Rend. Istit. Mat. Univ. Trieste
Volume 42 (2010), 165–190.

Solvable Set/Hyperset Contexts: III. A Tableau System for a Fragment of Hyperset Theory

Domenico Cantone, Massimo Felici
and Marianna Nicolosi Asmundo

Abstract. *We propose a decision procedure for a fragment of the hyperset theory,* HMLSS*, which takes inspiration from a tableau saturation strategy presented in [3] for the fragment* MLSS *of well-founded set theory. The procedure alternates deduction and model checking steps, driving the correct application of otherwise very liberal rules, thus significantly speeding up the process of discovering a satisfying assignment of a given* HMLSS*-formula or proving that no such assignment exists.*

Keywords: Decision Procedures, Tableau Calculus, Multi-Level Syllogistic, Computable Set Theory, Hyperset Theory.
MS Classification 2010: 03B25, 03B35, 03E70, 68T15

1. Introduction

We present a result in the field of computable set theory relative to a tableau-based decision procedure for the satisfiability problem for a fragment of the hyperset theory. Specifically, the theory of our interest, HMLSS (Hyper Multi-Level Syllogistic with Singleton), is a semantic generalization to the universe of hypersets of the unquantified fragment MLSS, studied in the context of well-founded sets [7].

The satisfiability problem for HMLSS consists in determining for any given HMLSS-formula φ whether there exists an assignment of hypersets to the variables of φ such that the resulting interpreted formula expresses a true statement about hypersets.

The decision problem for this fragment of hyperset theory has been already solved in [8] by means of a unification algorithm, rather than by a tableau calculus. Later, in [6], a unification algorithm for the blended case has been provided. In [9], a tableau based decision procedure is presented for a language containing the operators and relators of HMLSS with the exception of the singleton operator, $\{\cdot\}$, plus a weak form of the powerset construct. Then, in [10],

a tableau-like decision procedure is introduced for a set-theoretical fragment admitting a restricted universal quantification, the powerset operator, but not including the singleton operator. It is shown that the decidability result permits to characterize several decidable modal logics such as K, T, $S4$, $S5$.

In this paper we propose a decision procedure for HMLSS that is inspired to a tableau calculus and relative saturation strategy presented in [3] for the MLSS fragment. Specifically, our tableau calculus contains rules for propagating positive and negative membership relations as well as equalities. Among the rules of the latter type, we mention one which allows one to deduce equalities between variables which are bisimilar in a suitable membership graph.

Roughly speaking, the overall behaviour of the procedure is the following. When a non-closed tableau branch is selected, a partial assignment based on the literals contained in it is constructed. If such an assignment already satisfies all literals in the branch, the root formula is declared to be satisfiable; otherwise, based on one of the non-satisfied literals, a suitable tableau rule is selected and applied to the branch. The interplay between model checking and deduction steps speeds up considerably the process of discovering a satisfying assignment, if any exists, or proving that the given HMLSS-formula is unsatisfiable.

The paper is organized as follows. In Section 2 we recall some elementary notions on hypersets according to [2]. In Section 3 we introduce the HMLSS fragment, its syntax and semantics, based on hypersets, and we extend to the hyperset case the notion of realization, reviewing some of its properties which will be used in the rest of the paper. Section 4 describes the tableau calculus for HMLSS together with the related decision procedure in the form of a terminating tableau saturation strategy. In Section 5 we prove the total correctness of the decision procedure, in Section 6 we give some examples of tableaux constructed with the procedure presented and, finally, in Section 7 we draw our conclusions.

2. Preliminaries on Hypersets

Hypersets have been represented by Barwise in [2] as elements of the solution set of flat systems of equations and by Aczel in [1] by the notion of pointed graphs. In this paper we mainly refer to the former representation. For this reason we briefly recall it in what follows, together with some basic notions that will be used later. We refer the reader to [2] for further details.

Definition 2.1 (Flat system of equations).

1. *A* flat system of equations *is a tuple* $\mathcal{E} = (X, A, e)$ *consisting of two disjoint sets* X *and* A*, and of a function* $x \mapsto e_x$ *from* X *into* $\mathscr{P}(X \cup A)$.
2. X *is called the set of* indeterminates *of* $\mathcal{E}$*, and* A *is called the set of* atoms *of* $\mathcal{E}$*. For every* $x \in X$*,* $e_x \cap X$ *is the set of indeterminates on*

which x depends immediately. Analogously, $e_x \cap A$ is the set of atoms on which x depends immediately.

