equations of the form $p = q$! Our focus will therefore be on convergent rewrite systems which are complete for three-valued propositional logic.

2.1 Preliminary discussion

We let x, y, z denote propositional variables, whereas p and q denote propositional formulas. We use the following precedence for the logical operators (most tight binding first): not $(\neg)$, and $(\wedge)$, or $(\vee)$, implication $(\Rightarrow)$, equivalence $(\Leftrightarrow)$, and exclusive or $(\mid)$. Consider the following classical rules:

$$\begin{array}{lll}
x \wedge y & \to & \neg(\neg x \vee \neg y) \\
x \vee y & \to & \neg(\neg x \wedge \neg y) \\
x \Rightarrow y & \to & \neg x \vee y) \\
x \Leftrightarrow y & \to & (x \Rightarrow y) \wedge (y \Rightarrow x) \\
x \mid y & \to & \neg(x \Leftrightarrow y) \\
\neg x & \to & \mathbf{T} \mid x
\end{array}$$

These laws (and Also de Morgans laws) are valid in $\mathcal{S}$, $\mathcal{L}$, and $\mathcal{K}$, and may be used to reduce the set of operators, as in classical logic.[2]

Rewrite systems based on reduction to disjunctive, or conjunctive, normal form, or reduction to if-form, may be convergent and complete for all propositional tautologies; but not in general for all formulas [13, 5]. However, the Hsiang system, based on exclusive or (xor) and 'and', is convergent (modulo associativity and commutativity axioms) and complete for all propositional formulas [8]. The same holds for the dual system, based on equivalence and 'or'. No other convergent and complete rewrite system for propositional logic is known.

The lack of completeness for rewriting to disjunctive normal form is due to the absence of unique minimal implicants, for instance,

$$\neg x \wedge y \vee \neg y \wedge z \vee \neg z \wedge x$$

[2] All three extensions of the and- and or-operators satisfy distributivity; and they are commutative, except for the left-strict ones, associative, idempotent, and have the standard identities ($\mathbf{T}$ and $\mathbf{F}$, respectively). In contrast to the others, the Kleene and- and or-operators satisfy $x \wedge \mathbf{F} = \mathbf{F}$ and $x \vee \mathbf{T} = \mathbf{T}$.

is equal to

$$x \wedge \neg y \vee y \wedge \neg z \vee z \wedge \neg x$$

By generalizing this to arbitrary many variables, one cannot capture such equalities by finite rewrite systems [15, 14]. This situation may at first seem better in the case of three truth values with Kleene operators, since the two terms above are not equal! (The assignment $x, y, z := \mathbf{U}, \mathbf{T}, \mathbf{F}$ gives $\mathbf{U}$ and $\mathbf{T}$, respectively.) However, other complications appear: we may simplify (by removing) the disjunct $x \wedge \neg x$ when added to implicants as above, i.e. the first disjunct may be removed in terms like

$$x \wedge \neg x \vee \neg x \wedge y \vee \neg y \wedge z \vee \neg z \wedge x$$

$$x \wedge \neg x \vee x \wedge \neg y \vee y \wedge \neg z \vee z \wedge \neg x$$

Such a reduction can only be done with rewrite rule(s) having at least as many variables as the implicant, which have no bound. (The more propositional variables, the more complex implicants can be constructed.) Therefore one cannot define a fixed set of convergent rules. A similar discussion applies to conjunctive normal form. Thus there is no convergent and complete rewrite system for Kleene operators based on disjunctive or conjunctive normal forms.

Reduction to if-form does not work in the case of three truth values with a left-strict if-operator, because “associativity” of **if** is lost (i.e. one may not lift out inner if-terms in then- and else-branches). And neither disjunctive normal form nor conjunctive normal form is suitable for the left-strict and- and or-operators, as they are not commutative; and again there is no minimal implicant. With a three-way non-monotonic case-construct [4][3], reduction to case-form is possible, as in

$$x \wedge y \rightarrow \ \textbf{case}\ x\ \textbf{of}\ \mathbf{T} : y \mid \mathbf{F} : \mathbf{F} \mid \mathbf{U} : (\textbf{case}\ y\ \textbf{of}\ \mathbf{T} : \mathbf{U} \mid \mathbf{F} : \mathbf{F} \mid \mathbf{U} : \mathbf{U})$$

but convergence would require an ad-hoc ordering of the nesting of the cases. A monotonic, two-way case-construct suffers from the same weaknesses as the if-operator.

