Mathware & Soft Computing 4 (1997) 99-127

A Reduction-based Theorem Prover for 3-valued Logic*

G. Aguilera Venegas, I. P. de Guzmán, M. Ojeda Aciego
Dept. Matemática Aplicada. Univ. de Málaga.
E-29071 Málaga, Spain
{*gabri,aciego*}*@ctima.uma.es, pguzman@ccuma.uma.es*

Abstract

We present a new prover for propositional 3-valued logics, TAS-M3, which is an extension of the TAS-D prover for classical propositional logic. TAS-M3 uses the TAS methodology and, consequently, it is a *reduction-based* method. Thus, its power is based on the reductions of the size of the formula executed by the $\mathcal{F}$ transformation. This transformation dynamically "filters" the information contained in the syntactic structure of the formula to avoid as much distributions (of $\wedge$ wrt $\vee$ in our case) as possible, in order to improve efficiency. In our opinion, this filtering is the key of the TAS methodology which, as shown in this paper, allows the method to be extremely adaptable, because switching to different kinds of logic is possible without having to redesign the whole prover.

1 Introduction

In this work we present a new prover for propositional 3-valued logics, named TAS-M3, which is an extension of the TAS-D prover for classical propositional logic. TAS-M3 uses the TAS methodology which has been applied to classical propositional logic [2], first-order logic [5] and temporal logic [4]. As a *reduction-based*[1] method, the power of TAS-M3 is based on processes (reductions) which reduce of the size of the formula, grouped as the $\mathcal{F}$ transformation. This transformation dynamically "filters" the information contained in the syntactic structure of the formula to avoid as much distributions (of $\wedge$ wrt $\vee$ in our case) as possible, in order to improve efficiency. Roughly speaking, the idea is to get the information given by unitary partial interpretations; this idea, as seen in the classical and modal versions, has proved to be extremely useful.

*This work has been partially supported by CICYT project number TIC94-0847-C02-02.

[1]This name is introduced by the authors to refer to the TAS methods, since the core of all of them is, essentially, the use of reductions on the input formula in order to avoid as much distributions as possible.

TAS applies a sequence of transformations to the formula being considered. It is worth to note that the transformations are not just applied one after the other; through the efficient determination and manipulation of sets of unitary models of a formula, the method *investigates exhaustively* the formula, to detect if it is possible to decrease the size of the formula being analysed.

The power of the method is based not only on the intrinsically parallel design of the involved transformations, but also on the fact that every transformation in $\mathcal{F}$ reduces the size of the formula and, only when no information can be used, the last transformation commissioned on the distributions is applied. The distribution is made gradually, so we distribute only once and then apply the reducing $\mathcal{F}$ transformation once again on each parallel task.

Sets of unitary models, the Δ-sets, are associated to each node in the syntactic tree of the formula, they can be considered the key tool of TAS methodology, since they are used to conclude whether the structure of the syntactic tree has or has not direct information about the validity of the formula. This way, either the method ends giving this information or, otherwise, it decreases the size of the problem before applying the next transformation. So, it is possible to decrease the number of distributions or, even, to avoid them all.

The proposed theorem prover can be applied to any three-valued logic, not just M3, but for the sake of simplicity we describe it explicitly for M3 and in Section 6.1 we show how the general case can be achieved.

The paper is organised as follows:

- Firstly, the classical propositional version is briefly presented, to help the understanding of the method in the many-valued case.
- Later, the functionally complete M3 three-valued logic is introduced..
- Then, the TAS-M3 method is introduced, paying special attention to the $\mathcal{F}$ transformation and to the different reduction processes involved in it.
- Finally, some examples, a sketch of how the method can be applied to *any* three-valued logic, and the conclusions are presented.

1.1 The TAS prover in classical propositional logic (sketch)

Let L be the language of classical propositional logic, let T_A be the syntactic tree of a well formed formula (wff) A.

Given a wff A, the input of TAS-D is the syntactic tree of $\neg A$, $T_{\neg A}$. The flow diagram of the TAS provers is shown in Figure 1, where the outputs are either VALID or NON-VALID and a model for $\neg A$ (countermodel). Each module for the classical propositional version will be commented below.

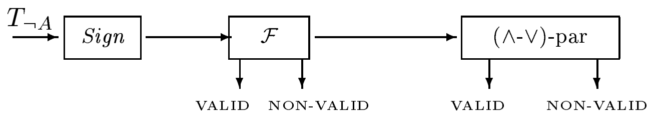


Figure 1: The flow diagram of a TAS method.

1.1.1 The Sign transformation

Definition 1.1 Given a wff A and $\kappa \in \{1, -1\}$, $A(\kappa)$ is recursively defined as follows:

$p\,(1) = p$	$p\,(-1) = \neg p$
$(\neg A)\,(1) = A\,(-1)$	$(\neg A)\,(-1) = A\,(1)$
$(A \vee B)\,(1) = A\,(1) \vee B\,(1)$	$(A \vee B)\,(-1) = A\,(-1) \wedge B\,(-1)$
$(A \wedge B)\,(1) = A\,(1) \wedge B\,(1)$	$(A \wedge B)\,(-1) = A\,(-1) \vee B\,(-1)$
$(A \to B)\,(1) = A\,(-1) \vee B\,(1)$	$(A \to B)\,(-1) = A\,(1) \wedge B\,(-1)$

Given a wff A, the first task in the method is to obtain $T_{A\,(1)}$. It is obvious that if A is a wff then $A\,(1)$ is a formula in negation normal form (nnf) equivalent to A.

The idea to use the information given by partial interpretations, borrowed from Quine's method, is taken in TAS-D just for unitary partial interpretations, which are used all over the method by means of the sets Δ_0 and Δ_1, the key tools of TAS-D:

Definition 1.2 Given a nnf A, the sets $\Delta_0(A)$ and $\Delta_1(A)$ are recursively defined by:

$$\Delta_0(p) = \{p\,0\}; \qquad \Delta_1(p) = \{p\,1\}; \qquad \Delta_0(\neg p) = \{p\,1\}; \qquad \Delta_1(\neg p) = \{p\,0\}$$

$$\Delta_0\left(\bigvee_{i=1}^{n} A_i\right) = \bigcap_{i=1}^{n} \Delta_0(A_i); \qquad \Delta_1\left(\bigvee_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} \Delta_1(A_i)$$

$$\Delta_0\left(\bigwedge_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} \Delta_0(A_i); \qquad \Delta_1\left(\bigwedge_{i=1}^{n} A_i\right) = \bigcap_{i=1}^{n} \Delta_1(A_i)$$

Remark 1.1 Note the relationship between the definition of the sets $\Delta_b(A)$ and the tableau rules for the extension of α- and β-formulas; the subscript b can be thought of as the *sign* of the formula A.

It is obvious that the elements of $\Delta_0(A)$ can be seen as the unitary models of $\neg A$ and the elements of $\Delta_1(A)$ can be seen as the unitary models of A.

If A is a wff, the output of the *Label* transformation is $T_{A\,(+)}$ whose nodes N are labelled with the ordered pair $\big(\Delta_0(B), \Delta_1(B)\big)$ where B is the subformula of A such that N is the root of T_B.

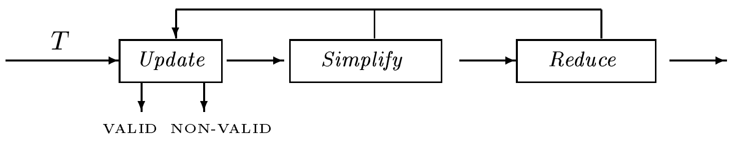


Figure 2: The $\mathcal{F}$ transformation.

1.1.2 The $\mathcal{F}$ transformation

The input of this transformation is a generalised syntactic tree T_C. In this stage we try to detect by means of the labels (Δ_0, Δ_1) whether the structure of T_C provides either complete information about the unsatisfiability of C or useful information to decrease the size of C before distributing.

The core of $\mathcal{F}$ is the sequence of performing, as many times as possible, the processes *simplify* and *reduce* with their corresponding *updates*, as indicated in Figure 2.

This transformation can be seen as a sequence of *filters* of the information contained in the Δ sets, that is why it is called $\mathcal{F}$.

How the information contained in the labels (Δ_0, Δ_1) is used by any of the previous processes is sketched below:

1. The process *simplify* (or *strong reduction*) allows to deduce that a subformula B, in particular the whole formula, is equivalent to $\top$, $\bot$ or a literal. This is done by using the following result: Let p be a propositional symbol, then

   (a) If $\{p0, p1\} \in \Delta_0(A)$ then $A \equiv \bot$.

   (b) If $\{p0, p1\} \in \Delta_1(A)$ then $A \equiv \top$.

   (c) If $\{p0\} = \Delta_0(A)$ and $\{p1\} = \Delta_1(A)$ then $A \equiv p$.

   (d) If $\{p1\} = \Delta_0(A)$ and $\{p0\} = \Delta_1(A)$ then $A \equiv \neg p$.

   where $A \equiv B$ means that A and B are logically equivalent, that is $I(A) = I(B)$ for every interpretation I.

2. The process *reduce* (or *standard reduction*) uses again the information contained in (Δ_0, Δ_1) in order to decrease the size of the tree before distributing; specifically, substituting A by a equisatisfiable formula in which the symbols in $\Delta_0(A)$ or $\Delta_1(A)$ occur at most once. Note that after *reducing* a tree, another simplifiable tree can be obtained and thus, a new *simplification* may avoid distributions (or end TAS-D).

   Here we use the following results:

   (a) If $p0 \in \Delta_0(A)$ then

      i. $A \equiv p \wedge A[p/\top]$
      ii. A and $A[p/\top]$ are equisatisfiable.
   (b) If $p1 \in \Delta_0(A)$ then
      i. $A \equiv \neg p \wedge A[p/\bot]$
      ii. A and $A[p/\bot]$ are equisatisfiable.
   (c) If $p0 \in \Delta_1(A)$ then $A \equiv \neg p \vee A[p/\top]$.
   (d) If $p1 \in \Delta_1(A)$ then $A \equiv p \vee A[p/\bot]$.
   (e) If $\Delta_1(A) \neq \varnothing$ then A is satisfiable; in this case we say that A is finalizable.

3. Essentially, the *update* transformation deletes the occurrences of the $\top$ and $\bot$ introduced by the reductions, and recalculates the labels of the modified nodes and their ascendants.