3. *A* solution *to $\mathcal{E}$ is a function s with domain X satisfying every equation*

$$s_x = \{s_y : y \in (e_x \cap X)\} \cup (e_x \cap A). \qquad \square$$

The anti-foundation axiom, in the form presented in [2, Chapter 6, Section 2], also called the (Flat) Solution Lemma, states that every flat system of equations has a unique solution. Thus, we can designate the *solution set* of any flat system $\mathcal{E} = (X, A, e)$ as

$$solution\_set(\mathcal{E}) =_{\text{Def}} \{s_x : x \in X\},$$

where s is the solution to $\mathcal{E}$.

The universe of hypersets is constituted by all the sets which are in the solution set of some flat systems of equations. We denote by $\mathcal{V}_H$ the universe of hypersets.

2.1. Bisimulations and Bisimilarity

The axiom of extensionality does not always help in deciding if two given hypersets are identical. In what follows we introduce the notions of bisimulation relation and of bisimilarity for hypersets together with the statement of a theorem proved in [2], informally saying that bisimilar hypersets have to be considered as being identical.

DEFINITION 2.2. *A* bisimulation relation over hypersets *is a binary relation B over hypersets satisfying the following condition. If uBv, then*

1. *for every $u' \in u$, there is a $v' \in v$ such that $u'Bv'$,*
2. *for every $v' \in v$, there is a $u' \in u$ such that $u'Bv'$,*
3. *the set of atoms in u is equal to the set of atoms in v.*

Hypersets u and v are bisimilar *if there is a bisimulation relation B over hypersets such that uBv.* $\square$

THEOREM 2.3 ([2]). *The identity relation between hypersets, $\sim_H$, is the largest bisimulation over hypersets, that is:*

1. *$\sim_H$ is a bisimulation,*
2. *if B is a bisimulation relation over hypersets, then B is a subrelation of $\sim_H$. That is, if uBv, then $u \sim_H v$.* $\square$

Identity of hypersets can be checked by employing the representation of hypersets as flat systems of equations. For this purpose we give the notions of bisimulation relation and of bisimilarity for flat systems and report an algorithm that can be applied to check bisimilarity (i.e., identity) of hypersets in case they are in the solution set of some finite flat systems of equations.

DEFINITION 2.4. *Let $\mathcal{E} = (X, A, e)$ and $\mathcal{E}' = (X', A', e')$ be two flat systems of equations.*

A bisimulation relation *between $\mathcal{E}$ and $\mathcal{E}'$ is a relation B over $X \times X'$ such that the following conditions hold:*

(a) If xBx', then

- *for each indeterminate $y \in (e_x \cap X)$ there is an indeterminate $y' \in (e'_{x'} \cap X')$ such that yBy';*
- *for each indeterminate $y' \in (e'_{x'} \cap X')$ there is an indeterminate $y \in (e_x \cap X)$ such that yBy'.*

(b) If xBx', then e_x and $e'_{x'}$ contain the same atoms.

$\mathcal{E}$ and $\mathcal{E}'$ are bisimilar *if there is a bisimulation relation between them such that:*

(a) for every $x \in X$ there is an $x' \in X'$ such that xBx';

(b) for every $x' \in X'$ there is an $x \in X$ such that xBx'. □

In [2] the following result is proved.

THEOREM 2.5. *Let $\mathcal{E}$ and $\mathcal{E}'$ be flat systems of equations. They have the same solution set if and only if they are bisimilar.* □

Next, it is convenient to introduce the following notion. A *pointed flat system of equations* is a tuple $\mathcal{E}_x = (X, A, e, x)$, where $\mathcal{E} = (X, A, e)$ is a flat system and $x \in X$. Intuitively, pointed flat systems are flat systems focusing on an element of particular interest. Two pointed flat systems $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are bisimilar if and only if there is a bisimulation relation B between $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ such that $x_1 B x_2$.