We next turn to the Hsiang system and discuss its adaptation to the case of three truth values.

[3] The value of $\textbf{case}\ x\ \textbf{of}\ \mathbf{T} : a \mid \mathbf{F} : b \mid \mathbf{U} : c$ is a when x evaluates to $\mathbf{T}$, b when x evaluates to $\mathbf{F}$, and c when x evaluates to $\mathbf{U}$.

3 Adaptations of the Hsiang system

In addition to associativity and commutativity (AC) axioms for $\wedge$ and xor (denoted $\mid$), the Hsiang system $\mathcal{H}$ consists of the following rules:

$$\begin{array}{llcl}
R0 & x \wedge (y \mid z) & \rightarrow & x \wedge y \mid x \wedge z \\
R1 & x \wedge x & \rightarrow & x \\
R2 & \mathbf{T} \wedge x & \rightarrow & x \\
R3 & \mathbf{F} \wedge x & \rightarrow & \mathbf{F} \\
R4 & \mathbf{F} \mid x & \rightarrow & x \\
R5 & x \mid x & \rightarrow & \mathbf{F}
\end{array}$$

Other logical operators are handled by the rules

$$\begin{array}{llcl}
D1 & \neg x & \rightarrow & \mathbf{T} \mid x \\
D2 & x \vee y & \rightarrow & x \mid y \mid x \wedge y \\
D3 & x \Rightarrow y & \rightarrow & \mathbf{T} \mid x \mid x \wedge y \\
D4 & x \Leftrightarrow y & \rightarrow & \mathbf{T} \mid x \mid y
\end{array}$$

Any propositional formula is reduced to a unique normal form (modulo the AC axioms), one of $\mathbf{T}$, $\mathbf{F}$, p, or $\mathbf{T} \mid p$, where p is of form

$$... \mid (x_1 \wedge x_2 \wedge ...) \mid ...$$

with at least one disjunct, each being a conjunction of distinct propositional variables (not $\mathbf{T}$ or $\mathbf{F}$), and such that each two disjuncts contain different sets of propositional variables. For instance, the formula

$$(x \Rightarrow y) \mid (\neg x \Rightarrow y)$$

reduces to $\mathbf{T} \mid y$; and so does the formula $\neg y$, which proves the equation

$$(x \Rightarrow y) \mid (\neg x \Rightarrow y) \;\; = \;\; \neg y$$

Conversely (by completeness of the Hsiang system), we may conclude that the formula
$(x \Rightarrow y) \mid (\neg x \Rightarrow y)$ is not valid since it does not reduce to $\mathbf{T}$.

3.1 Generalization of the Hsiang system to three truth values

The Hsiang system is convergent and complete over the four truth values $\mathbf{T}$, $\mathbf{F}$, $\mathbf{U}$ and $\mathbf{T} \mid \mathbf{U}$![4] Unfortunately it is not sound for (any version of) three-valued

[4] This extension of Boolean does not seem useful for our purposes, as the and- and xor-operators are not expressive enough to define strict (or Kleene) operators.

logic: regardless of how the Boolean operators are extended, some of the Hsiang equations become invalid. In order to prove this, we observe that the rules R1 to R5 specialize to

$$\begin{array}{rcl}
\mathbf{T} \wedge \mathbf{U} & = & \mathbf{U} \\
\mathbf{F} \wedge \mathbf{U} & = & \mathbf{F} \\
\mathbf{U} \wedge \mathbf{U} & = & \mathbf{U} \\
\mathbf{F} \mid \mathbf{U} & = & \mathbf{U} \\
\mathbf{U} \mid \mathbf{U} & = & \mathbf{F}
\end{array}$$

which give a non-monotonic xor operator and Kleene semantics for $\wedge$ [10]. Thus these rules may be added to PROP without loss of confluence. It remains to explore $\mathbf{T} \mid \mathbf{U}$, which must equal one of $\mathbf{T}$, $\mathbf{F}$ or $\mathbf{U}$ in three-valued logics. Unfortunately, each of the equations