For most formulas, if TAS-D does not finish after executing $\mathcal{F}$ then the size of the input tree of ($\wedge$-$\vee$)-par is drastically decreased. The interest of this strategy is based on the fact that it can reduce the size of the problem before applying the transformation which holds most of the complexity (($\wedge$-$\vee$)-par in our case).

1.1.3 The ($\wedge$-$\vee$)-par transformation

This transformation makes the distribution of $\wedge$ wrt $\vee$ that the previous application of $\mathcal{F}$ could not avoid. In ($\wedge$-$\vee$)-par only single distributions are made each time and, after generating parallel tasks (the children of a root node $\vee$), the $\mathcal{F}$ transformation is applied again to every parallel task; thus, either the execution finish or the size of the generated tasks is decreased (this can avoid ulterior distributions)

1.1.4 A traced example

Consider the following formula $A = (p \to (q \to r)) \to ((p \to q) \to (p \to r))$, the trace of the method to test the validity of A is sketched below:

1. Input $\neg A$:
$$\neg((p \to (q \to r)) \to ((p \to q) \to (p \to r)))$$
2. *Sign*:
$$(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q) \wedge p \wedge \neg r$$
3. Reduction 2(a)ii wrt p and $\neg r$; update:
$$q \wedge \neg q$$
4. Simplification 1a:
$$\bot$$
5. Output: The formula is valid.

2 The M3 logic

As indicated in the introduction, the aim of this work is to show how the TAS methodology can be extended to many-valued logic. Specifically, we show a reduction-based prover for the M3 logic, a functionally complete 3-valued logic, which is an extension of that introduced in [11] and used in [7].

Definition 2.1 The M3 logic is defined by the pair $(\mathcal{L}, \mathcal{M})$ in which the algebra $\mathcal{L} = (Q, \bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow)$ has similarity type $\langle 0,0,0,1,1,2,2,2\rangle$ and its associated matrix is $\mathcal{M} = (\mathbf{3}, 1, \bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow)$, where $\mathbf{3} = \{0, \frac{1}{2}, 1\}$. Elements in Q are called *propositional symbols*, $\neg$ is the *strong negation*, $\sim$ is the *weak negation*, $\vee$ is the *disjunction*, a $\wedge$ is the *conjunction* and $\rightarrow$ is the *weak implication*.

The semantics of these connectives is defined as follows:

1. $\bot, \oslash$ and $\top$ (0-ary connectives) are, respectively, the constants $0, \frac{1}{2}$ and 1.
2. $\neg i = 1 - i$
3. $\sim i = \begin{cases} 1 & \text{if } i \in \{0, \frac{1}{2}\} \\ 0 & \text{if } i \in \{1\} \end{cases}$
4. $i \wedge j = \min\{i, j\}$.
5. $i \vee j = \max\{i, j\}$.
6. $i \rightarrow j = \begin{cases} 1 & \text{if } i \in \{0, \frac{1}{2}\} \\ j & \text{if } i \in \{1\} \end{cases}$

The truth tables for the connectives just defined are the following

	$\neg$	$\sim$
0	1	1
$\frac{1}{2}$	$\frac{1}{2}$	1
1	0	0

$\vee$	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
1	1	1	1

$\wedge$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	$\frac{1}{2}$	1

$\rightarrow$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	1	1	1
1	0	$\frac{1}{2}$	1

Definition 2.2 An interpretation in M3 is, as usual, an assignment of truth values to each formula of the language. These assignments are generated by any mapping $I: Q \rightarrow \mathbf{3}$, as I defines a unique homomorphism $\mathcal{L} \rightarrow \mathcal{M}$ also denoted I.

The truth value assertions J_i can be defined in M3 using the subset of primitive connectives $\{\neg, \sim, \wedge\}$ as follows:

$$J_0 A \equiv \neg \sim \neg A \qquad J_{\frac{1}{2}} A \equiv \sim A \wedge \sim \neg A \qquad J_1 A \equiv \neg \sim A$$

where $\equiv$ means logical equivalence in M3, i.e. $A \equiv B$ if and only $I(A) = I(B)$ for every interpretation I.

Theorem 2.1 *The M3 logic is functionally complete.*

Proof. It follows directly from Lemma 2.9 in [12].

3 The TAS-M3 method

The design of the algorithm for the M3 logic is the same than that for classical logic, and also has the same properties of efficiency, flexibility and parallelism. This last property allows to execute in parallel the only part of the algorithm whose complexity is exponential.

TAS-M3 is a refutation method; to analyse the validity of a formula A, we use Theorem 3.1 in which we use the usual definition of validity, and we also introduce the notion of quasi-satisfiability:

Definition 3.1 Let A be a wff in M3, we say that:

1. A is *valid* if $I(A) = 1$ for every interpretation I.

2. A is *quasi-satisfiable* if there exists an interpretation I such that $I(A) \in \{\frac{1}{2}, 1\}$.

It is easy to note that satisfiability in classical logic corresponds to quasi-satisfiability in M3 in the sense of the following theorem:

Theorem 3.1 *Let A and A_i be wffs in M3, then*

- *A is not valid iff $\neg A$ is quasi-satisfiable.*
- *$A = \bigvee_{i \in I} A_i$ is quasi-satisfiable iff A_i is quasi-satisfiable for some $i \in I$.*

The flow diagram of the method is the same as in the classical case, and it appears in Figure 1, the input of TAS-M3 is the (syntactic tree of the) strong negation of a formula with no constants; by analysing the structure of this tree, the method dynamically transforms it, avoiding as much distributions as possible, and decreasing the size of the formula.

Definition 3.2 The *syntactic tree* of a wff A of M3, denoted by T_A, is the binary tree recursively defined as follows:

1. If $A \in Q \cup \{\top, \oslash, \bot\}$, then T_A is A.

2. If $A = *B$, where $* \in \{\neg, \sim\}$, then T_A is $\begin{array}{c} * \\ | \\ T_B \end{array}$

3. If $A = B * C$ where $* \in \{\wedge, \vee, \rightarrow\}$, then T_A is

$$\begin{tikzcd} & * \arrow[dl, dash] \arrow[dr, dash] & \\ T_B & & T_C \end{tikzcd}$$

3.1 The Sign transformation

The aim of this first process is to transform the syntactic tree of the input formula into the syntactic tree of an equivalent unary normal form formula (see Definition 3.3) in M3, this concept will defined later.

The following laws will be used by this transformation:

1. Elimination of $\rightarrow$ laws:

$$A \rightarrow B \equiv {\sim} A \vee B \quad \neg(A \rightarrow B) \equiv \neg{\sim} A \wedge \neg B \quad {\sim}(A \rightarrow B) \equiv \neg{\sim} A \wedge {\sim} B$$

2. de Morgan laws:

$$\begin{array}{cc} \neg(A \wedge B) \equiv \neg A \vee \neg B & \neg(A \vee B) \equiv \neg A \wedge \neg B \\ {\sim}(A \wedge B) \equiv {\sim} A \vee {\sim} B & {\sim}(A \vee B) \equiv {\sim} A \wedge {\sim} B \end{array}$$

3. Multiple negation laws.

   (a) $\neg^k A \equiv \begin{cases} A & \text{if } k \text{ is even} \\ \neg A & \text{if } k \text{ is odd} \end{cases}$

   (b) ${\sim}^k A \equiv \begin{cases} \neg{\sim} A & \text{if } k \text{ is even} \\ {\sim} A & \text{if } k \text{ is odd} \end{cases}$

   (c) ${\sim}\neg{\sim} A \equiv {\sim} A$

   (d) $\neg{\sim}\neg{\sim} A \equiv \neg{\sim} A$

   (e) ${\sim}\neg{\sim}\neg A \equiv {\sim}\neg A$

¿From the multiple negation laws we have that the only sequences of $\neg$ and ${\sim}$, up to equivalence, are the sequences in the set Λ defined below, where ϵ is the empty sequence.

$$\{\epsilon, \neg, {\sim}, \neg{\sim}, {\sim}\neg, \neg{\sim}\neg\}$$

using the semantics of M3, we can just consider the sequences in the set

$$\{\epsilon, \neg, \neg J_1, J_1, \neg J_0, J_0\}$$

Although $J_{\frac{1}{2}}$ and $\neg J_{\frac{1}{2}}$ cannot appear in the input formula, they can be generated during the execution of the method, as we will see in the *reduce* transformation. Consequently, we have the following definition:

Definition 3.3 A *3-valued literal* is any wff αp where $p \in Q$ and $\alpha \in \Lambda$, where

$$\Lambda = \{\epsilon, J_0, J_{\frac{1}{2}}, J_1, \neg, \neg J_0, \neg J_{\frac{1}{2}}, \neg J_1\}$$

The *prefix* of αp, is the sequence α, and the *suffix* of αp is the propositional symbol p.

A wff A in the M3 logic is said to be a unary normal form formula (unf) if the connectives in A are in the set $\{\neg, J_0, J_{\frac{1}{2}}, J_1, \wedge, \vee\}$ and the scope of every unary connective is either a propositional symbol or another unary connective.

Lemma 3.1 *Every 3-valued literal is satisfiable.*

Proof. The proof is trivial, since we have only to check that in every column in the following table there is at least a 1.

	ϵ	$\neg$	J_0	$\neg J_0$	$J_{\frac{1}{2}}$	$\neg J_{\frac{1}{2}}$	J_1	$\neg J_1$
0	0	1	1	0	0	1	0	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	1	1	0	0	1
1	1	0	0	1	0	1	1	0

The associative laws allow us to consider an expression like $A_1 \vee \cdots \vee A_n$ or $A_1 \wedge \cdots \wedge A_n$ as a wff (in short form $\bigvee_{i=1}^{n} A_i$ and $\bigwedge_{i=1}^{n} A_i$). This fact, and the consideration of the new literal prefixes in Λ, leads to the following extension of the definition of syntactic tree for a unf:

Definition 3.4 Given a formula in unf A of L, its *generalised syntactic tree*, denoted by $\widehat{T}_A$, is defined as follows:

1. If $A = \bigodot_{i=1}^{n} A_i$, where $\bigodot$ is $\bigwedge$ or $\bigvee$, then $\widehat{T}_A$ is

$$\begin{tikzcd}
& & \odot \arrow[dll, dash] \arrow[dl, dash] \arrow[drr, dash] & & \\
\widehat{T}_{A_1} & \widehat{T}_{A_2} & & \cdots & \widehat{T}_{A_n}
\end{tikzcd}$$

2. $\widehat{T}_A = A$ in other case.