THEOREM 2.6. *Let $\mathcal{E} = (X, A, e)$ and $\mathcal{E}' = (X', A', e')$ be two flat systems of equations with solution s and s', respectively. Then, the pointed flat systems $\mathcal{E}_{x_1} = (X, A, e, x_1)$ and $\mathcal{E}'_{x_2} = (X', A', e', x_2)$ are bisimilar if and only if $s_{x_1} = s'_{x_2}$.*

Proof. Let us prove the sufficient condition first. Consider the relation $B \subseteq X \times X'$ such that xBy if and only if $s_x = s'_y$, for every $x \in X$ and $y \in X'$.

We prove that B is a bisimulation for $\mathcal{E}$ and $\mathcal{E}'$ by showing that conditions (a) and (b) of Definition 2.4 hold.

(a) Suppose that xBy. Then for every $u \in e_x \cap X$, $s_u \in s_x$ and, by definition of B, $s_u \in s'_y$. Thus, there is a $v \in e'_y \cap X'$ such that $s_u = s'_v$, that is such that uBv. Analogously, it can be shown that for every $v \in e'_y \cap X'$ there is an $u \in e_x \cap X$ such that uBv.

(b) Let xBy and suppose by contradiction that there is an atom $a \in s_x$ such that $a \notin s'_y$. But then $s_x \neq s'_y$, which contradicts the hypothesis.

Since B is a bisimulation for $\mathcal{E}$ and $\mathcal{E}'$, it is also a bisimulation for $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$. By hypothesis $s_{x_1} = s'_{x_2}$, thus $x_1 B x_2$, and therefore $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are bisimilar.

The necessary condition is proved as follows. Let B be a bisimulation for $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ such that $x_1 B x_2$. Then, B is a bisimulation for $\mathcal{E}$ and $\mathcal{E}'$ too. We prove that $s_{x_1} = s'_{x_2}$ by showing that $s_x = s'_y$ for every $x \in X$ and $y \in X'$ such that xBy. For this purpose, we construct a flat system $\mathcal{E}^*$ whose set of indeterminates, X^*, is constituted by the pairs (x, y), with $x \in X$ and $y \in X'$, such that xBy. For every $(x, y) \in X^*$ we define

$$e^*_{(x,y)} = \{(u, v) \in X^* : u \in e_x \text{ and } v \in e'_y\} \cup (e_x \cap A).$$

Notice that putting $(e_x \cap A)$ instead of $(e'_y \cap A')$ as right term of the union does not cause loss of generality. In fact, since xBy, $(e_x \cap A)$ and $(e'_y \cap A')$ are identical. Let us consider the following candidate solutions to $\mathcal{E}^*$:

$$\begin{aligned} s^a_{(x,y)} &= s_x, \\ s^b_{(x,y)} &= s'_y, \end{aligned}$$

for every $(x, y) \in X^*$. Thanks to the fact that B is a bisimulation, we prove that these are both solutions of $\mathcal{E}^*$. We limit ourselves to showing only that s^a is a solution of $\mathcal{E}^*$, as the proof for s^b is very similar.

Let us assume that (x, y) is an indeterminate in X^*. We have to show that

$$s^a_{(x,y)} = \{s^a_{(u,v)} : (u, v) \in e^*_{(x,y)}\} \cup (e_x \cap A). \tag{1}$$

Let us take a $c \in s^a_{(x,y)}$. Since $s^a_{(x,y)} = s_x$, then $c \in s_x$, and therefore, either $c \in (e_x \cap A)$, or it is of the form s_{w_1}, where $w_1 \in (e_x \cap X)$. In the first case c plainly belongs to the right part of (1). In the second case there must be some $w_2 \in (e'_y \cap X')$ such that $w_1 B w_2$. This means that $(w_1, w_2) \in X^*$ (i.e., $(w_1, w_2) \in e^*_{(x,y)}$). Thus $c = s_{w_1} = s^a_{(w_1,w_2)}$ belongs to the right part of (1) too.

Now, let v be an element belonging to the right part of (1). If $v \in (e_x \cap A)$, then $v \in s_x = s^a_{(x,y)}$ and we are done. If v is equal to an element $s^a_{(w_1,w_2)}$ from the left part of the union in (1), where $w_1 \in e_x$ and $w_2 \in e'_y$, we have to prove that $s^a_{(w_1,w_2)} \in s^a_{(x,y)}$. This can be easily verified by noting that $s^a_{(w_1,w_2)} = s_{w_1}$ and $s_x = s^a_{(x,y)}$. Moreover $s_{w_1} \in s_x$ holds, since s is a solution of $\mathcal{E}$.