$$\begin{array}{rclc}
\mathbf{T} \mid \mathbf{U} & = & \mathbf{T} & (*) \\
\mathbf{T} \mid \mathbf{U} & = & \mathbf{F} & (**) \\
\mathbf{T} \mid \mathbf{U} & = & \mathbf{U} & (***)
\end{array}$$

invalidates the associativity axiom for xor. For instance, consider the term $(\mathbf{T} \mid \mathbf{T}) \mid \mathbf{U}$, which reduces to $\mathbf{U}$ in the Hsiang system. Only the last equation (***) is consistent with this (using associativity of xor). On the other hand, consider the term $\mathbf{T} \mid (\mathbf{U} \mid \mathbf{U})$, which reduces to $\mathbf{T}$ in the Hsiang system. The only equation consistent with this is the first (*).

We have seen that some of the Hsiang-rules are invalid in the case of three truth values. Even more, in order to accommodate Kleene-and, non-monotonic xor, and one of *, ** or ***, associativity of xor is lost.

Consider next the case of monotonic operators: For xor, monotonicity implies strictness, i.e. $\mathbf{U} \mid x = \mathbf{U}$, and thus it cannot satisfy R5 ($x \mid x \rightarrow \mathbf{F}$). The and operator may satisfy the rules R1 to R3, using Kleene semantics, but then distributivity (R0) will not be satisfied: the assignment $x, y, x := \mathbf{U}, \mathbf{T}, \mathbf{T}$ gives the equation

$$\mathbf{U} \wedge (\mathbf{T} \mid \mathbf{T}) = \mathbf{U} \wedge \mathbf{T} \mid \mathbf{U} \wedge \mathbf{T}$$

where the left hand side reduces to $\mathbf{F}$, whereas the right hand side reduces to $\mathbf{U}$ (by strictness of xor).

The AC axioms for $\wedge$ and $\mid$, together with distributivity, are crucial in the Hsiang system. In fact, these axioms taken together with rules R1, R2 and R3 (defining Kleene-and) are not confluent for any choice of (*), (**) or (***). Thus the combination of the Hsiang system and Kleene-and is not fruitful!

The combination of left-strict 'and' and strict xor would lead to problems similar to those of disjunctive normal form, for instance, $x \wedge y \mid y \wedge z \mid z \wedge x$ equals $x \wedge z \mid z \wedge y \mid y \wedge x$, but due to non-commutativity of and, there is no minimal normal form (when generalized to arbitrarily many variables).

The only remaining monotonic extension of $\wedge$ is strict and. Strictness of $\wedge$, the AC axioms (for xor) and distributivity imply that xor is strict. This case is discussed next:

3.2 Strict operators

In the case of three truth values and strict operators, distributivity is satisfied, as well as commutativity and associativity of $\mid$ and $\wedge$, but not the two Hsiang-equations $\mathbf{F} \wedge x \;=\; \mathbf{F}$ and $x \mid x \;=\; \mathbf{F}$, which both conflict with strictness. We present a revised rewrite system for this case, called $\mathcal{H}\ni$: In addition to AC axioms for $\wedge$ and $\mid$, it consists of the following rules:

$R0$	$x \wedge (y \mid z)$	$\to$	$x \wedge y \mid x \wedge z$
$R1$	$x \wedge x$	$\to$	x
$R2$	$\mathbf{T} \wedge x$	$\to$	x
$R4$	$\mathbf{F} \mid x$	$\to$	x
$R5'$	$x \mid x$	$\to$	$\mathbf{F} \wedge x$
$R6$	$\mathbf{F} \wedge x \mid \mathbf{F} \wedge y$	$\to$	$\mathbf{F} \wedge x \wedge y$
$R7$	$\mathbf{F} \wedge x \wedge y \mid x \wedge z$	$\to$	$\mathbf{F} \wedge y \mid x \wedge z$
$R8$	$\mathbf{F} \wedge x \mid x$	$\to$	x
$R9$	$\mathbf{F} \wedge x \mid x \wedge y$	$\to$	$x \wedge y$
$R10$	$\mathbf{F} \wedge x \wedge y \mid x$	$\to$	$\mathbf{F} \wedge y \mid x$
$R11$	$\mathbf{U} \wedge x$	$\to$	$\mathbf{U}$
$R12$	$\mathbf{U} \mid x$	$\to$	$\mathbf{U}$

Notice that rules R0, R1, R2, R4, and the AC axioms, are as in the Hsiang system, whereas rule R5' is a modification of R5. Rules R6 and R7 are weakened versions of R3, and rules R8, R9 and R10 represent simplified versions of rule R7 (where y and/or z are $\mathbf{T}$). The last two rules (and commutativity) express strictness and are referred to as the *strictness rules*. This system is obviously ground convergent and ground complete, defining the strict extensions of $\mid$ and $\wedge$. (Ground completeness follows by R1, R2, R4, R5', R11, R12 and the C axioms.)