Remark 3.1 In the rest of the paper, the syntactic tree of A will always mean the generalised syntactic tree of A and both of them will be denoted by T_A.

The transformation into an equivalent unf is done by a recursive process applied to the root of the corresponding syntactic tree, eliminating the connective $\rightarrow$ and putting the negations down to the leaves.

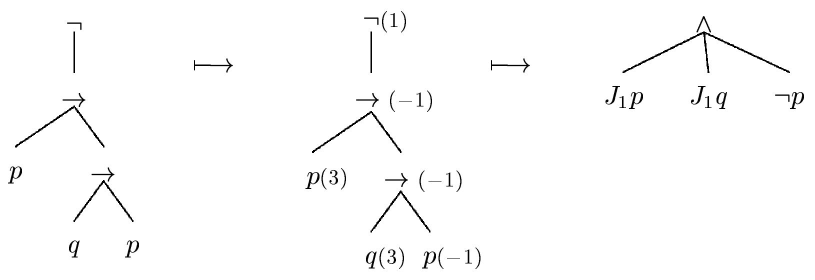
Definition 3.5 Let A be a wff and $\kappa \in \{-3, -2, -1, 1, 2, 3\}$, then $A(\kappa)$ is recursively defined as follows:

$p(1) = p$	$p(-1) = \neg p$
$p(2) = \neg J_0 p$	$p(-2) = J_0 p$
$p(3) = J_1 p$	$p(-3) = \neg J_1 p$
$(A * B)(\kappa) = A(\kappa) * B(\kappa)$	if $\kappa > 0$ and $* \in \{\vee, \wedge\}$
$(A * B)(\kappa) = A(\kappa) \overline{*} B(\kappa)$	if $\kappa < 0$, $* \in \{\vee, \wedge\}$, $\overline{\vee} = \wedge$ and $\overline{\wedge} = \vee$
$(A \rightarrow B)(\kappa) = A(-3) \vee B(\kappa)$	if $\kappa > 0$
$(A \rightarrow B)(\kappa) = A(3) \wedge B(\kappa)$	if $\kappa < 0$
$(\sim A)(\kappa) = A(-3)$	if $\kappa > 0$
$(\sim A)(\kappa) = A(3)$	if $\kappa < 0$
$(\neg A)(\kappa) = A(\gamma(\kappa))$	where γ is the function defined below

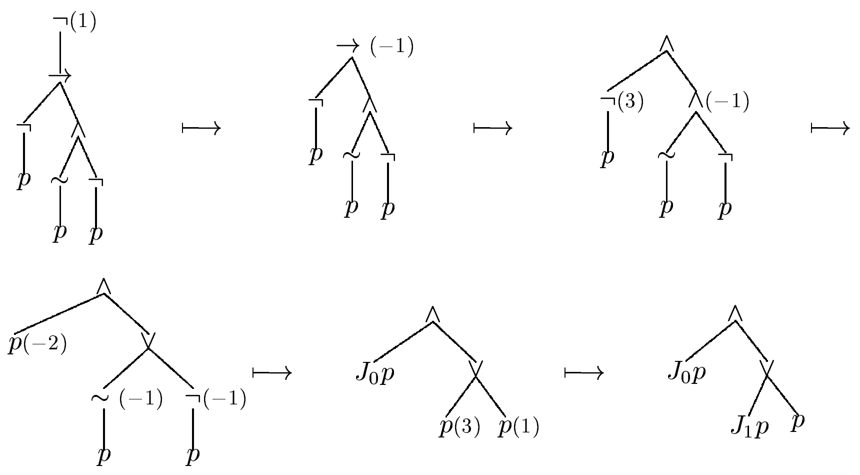
$$
\begin{array}{ll}
\multicolumn{2}{c}{\gamma : \{-3,-2,-1,1,2,3\} \longrightarrow \{-3,-2,-1,1,2,3\}} \\
\gamma(1) = -1 & \gamma(-1) = 1 \\
\gamma(2) = -3 & \gamma(-2) = 3 \\
\gamma(3) = -2 & \gamma(-3) = 2
\end{array}
$$

Given a wff A, to *sign* its syntactic tree T_A is to get the generalised syntactic tree associated to $A\,(1)$.

Example 3.1 Consider the formula $A = p \to (q \to p)$. The input of TAS-M3 is the syntactic tree of $\neg A$. After *signing*, the last tree of the sequence shown below is obtained:



Example 3.2 Given the formula $A = \neg p \to (\sim p \wedge \neg p)$, the input of TAS-M3 is $T_{\neg A}$ and the application of the *sign* transformation is shown below



Note that although we have shown in both examples the whole trace of the transformation, it is executed by traversing the syntactic tree only once.

Theorem 3.2 *The Sign process is meaning-preserving, that is Sign*$(A) \equiv A$.

Proof. The proof is immediate, since the transformation only uses equivalence laws.

3.2 The Δ sets

The semantic basis of TAS-M3 is the same as in every TAS method, the use of unitary partial interpretations. This information will be used all over the method by means of the sets Δ_b, the key tools of TAS-M3. These Δ sets contain information allowing either to detect subformulas which are logically equivalent to $\top$ or $\bot$ or to substitute a formula with a smaller sized one.

In the 3-valued case we are introducing, the definition of the Δ sets will contain 3-valued literals satisfying the following properties:

1. If $\models (l \rightarrow A)$ then $A \equiv l \vee B$
2. If $\models (A \rightarrow l)$ then $A \equiv l \wedge B$

where the size of B is smaller than that of A.

These sets are Δ_0, $\Delta_{\frac{1}{2}}$ and Δ_1; the elements in the Δ sets are shaped pb with $p \in Q$ and $b \in \mathbf{3}$. Our aim when defining $\Delta_b(A)$ is that whenever $pb_1 \in \Delta_{b_2}(A)$, where $b_1, b_2 \in \mathbf{3}$, then for any interpretation I satisfying $I(p) = b_1$ we have $I(A) = b_2$. This idea leads to the following definition:

Definition 3.6 Let p be a propositional symbol, then

$$\Delta_0(p) = \{p0\} \qquad \Delta_{\frac{1}{2}}(p) = \{p\tfrac{1}{2}\} \qquad \Delta_1(p) = \{p1\}$$

the definition for the rest of the cases uses the semantics of the 3-valued literals and the connectives $\wedge$ and $\vee$. Let A and A_i be unfs. The sets Δ_0, $\Delta_{\frac{1}{2}}$ and Δ_1 are recursively defined as follows:

$\Delta_0(\neg p) = \{p1\}$	$\Delta_{\frac{1}{2}}(\neg p) = \{p\frac{1}{2}\}$	$\Delta_1(\neg p) = \{p0\}$
$\Delta_0(J_0 p) = \{p\frac{1}{2}, p1\}$	$\Delta_{\frac{1}{2}}(J_0 p) = \varnothing$	$\Delta_1(J_0 p) = \{p0\}$
$\Delta_0(\neg J_0 p) = \{p0\}$	$\Delta_{\frac{1}{2}}(\neg J_0 p) = \varnothing$	$\Delta_1(\neg J_0 p) = \{p\frac{1}{2}, p1\}$
$\Delta_0(J_{\frac{1}{2}} p) = \{p0, p1\}$	$\Delta_{\frac{1}{2}}(J_{\frac{1}{2}} p) = \varnothing$	$\Delta_1(J_{\frac{1}{2}} p) = \{p\frac{1}{2}\}$
$\Delta_0(\neg J_{\frac{1}{2}} p) = \{p\frac{1}{2}\}$	$\Delta_{\frac{1}{2}}(\neg J_{\frac{1}{2}} p) = \varnothing$	$\Delta_1(\neg J_{\frac{1}{2}} p) = \{p0, p1\}$
$\Delta_0(J_1 p) = \{p0, p\frac{1}{2}\}$	$\Delta_{\frac{1}{2}}(J_1 p) = \varnothing$	$\Delta_1(J_1 p) = \{p1\}$
$\Delta_0(\neg J_1 p) = \{p1\}$	$\Delta_{\frac{1}{2}}(\neg J_1 p) = \varnothing$	$\Delta_1(\neg J_1 p) = \{p0, p\frac{1}{2}\}$
$\Delta_0(\bigwedge_{i=1}^{n} A_i) = \bigcup_{i=1}^{n} \Delta_0(A_i)$		$\Delta_1(\bigwedge_{i=1}^{n} A_i) = \bigcap_{i=1}^{n} \Delta_1(A_i)$
$\Delta_0(\bigvee_{i=1}^{n} A_i) = \bigcap_{i=1}^{n} \Delta_0(A_i)$		$\Delta_1(\bigvee_{i=1}^{n} A_i) = \bigcup_{i=1}^{n} \Delta_1(A_i)$

Finally, we have to define $\Delta_{\frac{1}{2}}$ for conjunctions and disjunctions; in the binary case we have

- $\Delta_{\frac{1}{2}}(A_1 \wedge A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_1(A_2))] \cup (\Delta_1(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$
- $\Delta_{\frac{1}{2}}(A_1 \vee A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_0(A_2))] \cup (\Delta_0(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$

and in the generalised case the set $\Delta_{\frac{1}{2}}$ is built by left associativity and using the binary definition, i.e.

$$\Delta_{\frac{1}{2}}\left(\bigwedge_{i=1}^{n} A_i\right) = \Delta_{\frac{1}{2}}\left(\bigwedge_{i=1}^{n-1} A_i \wedge A_n\right) \qquad \Delta_{\frac{1}{2}}\left(\bigvee_{i=1}^{n} A_i\right) = \Delta_{\frac{1}{2}}\left(\bigvee_{i=1}^{n-1} A_i \vee A_n\right)$$

Remark 3.2 For a better understanding, it is worth to consider the correspondence between the definition of the Δ sets and certain tableau rules. Note, for instance, how our definition of $\Delta_{\frac{1}{2}}(A_1 \vee A_2)$ coincides with the tableau rule in [7, pg. 87].

Remark 3.3 Note that the difficulty in the calculation of $\Delta_{\frac{1}{2}}$ is only apparent, as the following results show:

- If l is a literal whose prefix is neither ϵ nor $\neg$, then $\Delta_{\frac{1}{2}}(l) = \varnothing$.
- Let $A_1, \dots, A_n$ be wffs such that $\Delta_{\frac{1}{2}}(A_i) = \varnothing$ for all $i = 1, \dots, n$, then $\Delta_{\frac{1}{2}}\left(\bigvee_{i=1}^{n} A_i\right) = \Delta_{\frac{1}{2}}\left(\bigwedge_{i=1}^{n} A_i\right) = \varnothing$.