Thus s^a and s^b are both solutions of $\mathcal{E}^*$. By the antifoundation axiom $s^a = s^b$, but then, for every $(x, y) \in B$, $s_x = s^a_{(x,y)} = s^b_{(x,y)} = s'_y$. This holds in particular for the pair (x_1, x_2) which belongs to B by hypothesis. □

Let h and h' be two hypersets in the solution sets of some finite flat systems of equations. An algorithm to verify if they are identical is described next.

We put $h = s_{x_1}$ and $h' = s'_{x_2}$, where $x_1 \in X$, $x_2 \in X'$, and s and s' are the solutions of the flat systems of equations $\mathcal{E} = (X, A, e)$ and $\mathcal{E}' = (X', A', e')$, respectively.

Then, let us construct a relation $\nsim \subseteq X \times X'$ by executing the following steps until the condition $\nsim_{n+1} = \nsim_n$ becomes true:

Step 1: Let $\nsim_1$ be the set of all pairs $(x, y) \in X \times X'$ such that either e_x and e'_y differ on some atoms or only one of them is nonempty.

Step $n + 1$: Given $\nsim_n$, let P_n be the set of all the pairs (x, y) such that either

- there is an $u \in (e_x \cap X)$ such that $u \nsim_n v$, for all $v \in (e'_y \cap X')$, or
- there is a $v \in (e'_y \cap X')$ such that $u \nsim_n v$, for all $u \in (e_x \cap X)$.

Then we put $\nsim_{n+1} = \nsim_n \cup P_n$.

We call the resulting relation $\nsim$ and call its complement $\sim$. It can easily be proved that $\sim$ is a bisimulation relation for $\mathcal{E}$ and $\mathcal{E}'$. In fact, for every $x \in X$ and $y \in X'$ it holds that

- if $x \sim y$, then for every $u \in (e_x \cap X)$ there is a $v \in (e'_y \cap X')$ such that $u \sim v$, and for every $v \in (e'_y \cap X')$ there is a $u \in (e_x \cap X)$ such that $u \sim v$ (otherwise $x \nsim y$);
- moreover, if $x \sim y$, then the set of atoms in e_x is equal to the set of atoms in e'_y (otherwise $x \nsim y$).

This is enough to prove that $\sim$ is a bisimulation relation for $\mathcal{E}$ and $\mathcal{E}'$. Clearly, if $x_1 \sim x_2$, then the pointed flat systems $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are bisimilar.

It can be shown that $\sim$ coincides with the maximal bisimulation ($\sim_H$) between $\mathcal{E}$ and $\mathcal{E}'$. The proof is carried out by induction over n by showing that if $x_1 \nsim_n x_2$, then $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are not bisimilar. In addition, if $\nsim_n = \nsim_{n+1} = \nsim$, for all $x \sim y$, then $\sim$ is a bisimulation relation between $\mathcal{E}_x$ and $\mathcal{E}'_y$ and $s_x = s'_y$.

3. The HMLSS Fragment

3.1. Syntax

The HMLSS language involves:

(i) a countably infinite collection of hyperset variables $Vars = \{x, y, z, \ldots\}$;

(ii) the set predicate symbols $\in$ (membership), $=$ (equality), $\subseteq$ (set inclusion);

(iii) the binary set operator symbols $\cap$ (intersection), $\cup$ (union), $\setminus$ (set difference), $\{\cdot\}$ (singleton);

(iv) the null set constant $\boldsymbol{\emptyset}$ (characterized by the relation $\boldsymbol{\emptyset} = \boldsymbol{\emptyset} \setminus \boldsymbol{\emptyset}$);

(v) the constant $\boldsymbol{\Omega}$ (characterized by the relation $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}\}$);

(vi) parentheses (to construct compound terms);

(vii) the propositional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ (to construct compound formulae).

HMLSS-terms and formulae are constructed in the standard way. Notice that HMLSS does not admit any explicit quantification. Notice also that the collection of predicate, operator, and constant symbols we have adopted for HMLSS is not minimal, resulting in a more natural language.