Strict $\wedge$ and $\mid$ are expressive enough to define the strict extension of any two-valued Boolean operator. In particular, the strict extensions of $\neg$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$ are defined exactly as before (by D1-D4). And it may be useful to introduce a (strict) unary "definedness" operator, δ:

$$\delta x = x \vee \neg x$$

(which reduces to $\mathbf{T} \mid \mathbf{F} \wedge x$). The δ operator expresses (the strict version of) the well-definedness of a formula p, in the sense that δp is valid for all ground valuations making p defined, and $\mathbf{U}$ otherwise. Notice that δx is the same as $x \Leftrightarrow x$, but not $x = x$ (which is by definition always true).

Theorem 1. System $\mathcal{H}\ni$ (and D1-D4) is convergent, and also (inductively) complete for three-valued propositional logic with strict operators (i.e. two propositional formulas, p and q, are reduced to the same normal form modulo the AC axioms if and only if $p = q$ is valid).

Thus $\mathcal{H}\ni$ is also a decision procedure for three-valued propositional logic with strict operators — and even for equations $p = q$.

Proof outline: Termination modulo AC is quite obvious. (One may define an AC-simplification ordering, by means of natural numbers greater than 1, by taking $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{U}$ as 2, $\wedge$ as multiplication, $\mid$ as addition with one added to the result.)

Clearly the two strictness rules do not destroy convergence, since all other rules do not mention $\mathbf{U}$ explicitly, and have the same propositional variables on both sides of $\rightarrow$. The strictness rules can only be applied when reducing a term which already has an occurrence of $\mathbf{U}$, and will then result in $\mathbf{U}$. When rewriting a term without occurrences of $\mathbf{U}$ we may ignore the strictness rules. All the remaining rules of system $\mathcal{H}\ni$ may be derived in $\mathcal{H}$, and are thus valid in $\mathcal{H}$. In particular, a formula without occurrences of $\mathbf{U}$ can be reduced to its Hsiang normal form by first obtaining its $\mathcal{H}\ni$ normal form and then applying the Hsiang rules. This is explored in detail below:

Due to rule 6, it is clear that the $\mathcal{H}\ni$ normal form contains at most one disjunct with an occurrence of $\mathbf{F}$. If there is no such disjunct, no further reduction is possible in $\mathcal{H}$. In the opposite case, the $\mathcal{H}\ni$ normal form has the form $p \mid \mathbf{F} \wedge q$ (or simply $\mathbf{F} \wedge q$) where p and q are without $\mathbf{F}$. Now $\mathcal{H}$ reduction is possible: one application of the rule $\mathbf{F} \wedge x \rightarrow \mathbf{F}$ results in $p \mid \mathbf{F}$ (or simply $\mathbf{F}$); then one application of $\mathbf{F} \mid x \rightarrow x$ gives p, which must be $\mathcal{H}$-irreducible since it contains no $\mathbf{F}$, so rule R3 cannot be applied, and neither can the other $\mathcal{H}$ rules, since their left-hand sides appear as left-hand sides in $\mathcal{H}\ni$ as well. Thus the $\mathcal{H}\ni$ normal form for formulas without $\mathbf{U}$ is either the same as the $\mathcal{H}$ normal form, or has exactly one additional disjunct of form $\mathbf{F} \wedge q$ where q is a conjunction of distinct propositional variables (excluding $\mathbf{T}$, $\mathbf{F}$).

Furthermore, the variables in q may not occur in other disjuncts (in p), otherwise we may use one of the $\mathcal{H}\ni$ rules R7 to R10. The disjunct $\mathbf{F} \wedge q$ imposes strictness for propositional variables not mentioned in p; it has no effect when the variables are assigned defined values. Clearly this disjunct represents a unique way of imposing strictness not already present in p.