Definition 3.7 Given a unf A, to *label* is a recursive process which associates to each node N in T_A the triple $(\Delta_0(B), \Delta_{\frac{1}{2}}(B), \Delta_1(B))$, where B is the subformula of A determined by N.

Example 3.3 Continuing with Example 3.1, after *signing* we get the syntactic tree of $J_1p \wedge J_1q \wedge \neg p$; the result of *labelling* this tree is the following:

$$\begin{tikzcd}
 & \wedge(\{p0, p\tfrac{1}{2}, p1, q0, q\tfrac{1}{2}\}, \varnothing, \varnothing) \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
J_1p(\{p0, p\tfrac{1}{2}\}, \varnothing, \{p1\}) & J_1q(\{q0, q\tfrac{1}{2}\}, \varnothing, \{q1\}) & \neg p(\{p1\}, \{p\tfrac{1}{2}\}, \{p0\})
\end{tikzcd}$$

Example 3.4 Let T_B be the output of Example 3.2, after *labelling* this tree we get the following result

$$\begin{tikzcd}
 & \wedge(\{p0, p\tfrac{1}{2}, p1\}, \varnothing, \varnothing) \arrow[dl, dash] \arrow[dr, dash] & & \\
J_0p(\{p\tfrac{1}{2}, p1\}, \varnothing, \{p0\}) & & \vee(\{p0\}, \{p\tfrac{1}{2}\}, \{p1\}) \arrow[dl, dash] \arrow[dr, dash] & \\
 & J_1p(\{p0, p\tfrac{1}{2}\}, \varnothing, \{p1\}) & & p(\{p0\}, \{p\tfrac{1}{2}\}, \{p1\})
\end{tikzcd}$$

3.3 The information in the Δ sets

The information in the Δ sets associated to a formula allows to determine its equivalence to $\top$, to $\bot$ or to a 3-valued literal, or to transform it into a smaller-sized one by preserving quasi-satisfiability. The following sections show how to extract this information.

3.3.1 Information in $\boldsymbol{\Delta_0}$

The lemma below shows the semantic content of the set Δ_0:

Lemma 3.2 *Let A be a unf and $pb \in \Delta_0(A)$, an interpretation I satisfies $I(p) = b$ iff $I(A) = 0$.*

Proof. Follows by the correspondence between the definition of the Δ-sets and tableau rules, as stated in Remark 3.2.

The following theorem shows how the information in Δ_0 can be used.

Theorem 3.3 *Let A be a unf and p propositional symbol, then:*

1.	$p0 \in \Delta_0(A)$	*if and only if*	$A \equiv A \wedge \neg J_0 p$
2.	$p\frac{1}{2} \in \Delta_0(A)$	*if and only if*	$A \equiv A \wedge \neg J_{\frac{1}{2}} p$
3.	$p1 \in \Delta_0(A)$	*if and only if*	$A \equiv A \wedge \neg J_1 p$
4.	$\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$	*if and only if*	$A \equiv A \wedge J_1 p$
5.	$\{p0, p1\} \subseteq \Delta_0(A)$	*if and only if*	$A \equiv A \wedge J_{\frac{1}{2}} p$
6.	$\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$	*if and only if*	$A \equiv A \wedge J_0 p$
7.	$\{p0, p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$	*implies*	$A \equiv \bot$

Proof. The last item is immediate. To prove the necessity in the if and only if items we have only to apply the definition of Δ_0 to the conjunctions in the right hand side. Let us prove the sufficiency:

1. Suppose $p0 \in \Delta_0(A)$ and let I be a arbitrary interpretation. We have three cases to consider:

    - If $I(p) = 0$ then, by Lemma 3.2, we have $I(A) = 0$. Therefore, $I(A) = I(A \wedge \neg J_1 p) = 0$.
    - If $I(p) \in \{\frac{1}{2}, 1\}$, then $I(\neg J_1 p) = 1$. Therefore, $I(A) = I(A \wedge \neg J_1 p)$.

    The rest of the items are proved similarly.

3.3.2 Information in $\boldsymbol{\Delta_1}$

The following results, whose proof are similar to that of Lemma 3.2 and Theorem 3.3, show the usefulness of the set Δ_1:

Lemma 3.3 *Let A be a unf and $pb \in \Delta_1(A)$, an interpretation I satisfies $I(p) = b$ iff $I(A) = 1$.*

Theorem 3.4 *Let A be a unf and p a propositional symbol, then:*

1.	$p0 \in \Delta_1(A)$	*if and only if*	$A \equiv A \vee J_0 p$
2.	$p\frac{1}{2} \in \Delta_1(A)$	*if and only if*	$A \equiv A \vee J_{\frac{1}{2}} p$
3.	$p1 \in \Delta_1(A)$	*if and only if*	$A \equiv A \vee J_1 p$
4.	$\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$	*if and only if*	$A \equiv A \vee \neg J_1 p$
5.	$\{p0, p1\} \subseteq \Delta_1(A)$	*if and only if*	$A \equiv A \vee \neg J_{\frac{1}{2}} p$
6.	$\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$	*if and only if*	$A \equiv A \vee \neg J_0 p$
7.	$\{p0, p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$	*implies*	$A \equiv \top$

3.3.3 Information in $\Delta_{\frac{1}{2}}$

The following lemma, whose proof is similar to that of Lemma 3.2, show the semantic content of the set $\Delta_{\frac{1}{2}}$:

Lemma 3.4 *Let A be a unf and $pb \in \Delta_{\frac{1}{2}}(A)$, an interpretation I satisfies $I(p) = b$ iff $I(A) = \frac{1}{2}$.*

4 The transformation $\mathcal{F}$

The definitions and results in the previous section are used in the tree transformation $\mathcal{F}$ with the aim of decreasing the complexity of the formula before distributing. Specifically, this transformation dynamically "filters" the information in the Δ-sets to avoid as much distributions of $\wedge$ wrt $\vee$ as possible.

The flow diagram of this transformation appears in Figure 2, and it is the same as in the classical case. The input of TAS-M3 is the syntactic tree of the strong negation of the formula to be analysed. After the transformation *sign*, the tree corresponds to a formula in unary normal form which is the input of the transformation $\mathcal{F}$. The execution of $\mathcal{F}$ can either reduce the size of the formula or end the execution giving the information VALID or NON-VALID.

Definition 4.1 A wff A is said to be *finalizable* if it satisfies one of the following items:

- A is one of the constants $\bot$, $\oslash$ or $\top$.
- $\Delta_{\frac{1}{2}}(A) \neq \varnothing$.
- $\Delta_1(A) \neq \varnothing$.

As a simple consequence of the definition we have the following result

Lemma 4.1 *If $A \neq \bot$ is a finalizable wff, then A is quasi-satisfiable, i.e., $\neg A$ is not valid.*

Proof. If A is either $\oslash$ or $\top$, it is immediate. If $\Delta_{\frac{1}{2}}(A) \neq \varnothing$, by Lemma 3.4, if $pb \in \Delta_{\frac{1}{2}}(A)$, then for any interpretation I satisfying $I(p) = b$ we have $I(A) = \frac{1}{2}$. Therefore, A is quasi-satisfiable. If $\Delta_1(A) \neq \varnothing$, the proof is similar.

The following section introduces the definition and theorems related to the simplify transformation

4.1 The simplify transformation

The aim of this transformation is to detect, from the information contained in the Δ sets, the existence of subformulas which are equivalent to a 3-valued literal or to the constants $\top$ or $\perp$. To formally define the transformation we need to introduce some definitions and results:

Definition 4.2

A *3-valued cube* is either a 3-valued literal or a conjunction of 3-valued literals.

A *3-valued clause* is either a 3-valued literal or a disjunction of 3-valued literals.

A 3-valued cube or clause is said to be *restricted* if it does not contain two literals with the same suffix.

A wff is said to be in *disjunctive normal form* (dnf) if it is a 3-valued cube or a disjunction of 3-valued cubes.

Remark 4.1 In the following we will use extensively the following notation:

If p is a propositional symbol, then P_3 denotes the set $\{p0, p\frac{1}{2}, p1\}$

The following theorem shows in which cases it is possible to *strongly reduce* the formula.

Theorem 4.1 *Let A be a unf:*

1. *If there exists $p \in Q$ such that $P_3 \subseteq \Delta_0(A)$, then $A \equiv \perp$.*
2. *If there exists $p \in Q$ such that $P_3 \subseteq \Delta_1(A)$, then $A \equiv \top$.*
3.

(*a*) *If*	$\Delta_0(A) = \{p0\}$,	$\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$,	$\Delta_1(A) = \{p1\}$	*then* $A \equiv p$
(*b*) *If*	$\Delta_0(A) = \{p1\}$,	$\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$,	$\Delta_1(A) = \{p0\}$	*then* $A \equiv \neg p$
(*c*) *If*	$\Delta_0(A) = \{p\frac{1}{2}, p1\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p0\}$	*then* $A \equiv J_0 p$
(*d*) *If*	$\Delta_0(A) = \{p0\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p\frac{1}{2}, p1\}$	*then* $A \equiv \neg J_0 p$
(*e*) *If*	$\Delta_0(A) = \{p0, p1\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p\frac{1}{2}\}$	*then* $A \equiv J_{\frac{1}{2}} p$
(*f*) *If*	$\Delta_0(A) = \{p_{\frac{1}{2}}\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p0, p1\}$	*then* $A \equiv \neg J_{\frac{1}{2}} p$
(*g*) *If*	$\Delta_0(A) = \{p0, p\frac{1}{2}\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p1\}$	*then* $A \equiv J_1 p$
(*h*) *If*	$\Delta_0(A) = \{p1\}$,	$\Delta_{\frac{1}{2}}(A) = \varnothing$,	$\Delta_1(A) = \{p0, p\frac{1}{2}\}$	*then* $A \equiv \neg J_1 p$

Proof.

1. If there exists $p \in Q$ such that $P_3 = \{p0, p\frac{1}{2}, p1\} \subseteq \Delta(A)$, by the last item of Theorem 3.3 we have that $A \equiv \perp$.