3.2. Semantics

HMLSS-formulae are interpreted by means of assignments in the hypersets universe. A (hyperset) *assignment* is a function $M : \mathit{Vars} \rightarrow \mathcal{V}_H$. By the notions introduced before, it can easily be shown that assignments extend naturally to hyperset terms like $x \cap y$ or $x \setminus y$ and that they evaluate atomic formulae of type $x \in y$, $x \subseteq y$, $x = y$ to a truth value **true** or **false** in the usual way. Finally, evaluation of compound formulae plainly follows the standard rules of propositional logic.

Let M be an assignment and let φ be a formula of HMLSS. We say that M *satisfies* φ if M evaluates φ to **true**. In this case M is said to be a *model* for φ. A formula φ of HMLSS is *satisfiable* if it has a model. Two formulae φ and ψ are *equisatisfiable*, when φ is satisfiable if and only if ψ is also satisfiable.

The *decision problem* for HMLSS is then the problem of establishing for any given formula of HMLSS whether or not it is satisfiable.

3.2.1. Realizations

An important tool for constructing candidate models is represented by *realizations*.

DEFINITION 3.1 (Realization). *Let $G = (N, \hat{\in})$ be a directed graph, let (V, T) be a partition of N, and let $\{v_t : t \in T\}$ be a collection of sets. The realization of G relative to (V, T) and to $\{v_t : t \in T\}$ is the unique assignment R over $V \cup T$ satisfying the following equations:*

- $Rx = \{Rz : z \in V \cup T \text{ and } z \hat{\in} x\}$, *for* $x \in V$,
- $Rt = \{v_t\}$, *for* $t \in T$. □

Given $G = (N, \hat{\in})$, (V, T), and $\{v_t : t \in T\}$ as in Definition 3.1, for $x \in V$, let us put $G(x) =_{\text{Def}} \{y \in N : y \hat{\in} x\}$ and define the function $G^e : N \to \mathscr{P}(N \cup \{v_t : t \in T\})$ as follows:

- $G^e(x) = G(x)$, for $x \in V$,
- $G^e(t) = \{v_t\}$, for $t \in T$.

Then, consider the flat system of equations

$$\overline{\mathcal{E}} = (V \cup T, \{v_t : t \in T\}, G^e).$$

Plainly, the realization R of G relative to (V, T) and to $\{v_t : t \in T\}$ is the solution of $\overline{\mathcal{E}}$ and $solution\_set(\overline{\mathcal{E}}) = \{Rx : x \in V \cup T\}$. Using the same technique described in the algorithm of Section 2.1, we can construct from $\overline{\mathcal{E}}$ the relation $\nsim_R \subseteq (V \cup T) \times (V \cup T)$ whose complement, $\sim_R$, is such that $x \sim_R y$ if and only if the pointed flat systems of equations $\overline{\mathcal{E}}_x = (V \cup T, \{v_t : t \in T\}, G^e, x)$ and $\overline{\mathcal{E}}_y = (V \cup T, \{v_t : t \in T\}, G^e, y)$ are bisimilar, that is if and only if $Rx = Ry$, for every $x, y \in V \cup T$.

The construction of the flat system $\overline{\mathcal{E}}$ from V, T, $\{v_t : t \in T\}$, and G^e is basilar to the design of one of the rules of the calculus for HMLSS to be presented in the next section.

The following result on realizations will be useful in the correctness proof of our decision procedure.

LEMMA 3.2. *Let $G = (V \cup T, \hat{\in})$ be a directed graph, with $V \cap T = \emptyset$. Moreover, let $\{v_t : t \in T\}$ be a collection of sets, let R be the realization of G relative to (V, T) and to $\{v_t : t \in T\}$, and let us assume that:*

(a) $v_t \neq v_{t'}$, for all distinct t, t' in T,

(b) $v_t \neq Rx$, for all $t \in T$ and $x \in (V \cup T)$.

Then,

(i) $|\{x \in (V \cup T) : x \sim_R t\}| = 1$, for every $t \in T$,

(ii) if $G(x) = G(y) \cup G(z)$, then $Rx = Ry \cup Rz$, for $x, y, z \in V$,

(iii) if $G(x) = G(y) \cap G(z)$, then $Rx = Ry \cap Rz$, for $x, y, z \in V$,

(iv) if $G(x) = G(y) \setminus G(z)$, then $Rx = Ry \setminus Rz$, for $x, y, z \in V$,

(v) if $\{y_1, \ldots, y_k\} \subseteq G(x) \subseteq \bigcup_{i=1}^{k} \{y \in V : y \sim_R y_i\}$, then $Rx = \{Ry_1, \ldots, Ry_k\}$, for $x, y_1, \ldots, y_k \in V$.