Two distinct (modulo the AC equations) $\mathcal{H}\ni$-irreducible terms, say $p \mid \mathbf{F} \wedge q$ and $p' \mid \mathbf{F} \wedge q'$, have different strictness (if q and q' differ) or have different values for defined values ($\mathbf{T}$ or $\mathbf{F}$) of the propositional variables (if p and p' differ) since Hsiang's system is complete for two-valued propositional logic; and thus they can be instantiated (by the same ground valuation) so that they give unequal truth values ($\mathbf{T}$, $\mathbf{F}$ or $\mathbf{U}$), and therefore must represent unequal formulas (i.e. their equality is not valid). And as already explained, a formula with an occurrence of **U** is reduced to **U** in $\mathcal{H}\ni$, which is different from the above forms. We may conclude that $\mathcal{H}\ni$ is confluent and complete.

4 Kleene operators

In the previous sections we limited ourselves to adaptations of Hsiang normal form. By non-standard normal forms it is possible to handle Kleene operators. We show this briefly below: we now rewrite a formula p with Kleene operators to

$$p1 \ \textbf{when} \ p2$$

where $p1$ and $p2$ are two-valued Boolean formulas in Hsiang normal form, and where the intuitive meaning of **when** is given by x **when** $y =$ **if** y **then** x **else** **U** where the if-construct is left-strict. Thus $p2$ expresses the well-definedness of p and $p1$ its value when defined. In order to formulate $p2$, we use the non-monotonic well-definedness operator Δ, defined by structural induction (considering the constants **T**, **F**, **U** and the operators $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$ [5]) as follows:

$$\begin{aligned}
\Delta\mathbf{T} &\rightarrow \mathbf{T}\\
\Delta\mathbf{F} &\rightarrow \mathbf{T}\\
\Delta\mathbf{U} &\rightarrow \mathbf{F}\\
\Delta\neg x &\rightarrow \Delta x\\
\Delta(x \wedge y) &\rightarrow \Delta x \wedge_2 \Delta y \vee_2 \Delta x \wedge_2 \neg_2 x \vee_2 \Delta y \wedge_2 \neg_2 y\\
\Delta(x \vee y) &\rightarrow \Delta x \wedge_2 \Delta y \vee_2 \Delta x \wedge_2 x \vee_2 \Delta y \wedge_2 y\\
\Delta(x \Rightarrow y) &\rightarrow \Delta x \wedge_2 \Delta y \vee_2 \Delta x \wedge_2 \neg_2 x \vee_2 \Delta y \wedge_2 y\\
\Delta(x \Leftrightarrow y) &\rightarrow \Delta x \wedge_2 \Delta y
\end{aligned}$$

letting Δ bind tighter than the binary operators. Notice that Δx is irreducible when x is a propositional variable. Clearly the Δ operator is two-valued; one may therefore use two-valued operators in the right hand sides, as indicated by the 2-indexes.

> **Lemma 1.** Δp expresses the well-definedness of p in the sense that for each truth value assignment (including **U**) Δp is valid exactly when p is defined (**T** or **F**).

The proof is straight forward by structural induction. Thus when Δp holds, p is two-valued; and occurrences of **U** in p may be replaced by **T** (or by **F**) without

[5] Other operators may be handled as well, for instance $x \vee \neg x$ expresses δx (as above), $\neg(x \Leftrightarrow y)$ expresses strict xor, $\delta x \wedge \delta y \wedge (x \vee y)$ expresses strict or, and $\delta x \wedge \delta y \Rightarrow (x \wedge y)$ expresses strict and.

affecting the value of p. This assumes that p is monotonic, thus Δ may not be used inside p. We then have that Kleene meaning of p is the same as (p **when** Δp) where the p in front of **when** may be rewritten by rules for standard logic, say by the Hsiang system, without affecting the overall value.

This motivates the following rewrite system where K is used to specify Kleene-semantics and C is used to specify classical (two-valued) semantics:

$$\begin{array}{rcl}
K(x) & \rightarrow & C(x) \textbf{ when } \Delta x \\
x \textbf{ when } \mathbf{T} & \rightarrow & x \\
x \textbf{ when } \mathbf{F} & \rightarrow & \mathbf{U} \\
C(\neg x) & \rightarrow & \neg_2\, C(x) \\
C(x \wedge y) & \rightarrow & C(x) \wedge_2\; C(y) \\
C(x \vee y) & \rightarrow & C(x) \vee_2\; C(y) \\
C(x \Rightarrow y) & \rightarrow & C(x) \Rightarrow_2\; C(y)
\end{array}$$

(Other operators may be handled as usual.) Example: $K(x \Rightarrow x)$ reduces to $\mathbf{T}$ **when** Δx.