2. Similar to the previous one, using Theorem 3.4.

3. (a) Suppose $\Delta_0(A) = \{p0\}$, $\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$ and $\Delta_1(A) = \{p1\}$ then, by Lemmas 3.2, 3.3 and 3.4, for every interpretation I, we have that if $I(p) = b$ where $b \in \mathbf{3}$, then $I(A) = b$. Therefore $A \equiv p$.

   (c) Suppose $\Delta_0(A) = \{p\frac{1}{2}, p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ and $\Delta_1(A) = \{p0\}$ and let I be an interpretation: if $I(J_0p) = 0$ then either $I(p) = \frac{1}{2}$ or $I(p) = 1$, and by hypothesis $I(A) = 0$; if $I(J_0p) = 1$ then $I(p) = 0$ and $I(A) = 1$.

   On the other hand, given an interpretation I, from $\{p\frac{1}{2}, p1\} = \Delta_0(A)$ we have that if $I(p) \in \{0, \frac{1}{2}\}$ then $I(A) = 0$, by Lemma 3.2; and from $\{p0\} = \Delta_1(A)$ we have that if $I(p) = 1$ then $I(A) = 1$, by Lemma 3.3. Therefore $J_0p \equiv A$.

   The rest of the items can be proven similarly.

Remark 4.2 One can argue that, in the proof of item 3 above, only inclusions $\{p\,i\} \subseteq \Delta_b(A)$ are used and, thus, the statement could be easily extended to allow the strict inclusion. This greater generality is only apparent, since whenever the theorem is applied one clearly notes that no propositional symbol different from p could be in the Δ-sets.

The definition of *simplify* is closely related to the previous theorem, and it is the following

Definition 4.3 Let A be a unf, we say that

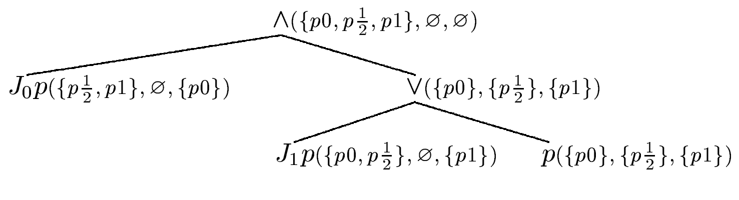
1. A is *0-conclusive* if there exists $p \in Q$ such that $P_{\mathbf{3}} \subseteq \Delta_0(A)$.

2. A is *1-conclusive* if there exists $p \in Q$ such that $P_{\mathbf{3}} \subseteq \Delta_1(A)$.

3. A is *p-simple* if it satisfies one of the conditions in item 3 of Theorem 4.1.

4. Let A be a unf, to *simplify* A is to check if A has a b-conclusive or p-simple subtree, and to apply the results in Theorem 4.1.

Example 4.1 In Example 3.4, after *labelling* the following tree was obtained:


$\wedge(\{p0, p\frac{1}{2}, p1\}, \varnothing, \varnothing)$
$J_0p(\{p\frac{1}{2}, p1\}, \varnothing, \{p0\})$
$\vee(\{p0\}, \{p\frac{1}{2}\}, \{p1\})$
$J_1p(\{p0, p\frac{1}{2}\}, \varnothing, \{p1\})$
$p(\{p0\}, \{p\frac{1}{2}\}, \{p1\})$

Note that T_B is *0-conclusive* since $P_{\mathbf{3}} \subseteq \Delta_0(B)$, thus *simplifying* the formula we get $\perp$. Therefore the input formula is valid.

4.2 The reduce transformation

The aim of this transformation is to provide more general conditions which allow to use the information in the Δ sets, that cannot be used by the *simplify* transformation: The *simplify* transformation uses the information in the Δ sets in the *strong sense*, that is, to globally substitute a formula by $\top$, $\bot$ or a 3-valued literal.

It is not always possible to use the information in the strong sense and, therefore our next objective is to be able to use this information in the *weak sense*, that is, to decrease the size of the formula obtaining another one in which the propositional symbols occurring in $(\Delta_0, \Delta_{\frac{1}{2}}, \Delta_1)$ appear at most once. This is the aim of the *reduce* transformation.

The formal description requires to extend Theorems 3.3 and 3.4; and also needs the notation below:

- $A[B/C]$ means that all the occurrences of B in A have been substituted by C.
- $A[\![l/C]\!]$ means that all the occurrences of the 3-valued literal l in A have been substituted by C.

For instance, if A is the formula $(p \wedge q \wedge J_1 p) \vee (\neg p \wedge q)$, then

$$\begin{array}{rcl} A[p/\top] & = & (\top \wedge q \wedge J_1 \top) \vee (\neg \top \wedge q) \quad \text{but} \\ A[\![p/\top]\!] & = & (\top \wedge q \wedge J_1 p) \vee (\neg p \wedge q) \end{array}$$

The following theorem extends Theorem 3.3 using information in Δ_0 in the weak sense.

Theorem 4.2 *Let A be a unf and let p be a propositional symbol:*

1. *If* $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$ *then* $A \equiv J_1 p \wedge A[p/\top]$
2. *If* $\{p0, p1\} \subseteq \Delta_0(A)$ *then* $A \equiv J_{\frac{1}{2}} p \wedge A[p/\oslash]$
3. *If* $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$ *then* $A \equiv J_0 p \wedge A[p/\bot]$
4. *If* $\{p\, b\} = \Delta_0(A) \cap P_3$ *then* $A \equiv \neg J_b p \wedge A[\![\neg J_b p/\top, J_b p/\bot]\!]$, *for* $b \in \mathbf{3}$.

Proof.

1. Suppose $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$.

   Let I be any interpretation. We can distinguish two cases:

   - If $I(p) \in \{0, \frac{1}{2}\}$, by Lemma 3.2, we have $I(A) = 0$. Therefore, $I(A) = I(J_1 p) = 0 = I(J_1 p \wedge A[p/\top])$.
   - If $I(p) = 1$ we have that $I(J_1 p) = 1$ and the behaviour of p in A wrt to I is the same than that of $\top$. Therefore, $I(A) = I(A[p/\top]) = I(J_1 p \wedge A[p/\top])$.

4. Suppose $\{p0\} = \Delta_0(A) \cap P_3$, then by Theorem 3.3 we have that $A \equiv A \wedge \neg J_0 p$. Let I be any interpretation. We can distinguish two cases:

   - If $I(p) = 0$, by Lemma 3.2, we have that $I(A) = 0$. Therefore $I(A) = I(\neg J_0 p) = 0 = I(\neg J_0 p \wedge A[\![\neg J_0 p/\top, J_0 p/\bot]\!])$.
   - If $I(p) \in \{\frac{1}{2}, 1\}$ we have that $I(\neg J_0 p) = 1$, that is, the behaviour of $\neg J_0 p$ in A wrt I is the same than that of $\top$ and the behaviour of $J_0 p$ in A wrt I is the same than that of $\bot$. Therefore $I(A) = I(A[\![\neg J_0 p/\top, J_0 p/\bot]\!]) = I(\neg J_0 \wedge A[\![\neg J_0 p/\top, J_0 p/\bot]\!])$.

The cases $b = \frac{1}{2}$ and $i = 1$ are proven in a similar manner.

Remark 4.3 Recall that the hypothesis in item 4 means that there is only one occurrence of $p\,i$ in $\Delta_0(A)$. It is worth to remark that all the information in the Δ-sets can be used somehow: If $P_{\mathbf{3}} \subseteq \Delta_0(A)$, then A is valid; if there are two occurrences of symbol p in $\Delta_0(A)$, then some of the items 1–3 can be applied; if only there is one occurrence of p in $\Delta_0(A)$, then item 4 is applied.

By duality we get Theorem 4.3 as an extension of Theorem 3.4 using information in Δ_1 in the weak sense:

Theorem 4.3 *Let A be a unf and p a propositional symbol:*

1. *If $\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$ then $A \equiv \neg J_1 p \vee A[p/\top]$*
2. *If $\{p0, p1\} \subseteq \Delta_1(A)$ then $A \equiv \neg J_{\frac{1}{2}} p \vee A[p/\oslash]$*
3. *If $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ then $A \equiv \neg J_0 p \vee A[p/\bot]$*
4. *If $\{p\,b\} = \Delta_1(A) \cap P_3$ then $A \equiv J_b p \vee A[\![J_b p/\bot, \neg J_b p/\top]\!]$, for $b \in \mathbf{3}$.*

Remark 4.4 To analyse the usefulness of the previous theorems we have to recall that TAS methods distribute $\wedge$s over $\vee$s (if not all the distributions are avoided), and consequently the following result is used:

$\bigvee_{i\in I} A_i$ is unsatisfiable if and only if A_i is unsatisfiable for all $i \in I$

Therefore the usefulness of Theorem 4.3 is clear; since it makes substitutions to decrease the size of the formula and a $\vee$ connective is taken up. On the other hand, Theorem 4.2 decreases the size of the formula but, the $\wedge$ connective which is taken up might introduce the necessity of a future $\wedge$-$\vee$ distribution.

The previous comments show that, as in the TAS methods in classical logic [2, 5], we have to restrict the use of Theorem 4.2. That is why we introduce the following definition:

Definition 4.4 Let A be a unf. If the root of T_A is $\wedge$, then

1. $\Delta_0^{lit}(A) = \{\ell \mid \ell$ is a 3-valued literal which is a child of the root of $T_A\}$

   Otherwise, if the root of T_A is not $\wedge$, then $\Delta_0^{lit}(A) = \varnothing$.
2. $\Delta_0^*(A) = \{pb \mid pb \in \Delta_0(\ell)$ where $\ell \in \Delta_0^l(A)\}$

On the one hand, the use of $\Delta_0^*(A)$ allows to use Theorem 4.2 on the literals of a conjunction, as in this case no $\wedge$ is generated. On the other hand, the use of $\Delta_0^l(A)$ allow to improve the results in Theorems 4.2 and 4.3 when literal children exist. Note that the set $\Delta_0^{lit}(A)$ is not a Δ set in the sense that its elements are not $p\,b$ but 3-valued literals; we retain the Δ notation since its elements can be used in a similar manner.