> **Theorem 2.** The system $\mathcal{K}\infty$ consisting of the above rules (including the Δ-rules) and the Hsiang system (with D1-D4) for 2-indexed operators reflects Kleene semantics, and is convergent (modulo AC), and complete for terms $K(p)$ where p is without explicit $\mathbf{U}$ (or Δ). Thus $\mathcal{K} \models p = q$ is the same as $\mathcal{K}\infty \vdash K(p) = K(q)$.

Proof of ground completeness and Kleene semantics: Assume p is ground and defined. By lemma 1 Δp reduces to $\mathbf{T}$. The rest follows by completeness of the Hsiang system, and the fact that the Hsiang rules are sound on subterms of well-defined Kleene formulas. For undefined p the rewriting result is $\mathbf{U}$.
Proof of uniqueness: Assume p is without $\mathbf{U}$. Convergence is obvious since the Hsiang system is convergent for formulas without $\mathbf{U}$ (occurrences of Δx may be seen as independent variables).
Proof of completeness: If $p = q$ is valid with Kleene semantics, p and q must have the same well-definedness. For formulas without $\mathbf{U}$ and Δ, we have that validity of $p = q$ with Kleene semantics implies validity of $p = q$ with standard Boolean semantics. Thus Hsiang rewriting on (each side of **when** in) p and q will give the same normal form. (End of proof)

In order to have a unique normal form for any Kleene formula p (with $\mathbf{U}$), we need to fix the value in front of **when** for the case that p is not well-defined. Since $K(p) = (\Delta p \wedge_2 K(p)$ **when** $\Delta p)$ is valid in $\mathcal{K}\infty$, this may be done by modifying the K-rule to

$$K(p) \rightarrow \Delta p \wedge_2 C(p) \textbf{ when } \Delta p$$

in the system above. This results in a unique normal form! The only normal form containing $\mathbf{U}$ is $\mathbf{U}$ itself, but Δ may occur in front of variables. This gives us a complete system for all Kleene formulas. For instance, the normal form of $K(x \wedge \mathbf{U})$ is now

$$\mathbf{F}\ \mathbf{when}\ \Delta x \mid_2 \Delta x \wedge_2 x$$

and the normal form of $K(x \vee \mathbf{U})$ is now

$$\Delta x \wedge_2 x\ \mathbf{when}\ \Delta x \wedge_2 x$$

It is possible to define a system with a simpler normal form; for instance, since all occurrences of a variable x is "protected by" a conjunct Δx, the latter may be omitted before **when**; but a full discussion of this is beyond the scope of this paper.

As a final adjustment, we may even avoid the non-monotonic Δ operator. Since $\Delta p = \mathbf{T}$ iff $\delta p = \mathbf{T}$, we have

$$(C(q)\ \mathbf{when}\ \Delta p) = (C(q)\ \mathbf{when}\ \delta p)$$

letting the **when** operator be strict and adding the rule x **when** $\mathbf{U} \to \mathbf{U}$. Now δp must be rewritten with rules as for Δ (replacing Δ by δ), except for the rule $\delta\mathbf{U} \to \mathbf{U}$ (replacing $\Delta\mathbf{U} \to \mathbf{F}$). Notice that we may keep the 2-indexes in the right hand sides of the δ-rules. All in all, we have a convergent and complete system, with monotonic operators only, for Kleene formulas.

The use of order-sorted algebra, error supersorts and stratification [7, 16] would not help us to do convergent rewriting of formulas that have both defined and undefined ground instances, but it could help us to integrate rewriting for standard Boolean with that for three truth values, and we would not need K, C and 2-indexes. In fact, the rules needed for convergent and complete rewriting over the error supersort of Boolean are exactly what we have discussed.

5 Conclusions

In standard propositional logic, much reasoning can be done by a rewrite system which is (convergent and) complete for valid formulas. This is not the case in the presence of three truth values, as there are few interesting valid formulas, and as an equality cannot be reformulated as an equivalence. Therefore inductive completeness is essential for practical applications in this case. This has motivated our work.

We have argued that convergent rewriting for Kleene operators based on disjunctive or conjunctive normal forms is not feasible. And we have discussed extensions and modifications of the Hsiang system to the case of three truth values. Without modification, this system is unsound in the case of three truth values, regardless of how the Boolean operators are extended; and unfortunately, even with modifications it is not suited for non-strict and-operators.

We have proved that a modified version of the Hsiang system is convergent and complete (modulo AC) for propositional logic over three truth values with strict operators. This modified version has the same normal form as the Hsiang system, with the addition of (at most) one extra disjunct controlling strictness. A similar result may easily be obtained for the dual Hsiang system (based on $\vee$ and $\Leftrightarrow$). This system might be applied in OBJ3 (for the error supersort of Boolean) which has strict operators.

The crucial axioms of the Hsiang system are the AC axioms for 'and' and exclusive or, together with distributivity. These axioms are satisfied for strict and- and xor-operators, but not for left-strict 'and' and not for Kleene-and (regardless of choice of xor). There are also non-monotonic versions of the operators satisfying these axioms, but they could not express the interesting extensions of the Boolean operators, not even the strict extension of the not-operator.

We have here focused on adaptations of rewrite systems for standard Boolean, reusing their normal forms. We have shown that by introducing an additional operator it is possible to define a convergent and complete system for propositional logic over three truth values with Kleene operators, using Hsiang rewriting of subterms. However, there may well be other systems for Kleene formulas with simpler normal forms.

References

[1] L. Bachmair and N. Dershowitz: "Critical pair criteria for completion", *Journal of Symbolic Computation* **6** (1988) pp. 1-18.

[2] S. Blamey: "Partial Logic", in *Handbook of Philosophical Logic,* Vol III, D. Gabbay and F. Guenthner (eds.) 1986 D. Reidel Publ. Company, pp. 1-70.

[3] J.H. Cheng: "A logic for partial functions",
PhD thesis, University of Manchester, Dept. of CS, 1986.

[4] O.-J. Dahl: *Verifiable programming*, Prentice Hall, 1992.

[5] N. Dershowitz and J.-P. Jouannaud: "Rewrite Systems",
in *Handbook of Theoretical Computer Science, Vol. B*, Elsevier, pp. 244-320, 1990.

[6] J.A. Goguen: "Abstract errors for abstract data types",
in *Formal Description of Programming Concepts*, North Holland, 1978.

[7] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi and J.-P. Jouannaud:
"Introducing OBJ", Oxford University Computing Laboratory, 1993.

[8] J. Hsiang: “Topics in automated theorem proving and program generation”,
PhD thesis, Dept. of C.S., University of Illinois at Champaign-Urbana, 1982.

[9] J. Hsiang: “Refutational theorem proving using term rewriting systems”, *Artificial Intelligence* **25** (1985) 255-300.

[10] S.C. Kleene: *Introduction to Metamathematics,* North Holland, 1952.

[11] D.E. Knuth and P.B. Bendix: “Simple word problems in universal algebras”,
in *Computational Problems in Abstract Algebras*, Pergamon Press, 1970, pp. 263-297.

[12] O. Owe: “Partial Logics Reconsidered: A Conservative Approach”, *Formal Aspects of Computing* **5** (1993) pp. 208-223.

[13] E. Paul: “Equational Methods in First Order Predicate Calculus”, *Journal of Symbolic Computation* **1** (1985) pp. 7-29.

[14] G.E. Peterson and M.E. Stickel: “Complete Sets of Reductions for Some Equational Theories”, *JACM* **28**, 2 (1981) pp. 233-264.

[15] W.V. Quine: “The problem of simplifying truth functions”, *Am. Math. Monthly* **59**, 8 (1952) pp. 521-531.

[16] G. Smolka, W. Nutt, J.A. Goguen and J. Meseguer:
“Order-sorted equational computation”, in *Resolution of Equations in Algebraic Structures, Vol. II: Rewriting Techniques*, Academic Press, 1989, pp. 299-369.

[17] A. Urquhart: ”Many-valued Logic”, in *Handbook of Philosophical Logic,* Vol III, D. Gabbay and F. Guenthner (eds.) 1986 D. Reidel Publ. Company, pp. 71-116.