Now, the following result is an immediate consequence of Theorem 4.2:

Corollary 4.1 *Let $A = \bigwedge_{i=1}^n A_i$ be a unf and $p \in Q$:*

1. *If* $\{p0, p\frac{1}{2}\} \subseteq \Delta_0^*(A)$ *then* $A \equiv J_1 p \wedge A[p/\top]$
2. *If* $\{p0, p1\} \subseteq \Delta_0^*(A)$ *then* $A \equiv J_{\frac{1}{2}} p \wedge A[p/\oslash]$
3. *If* $\{p\frac{1}{2}, p1\} \subseteq \Delta_0^*(A)$ *then* $A \equiv J_0 p \wedge A[p/\bot]$

If there exists $p\,b$ such that $\{p\,b\} = P_3 \cap \Delta_0^*(A)$, then

4. *If* $p \in \Delta_0^{lit}(A)$ *then* $A \equiv p \wedge A[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$
5. *If* $\neg p \in \Delta_0^{lit}(A)$ *then* $A \equiv \neg p \wedge A[\![\neg p/\top, \neg J_1 p/\top, J_1 p/\bot]\!]$
6. *If* $\neg J_b p \in \Delta_0^{lit}(A)$ *then* $A \equiv \neg J_b p \wedge A[\![\neg J_b p/\top, J_b p/\bot]\!]$

Proof. The three first items are particular cases of Theorem 4.2.

4. Suppose $p \in \Delta_0^{lit}(A)$. By commutativity and associativity of $\wedge$ we have that $A \equiv p \wedge B$.

   In M3 we have the distributive property of $\wedge$ wrt $\vee$, then we can suppose without loss of generality that the unf B is in dnf, that is, $B = \bigvee_{i \in I} D_i$ where D_i is a 3-valued cube for all i.

   Thus, we have $A \equiv p \wedge B \equiv p \wedge p \wedge \bigvee_{i \in I} D_i \equiv p \wedge \bigvee_{i \in I}(p \wedge D_i)$.

   Now, for all $i \in I$, we can distinguish two cases:

   - Some of the 3-valued literals in D_i have prefix either ϵ, J_0 or $\neg J_0$. In this case, using the equivalences

     $$p \wedge p \equiv p; \qquad p \wedge J_0 p \equiv \bot; \qquad p \wedge \neg J_0 p \equiv p$$

     we have that $p \wedge D_i \equiv p \wedge D_i[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$.
   - None of the 3-valued literals in D_i have prefix either ϵ, or J_0 or $\neg J_0$. It is obvious in this case that $D_i[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!] = D_i$ and, consequently, $p \wedge D_i \equiv p \wedge D_i[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$.

   Therefore, $A \equiv p \wedge B \equiv p \wedge p \wedge B \equiv p \wedge A[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$.

The rest of the items are particular cases of Theorem 4.2.

Corollary 4.1 shows which substitutions can be used when using information from Δ_0; this will correspond to the concept of 0-reduction (to be used on proper subtrees). Moreover, Theorem 4.2 is used again in the complete 0-reduction process (to be used only on the whole tree), whose theoretical justification appears below:

Corollary 4.2 *Let A be a unf and p a propositional symbol:*

1. *If $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$, then A and $A[p/\top]$ are equi-quasisatisfiable.*
2. *If $\{p0, p1\} \subseteq \Delta_0(A)$, then A and $A[p/\oslash]$ are equi-quasisatisfiable.*
3. *If $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$, then A and $A[p/\bot]$ are equi-quasisatisfiable.*

Proof.

1. If $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$.

   If A is quasi-satisfiable, let I be an interpretation such that $I(A) \in \{\frac{1}{2}, 1\}$; by Theorem 4.2 we have $A \equiv A[p/\top] \wedge J_1 p$, then $I(A[p/\top]) \in \{\frac{1}{2}, 1\}$, that is $A[p/\top]$ is quasi-satisfiable.

   Reciprocally, if $A[p/\top]$ is quasi-satisfiable let I be an interpretation such that $I(A[p/\top]) \in \{\frac{1}{2}, 1\}$. Then, for every interpretation I' such that $I'(p) = 1$ and $I'(q) = I(q)$ for all $q \in Q - \{p\}$ we have that $I'(A) \in \{\frac{1}{2}, 1\}$, that is, A is quasi-satisfiable.

The other two items are proved similarly.

Finally, here we have an improved result wrt 1-reduction, which uses the set $\Delta_1^{lit}(A)$ whose definition is dual to that of $\Delta_0^{lit}(A)$:

Corollary 4.3 *Let $A = \bigwedge_{i=1}^{n} A_i$ be a unf and $p \in Q$:*

1. *If $\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$ then* $A \equiv \neg J_1 p \vee A[p/\top]$
2. *If $\{p0, p1\} \subseteq \Delta_1(A)$ then* $A \equiv \neg J_{\frac{1}{2}} p \vee A[p/\oslash]$
3. *If $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ then* $A \equiv \neg J_0 p \vee A[p/\bot]$

If there exists pb such that $\{p\,b\} = P_{\mathbf{3}} \cap \Delta_1(A)$, then

4. *If $b = 0$ and $\neg p \in \Delta_1^{lit}(A)$ then* $A \equiv \neg p \vee A[\![\neg p/\bot, J_0 p/\bot, \neg J_0 p/\top]\!]$
   otherwise $A \equiv J_0 p \vee A[\![J_0 p/\bot, \neg J_0 p/\top]\!]$
5. *If $b = \frac{1}{2}$ then* $A \equiv J_{\frac{1}{2}} p \vee A[\![J_{\frac{1}{2}} p/\bot, \neg J_{\frac{1}{2}} p/\top]\!]$
6. *If $b = 1$ and $p \in \Delta_1^{lit}(A)$ then* $A \equiv p \vee A[\![p/\bot, J_1 p/\bot, \neg J_1 p/\top]\!]$
   otherwise $A \equiv J_1 p \vee A[\![J_1 p/\bot, \neg J_1 p/\top]\!]$

Proof. Similar as Corollary 4.1, but use Theorem 4.3 instead.

The *reduce* transformation is based on the following definitions:

Definition 4.5 Let A be a unf which is neither a 3-valued cube nor a 3-valued clause, A is said to be:

- *0-reducible* if $\Delta_0^l(A) \neq \varnothing$.
- *1-reducible* if there exists a proper subformula B of A such that $\Delta_1(B) \neq \varnothing$.
- *completely 0-reducible* if there exists $p \in Q$ such that $\{pb_1, pb_2\} \subseteq \Delta_0(A)$ where $b_1, b_2 \in \mathbf{3}$, with $b_1 \neq b_2$.

- *reducible* iff it is *0-reducible*, *1-reducible* or *completely 0-reducible*.

Definition 4.6 Let A be a reducible unf:

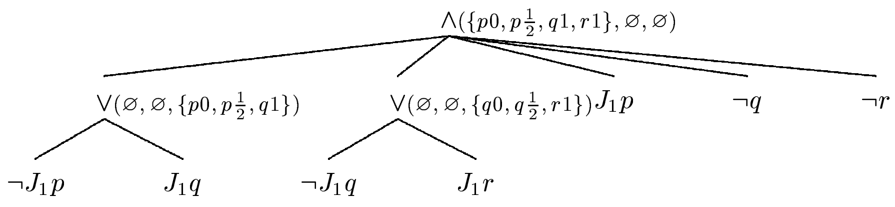
a) If T_A is completely 0-reducible, then to *reduce* T_A is to make the substitutions in Corollary 4.2 in the tree T_A.

b) Otherwise, *reduce* T_A is to traverse depth-first the tree T_A and, in the first proper 0-reducible or 1-reducible subtree T_B, to make either the substitutions in Corollary 4.1 if B is 0-reducible, or the substitutions in Corollary 4.3 if B is 1-reducible.

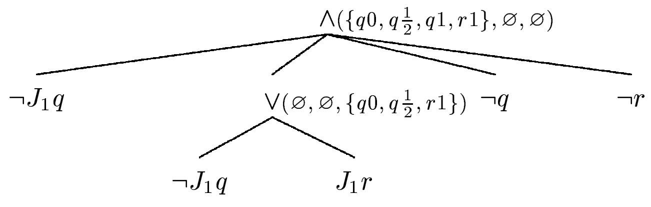
Theorem 4.4 *Let A, B be unfs such that Reduce$(A) = B$, then A is quasi-satisfiable if and only if B is quasi-satisfiable.*

Proof. Immediate consequence of Corollaries 4.1, 4.2, and 4.3.

Example 4.2 Let us consider the formula $(p \to q) \to ((p \to r) \to (p \to (q \wedge r)))$. After *signing* and *labelling* we get the following tree T_C:


$\wedge(\{p0, p\frac{1}{2}, q1, r1\}, \varnothing, \varnothing)$
$\vee(\varnothing, \varnothing, \{p0, p\frac{1}{2}, q1\})$
$\vee(\varnothing, \varnothing, \{q0, q\frac{1}{2}, r1\})$
$J_1 p$
$\neg q$
$\neg r$
$\neg J_1 p$
$J_1 q$
$\neg J_1 q$
$J_1 r$

T_C is *completely 0-reducible* since $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(C)$, then *reduce* T_C means to apply the substitution $[p/\top]$ obtaining:


$\wedge(\{q0, q\frac{1}{2}, q1, r1\}, \varnothing, \varnothing)$
$\neg J_1 q$
$\vee(\varnothing, \varnothing, \{q0, q\frac{1}{2}, r1\})$
$\neg q$
$\neg r$
$\neg J_1 q$
$J_1 r$

This tree T_D is *0-conclusive*, since $Q_3 \subseteq \Delta_0(D)$, therefore it can be *simplified* to $\perp$.

4.3 The update transformation

The *update* transformation, the first time it is applied, as seen in Figure 1, labels the tree and checks if it is finalizable (if the answer is affirmative then the method ends). When it is applied after *simplifying* or *reducing* it has to eliminate the constants $\top$, $\bot$ and/or $\oslash$ that these processes might have introduced, it also recalculates the labels of the modified nodes and their ancestors, and checks finalizability (once again, if the answer is affirmative then the method ends). This way *update* can either force the method to end or decrease the size of the analysed formula.

To eliminate the constants, *update* uses the following theorem:

Theorem 4.5

1. *For every unf A in M3 the following logical equivalences hold:*

$$\begin{array}{ll} A \vee \top \equiv \top & A \wedge \top \equiv A \\ A \vee \bot \equiv A & A \wedge \bot \equiv \bot \end{array}$$

2. *The constants $\bot$, $\oslash$ and $\top$ satisfy the following equivalences wrt the unary connectives $\neg, J_0, \neg J_0, J_{\frac{1}{2}}, \neg J_{\frac{1}{2}}, J_1, \neg J_1$:*

$$\begin{array}{llllll} \neg\bot \equiv \top & \neg\oslash \equiv \oslash & \neg\top \equiv \bot & & & \\ J_0\bot \equiv \top & J_0\oslash \equiv \bot & J_0\top \equiv \bot & \neg J_0\bot \equiv \bot & \neg J_0\oslash \equiv \top & \neg J_0\top \equiv \top \\ J_{\frac{1}{2}}\bot \equiv \bot & J_{\frac{1}{2}}\oslash \equiv \top & J_{\frac{1}{2}}\top \equiv \bot & \neg J_{\frac{1}{2}}\bot \equiv \top & \neg J_{\frac{1}{2}}\oslash \equiv \bot & \neg J_{\frac{1}{2}}\top \equiv \top \\ J_1\bot \equiv \bot & J_1\oslash \equiv \bot & J_1\top \equiv \top & \neg J_1\bot \equiv \top & \neg J_1\oslash \equiv \top & \neg J_1\top \equiv \bot \end{array}$$

3. *Let A be a unf and let $B \wedge \oslash$ be a subformula of A, then A is quasi-satisfiable if and only if $A[(B \wedge \oslash)/B]$ is quasi-satisfiable.*

4. *Let A be a unf and let $B \vee \oslash$ be a subformula of A, then A is quasi-satisfiable if and only if $A[(B \vee \oslash)/\top]$ is quasi-satisfiable.*

Proof. The proof of the equivalences is straightforward.

3. By structural induction using the fact that B is quasi-satisfiable if and only if $B \wedge \oslash$ is.

4. Similar to the previous one.

4.3.1 The output of update

If the method ends because T_B is *finalizable* then TAS-M3 can give one of the following outputs:

(i) VALID, if T_B is $\bot$.

(ii) NON-VALID otherwise.

If the input tree is not *finalizable* then it is the output of the *update* transformation, which is the input of *simplify*.

Example 4.3 Continuing with Example 3.3, the input of $\mathcal{F}$ was the syntactic tree of $J_1 p \wedge J_1 q \wedge \neg p$. When the *update* transformation is applied, it labels every node and we get the tree T_B below:

$$\begin{tikzcd}
 & \wedge(\{p0, p\frac{1}{2}, p1, q0, q\frac{1}{2}\}, \varnothing, \varnothing) \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
J_1 p(\{p0, p\frac{1}{2}\}, \varnothing, \{p1\}) & J_1 q(\{q0, q\frac{1}{2}\}, \varnothing, \{q1\}) & \neg p(\{p1\}, \{p\frac{1}{2}\}, \{p0\}) & \longmapsto \quad \bot
\end{tikzcd}$$

The obtained tree T_B is 0-conclusive since $P_3 \subseteq \Delta_0(B)$, therefore *simplify* substitutes it by $\bot$, this is *finalizable* and TAS-M3 ends with the output VALID.

Note that, as the processes *sign* and *label* can be implemented in a single traverse of the syntactic tree, the validity of the previous formula is detected by traversing the tree only once.

Theorem 4.1 shows that all the transformations executed when simplifying a tree preserve logical equivalence; Corollaries 4.1, 4.2, and 4.3 show that reducing preserves quasi-satisfiability; Lemma 4.1 shows that the finalizability test preserves quasi-satisfiability. Formally, we have:

Theorem 4.6 *Let A and B be unfs, if $\mathcal{F}(T_A) = T_B$ then A is quasi-satisfiable if and only if B is quasi-satisfiable.*

4.4 The tree-transformation ($\wedge$-$\vee$)-par

As other automated theorem provers, the complexity of TAS-M3 is exponential in the worst case; on the other hand, the reductions made by $\mathcal{F}$ solve tractable problems. Consequently, the exponential complexity of TAS-M3 is due to its last stage, thus TAS-M3 will be a good method provided this last stage has an acceptable behaviour for most inputs. In our opinion, this will only be possible if the following aims are reached:

1. Make feasible the parallel execution of non-avoidable distributions.
2. Avoid as many distributions of $\wedge$ over $\vee$ as the structure of the formula admits. For this, the minimum non-avoidable distribution is executed and then the tree-transformation $\mathcal{F}$ is used on each generated subtask to try to reduce them before a new distribution.

These two goals have led the design of the tree-transformation ($\wedge$-$\vee$)-par, which is based on the *tree-transformation* ($\wedge$-$\vee$) which is the translation in terms of syntactic trees of the generalised distributive law $A \wedge (\bigvee_1^n B_i) \equiv \bigvee_1^n (A \wedge B_i)$

Now, ($\wedge$-$\vee$)-par is described by the flow diagram in Figure 3.
rThe output of ($\wedge$-$\vee$)-par is:

- NON-VALID if some of the generated subtasks has the output NON-VALID.

- VALID if all of the generated subtasks have the output VALID.

It is important to note that the trace of the execution of ($\wedge$-$\vee$)-par can be represented by an n-ary tree, whose nodes are labelled with syntactic trees (that is, a tree of trees). The root of the execution trace tree is the input of ($\wedge$-$\vee$)-par; any node has just one child if it is the output of $\mathcal{F}$, otherwise, it has as many children as generated subtasks. Obviously, every leaf of the execution trace tree is either $\perp$ or $\top$ (if all of them are $\perp$, then the output is VALID; otherwise, if at least one leaf is $\top$, then the output is NON-VALID).

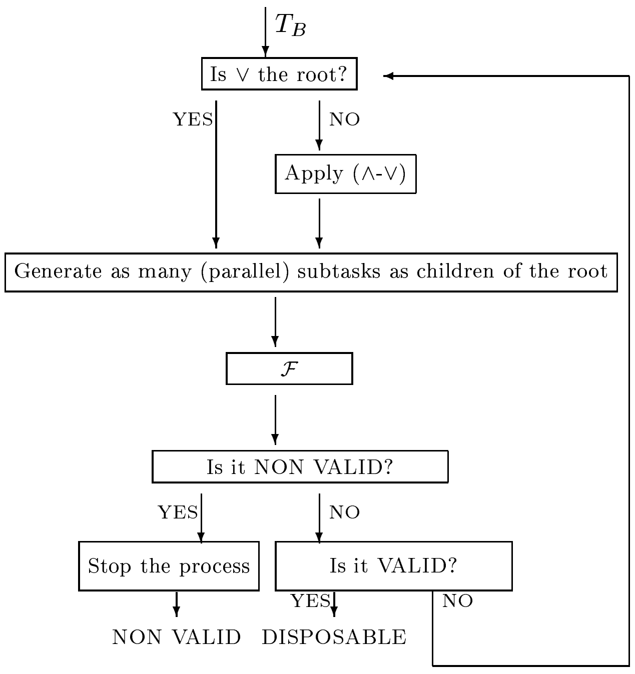
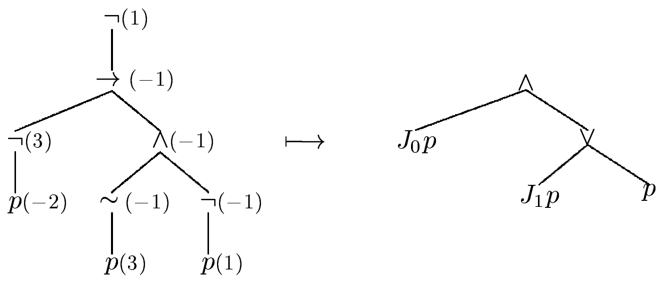


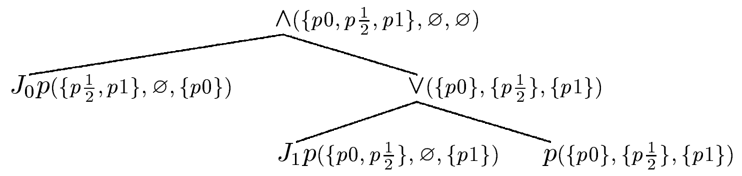
Figure 3: The tree-transformation ($\wedge$-$\vee$)-par.

5 Complete examples

Example 5.1 Given the formula $A = (\neg p \rightarrow ({\sim} p \wedge \neg p))$, taken from [7], the transformation of $\neg A$ into an equivalent unf is shown below:



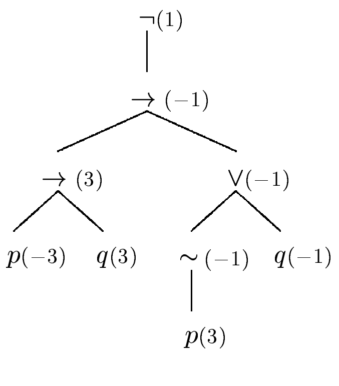
Now, the labelling of the previous tree yields the following result:


$\wedge(\{p0, p\frac{1}{2}, p1\}, \varnothing, \varnothing)$
$J_0 p(\{p\frac{1}{2}, p1\}, \varnothing, \{p0\})$
$\vee(\{p0\}, \{p\frac{1}{2}\}, \{p1\})$
$J_1 p(\{p0, p\frac{1}{2}\}, \varnothing, \{p1\})$
$p(\{p0\}, \{p\frac{1}{2}\}, \{p1\})$

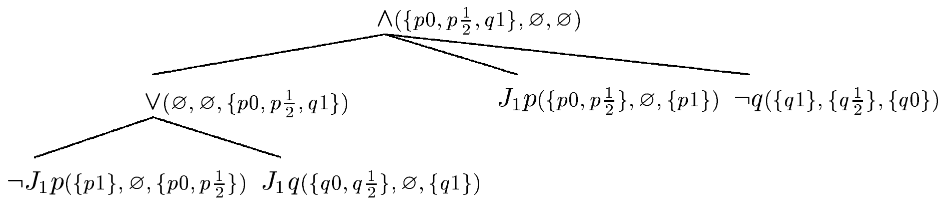
This tree T_B is 0-conclusive, since $P_{\mathbf{3}} \subset \Delta_0(T_B)$, *simplify* substitutes it by $\perp$, which is a finalizable formula. The method ends with the output VALID.

As in the previous example, the validity of the formula is detected by traversing the tree only once.

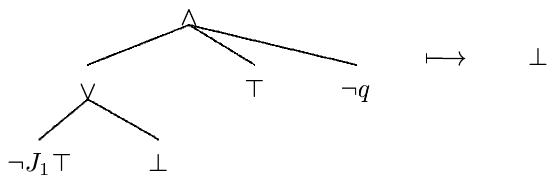
Example 5.2 Consider the formula $A = (p \rightarrow q) \rightarrow ({\sim} p \vee q)$, taken from [6], the syntactic tree of the strong negation of A is signed and labelled:



$\downarrow$


$\wedge(\{p0, p\frac{1}{2}, q1\}, \varnothing, \varnothing)$
$\vee(\varnothing, \varnothing, \{p0, p\frac{1}{2}, q1\})$
$J_1 p(\{p0, p\frac{1}{2}\}, \varnothing, \{p1\})$
$\neg q(\{q1\}, \{q\frac{1}{2}\}, \{q0\})$
$\neg J_1 p(\{p1\}, \varnothing, \{p0, p\frac{1}{2}\})$
$J_1 q(\{q0, q\frac{1}{2}\}, \varnothing, \{q1\})$

This tree T_B is not simplifiable, but reducible, specifically it is *completely 0-reducible* since $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(B)$ and it is *0-reducible* since $\neg q \in \Delta_0^l(B)$, reducing wrt both literals making the substitutions $[p/\top]$ and $[\![J_1 q/\bot]\!]$) we obtain the tree below. Finally, by updating the constants we get the tree $\bot$:



As $\bot$ is finalizable, the output of the method is that the formula is VALID.

The last example we show, includes an application of the ($\wedge$-$\vee$)-par transformation

Example 5.3 Let us consider the formula A

$$\begin{aligned}&((p \vee (r \to t)) \wedge (q \vee (t \to \neg(s \wedge \neg p))))\\ &\qquad \to ((p \wedge \neg(q \to t)) \vee (\neg r \to ((q \to (s \to r)) \wedge \sim s)))\end{aligned}$$

Because of the size of this formula, we will only sketch the trace of the method.

The tree $T_{\neg A}$ is the input of the method and after *signing* and *labelling* we obtain the tree T_B where

$$\begin{aligned}B = {}&(J_1 p \vee \neg J_1 r \vee J_1 t) \wedge (J_1 q \vee \neg J_1 t \vee J_0 s \vee J_1 p) \wedge\\ &\qquad (\neg p \vee \neg J_1 q \vee t) \wedge J_0 r \wedge ((J_1 q \wedge J_1 s \wedge \neg r) \vee J_1 s)\end{aligned}$$

The tree T_B is *simplifiable*, specifically it is *s-simple* since the label of the node corresponding with the subformula $((J_1 q \wedge J_1 s \wedge \neg r) \vee J_1 s)$ is $(\{s0, s\frac{1}{2}\}, \varnothing, \{s1\})$ so the method substitutes the corresponding subtree by $J_1 s$, thus the tree T_C is obtained where

$$C = (J_1 p \vee \neg J_1 r \vee J_1 t) \wedge (J_1 q \vee \neg J_1 t \vee J_0 s \vee J_1 p) \wedge (\neg p \vee \neg J_1 q \vee t) \wedge J_0 r \wedge J_1 s$$

The tree T_C is *reducible*, specifically, it is *completely 0-reducible* since $\{r\frac{1}{2}, r1\} \subseteq \Delta_0(C)$ and $\{s0, s\frac{1}{2}\} \subseteq \Delta_0(C)$, so the substitutions $[r/\bot]$ and $[s/\top]$ are applied in C. After removing the constants using *update* the tree T_D is obtained, where

$$D = (J_1 q \vee \neg J_1 t \vee J_1 p) \wedge (\neg p \vee \neg J_1 q \vee t)$$

This tree T_D is the output of the process $\mathcal{F}$ since it is not *simplifiable* and not *reducible*. Now, the process ($\wedge$-$\vee$)-par is applied, generating three tasks corresponding to the formulas

$$\begin{array}{rcl} E & = & J_1 q \wedge (\neg p \vee \neg J_1 q \vee t) \\ F & = & \neg J_1 t \wedge (\neg p \vee \neg J_1 q \vee t) \\ G & = & J_1 p \wedge (\neg p \vee \neg J_1 q \vee t) \end{array}$$

when labelling the tree T_E, its root is labelled with $(\{q0, q\frac{1}{2}\}, \varnothing, \{q1\})$. As $\Delta_1(E) \neq \varnothing$, T_E is *finalizable* and its output is $\top$, that is, is NON-VALID.

As one of the tasks ends with the output NON-VALID then TAS-M3 ends with the output NON-VALID.

6 Flexibility and efficiency of TAS-M3

The presented theorem prover can be extended to arbitrary three-valued logics; and this can be done easily, by only modifying the Sign transformation, since the rest of the prover is logic-independent.

6.1 A straight generalisation to any three-valued logic

As any (finite) many-valued logic can be expressed in Signed Logic [3, 9, 10] with only polynomial increasing of the formula [8]; by extending TAS-M3 to Signed Logic we will show the generality of our approach. The main, and only, point is to redefine the Sign transformation for a signed many-valued formula $S{:}\theta(A_1, \dots, A_n)$, where S is a set of truth values (i.e. a sign) and θ is any connective.

It is known that any such formula can be expressed in the form $\bigvee_i \bigwedge_j (S_{i_j}{:}A_{i_j})$ for suitable signs S_{i_j}, obtained from a tableau rule for $S{:}\theta(A_1, \dots, A_n)$. By using this transformation the literals would be of the form $S{:}p$. Now, the definition of the Δ sets can be done by using the sign S as follows:

$$\Delta_0(S{:}p) = \{p\,i \mid i \notin S\} \qquad \Delta_1(S{:}p) = \{p\,i \mid i \in S\}$$

being the definition for $\wedge$ and $\vee$ the same as before.

The rest of the sections remain unchanged, we have only to keep in mind the equivalences between our 3-valued literals in the set Λ and the signed literals $S{:}p$. It is straightforward to obtain the following correspondence

$$\begin{array}{rcl@{\qquad}rcl} \{0\}{:}p & \leftrightsquigarrow & J_0 p & \{1\}{:}p & \leftrightsquigarrow & J_1 p \\ \{\frac{1}{2}, 1\}{:}p & \leftrightsquigarrow & \neg J_0 p & \{0, \frac{1}{2}\}{:}p & \leftrightsquigarrow & \neg J_1 p \\ \{\frac{1}{2}\}{:}p & \leftrightsquigarrow & J_{\frac{1}{2}} p & \varnothing{:}p & \leftrightsquigarrow & \bot \\ \{0, 1\}{:}p & \leftrightsquigarrow & \neg J_{\frac{1}{2}} p & S{:}p & \leftrightsquigarrow & \top \end{array}$$

and we have now a theorem prover for Signed Logic.

It is worth to notice that in the description of the method we have used two more literals (p and $\neg p$) and the constant $\oslash$; this is due to our use of the strong negation $\neg$ to refute the input formula. If we used the weak negation $\sim$ then our set of literals and constants would be in one to one correspondence with the signed literals, and all the $\Delta_{\frac{1}{2}}$ sets would be empty.

6.2 On the efficiency of TAS-M3

On the one hand, the determination of which simplification or reduction could be applied to a certain node in the tree is an easy matter: the labels of each node are calculated just once at the beginning of the method by traversing the tree only once; when a reduction is applied only the labels corresponding to the ascendants of the modified nodes are calculated. As a consequence, although there are many potentially applicable rules, that one which is to be applied is dynamically and univocally selected by the reading of the label of the node.

On the other hand, although we have not presented a complexity analysis of the algorithm, it is clear that all processes involved in TAS-M3, but the last one, have at most polynomial complexity. Obviously, this fact does not guarantee a good performance of the method; but efficiency is achieved because we consider the philosophy of making a distribution only in the worst case, when no more reductions are possible.

7 Conclusions

We have presented a new *reduction-based* ATP for propositional 3-valued logics which is an extension of the TAS-D prover for classical propositional logic. The use of reductions, as in every TAS method, allows to filter the information contained in the syntactic structure of the formula to avoid as much distributions (of $\wedge$ wrt $\vee$ in our case) as possible, in order to improve efficiency; this filtering of syntactic information is the key of the efficiency of the TAS methods. In our opinion, the main advantage of our method wrt others is the use of the reductions to avoid distributions.

On the other hand, it is shown the adaptability of the TAS reduction method, because switching to different kinds of logic is possible without having to redesign the structure of the whole prover. Indeed, the flow diagram of TAS-M3 is the same than that of TAS-D, only some modules have to be adapted.

Acknowledgements

We would like to acknowledge Daniele Mundici for his great interest in the TAS methodology and for encouraging the authors to extend it to many-valued logics. We also thank the referees for many helpful suggestions.

References

[1] G. Aguilera. *Reducciones totales y parciales para el análisis de validez y construcción de modelos en M3.* PhD thesis, Dept. Matemática Aplicada. Univ. de Málaga, 1997.

[2] G. Aguilera, I. P. de Guzmán, M. Ojeda. Increasing the efficiency of automated theorem proving. *Journal of Applied Nonclassical Logics* 5(1):9–29, 1995.

[3] M. Baaz, C. G. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation*, 19(4):353–391, April 1995.

[4] M. Enciso, I. P. de Guzmán. A new and complete theorem prover for temporal logic. In *Proceedings of the IJCAI Workshop on Executable Temporal Logics*, Montreal, 1995.

[5] I. P. de Guzmán, M. Ojeda. TAS methods in first-order logic. In *Proceedings of Logic Colloquium'96*, San Sebastián. To appear in the Bulletin of Symbolic Logic, 1997.

[6] W.A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *Journal of Symbolic Logic* 52(2):473–493, 1987.

[7] R. Hähnle. *Automated deduction in multiple-valued logics.* Oxford University Press, 1994.

[8] R. Hähnle. Short conjunctive normal forms in finitely valued logics. *Journal of Logic and Computation* 4(6):905–927, 1994.

[9] N. Murray, E. Rosenthal. Signed formulas: A liftable meta logic for multiple-valued logics. In *Proceedings ISMIS'93, Trondheim, Norway.* LNCS 689, pages 275–284. Springer-Verlag, 1993.

[10] N. Murray, E. Rosenthal. Adapting classical inference techniques to multiple-valued logics using signed formulas. *Fundamenta Informaticae,* 21(3):237–253, 1994.

[11] J.B. Rosser, A.R. Turquette. *Many-valued logics.* North Holland, 1952.

[12] A. Urquhart. Many-valued logics. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. III: Alternatives in Classical Logic*, chapter 2, pages 71–116. Reidel, Dordrecht, 1986.