(1) $x \in y$	(2) $x \notin y$	(3) $x = y$
(4) $x \neq y$	(5) $z = x \cup y$	(6) $z = x \cap y$
(7) $z = x \setminus y$	(8) $x = \{y_1, \ldots, y_k\}$	

Table 1: Normalized HMLSS-literals.

Proof. (i) follows immediately from assumptions (a) and (b) and from the very definitions of the realization R and of the relation $\sim_R$.

Concerning (ii), at first we prove that $Rx \subseteq Ry \cup Rz$. Let $u \in Rx$, then, by definition of R, $u = Rw$, for some $w \in V \cup T$ such that $w\hat{\in}x$. By definition of $G(x)$, $w \in G(x)$ and, by hypothesis, either $w\hat{\in}y$ or $w\hat{\in}z$. Thus, by definition of R, either $Rw \in Ry$ or $Rw \in Rz$, and finally $Rw \in Ry \cup Rz$, that is $u \in Ry \cup Rz$. To prove the converse, $Ry \cup Rz \subseteq Rx$, let us pick a $u \in Ry \cup Rz$ and suppose, without loss of generality, that $u \in Ry$. Then, by definition of R, there is a $w \in V \cup T$ such that $Rw = u$ and $w\hat{\in}y$. Thus $w \in G(y)$ and hence, by hypothesis, $w \in G(x)$. By definition of $G(x)$, $w\hat{\in}x$, and finally $Rw \in Rx$, that is $u \in Rx$, yielding (ii).

(iii) and (iv) can be proved in a similar way.

Concerning (v), assume that $\{y_1, \ldots, y_k\} \subseteq G(x) \subseteq \bigcup_{i=1}^{k} \{y \in V : y \sim_R y_i\}$. We need to show that $Rx = \{Ry_1, \ldots, Ry_k\}$. Let $u \in Rx$. Then, by definition of R, $u = Rw$, for some $w \in V \cup T$ such that $w\hat{\in}x$. Thus $w \in G(x)$ and therefore, for some $i = 1, 2, \ldots, k$, either $w = y_i$ or $w = y$, for some y such that $y \sim_R y_i$. In both cases $w \sim_R y_i$ and thus $Rw = Ry_i$, for some $i = 1, 2, \ldots, k$. Hence $Rx \subseteq \{Ry_1, \ldots, Ry_k\}$. Conversely, let $u \in \{Ry_1, \ldots, Ry_k\}$. Then $u = Ry_i$. By hypothesis $y_i\hat{\in}x$ and thus $Ry_i \in Rx$. Since $u = Ry_i$ we obtain $\{Ry_1, \ldots, Ry_k\} \subseteq Rx$ which, together with the previously established inverse inclusion, yields $Rx = \{Ry_1, \ldots, Ry_k\}$, completing the proof of (v). □

4. A Tableau Calculus for HMLSS

In view of a normalization procedure of the type described for the MLSS context in [3] (see Figure 1 below), without any loss of generality we can limit our considerations to flat HMLSS-conjunctions, namely HMLSS-formulae that are conjunctions of literals of the special forms illustrated in Table 1 (normalized HMLSS-literals).

Indeed, let $\langle \varphi_1, \ldots, \varphi_k \rangle$ be the k-tuple returned by *Normalize*(φ), for a given HMLSS-formula φ. Then it is easy to prove that every φ_i is a flat HMLSS-conjunction and that φ is satisfiable if and only if at least one φ_i is satisfiable.

procedure *Normalize*(φ)
– φ *is a* HMLSS-*formula.*
1. $\Phi := true$;
2. $\psi := \varphi$;
3. let q_0 and q_1 be two new variables not occurring in φ;
4. $\Phi := \Phi \wedge q_0 = q_0 \setminus q_0 \wedge q_1 = \{q_1\}$;
5. replace in ψ each occurrence of the constant $\emptyset$ by q_0
and each occurrence of $\mathbf{\Omega}$ by q_1;
6. **while** ψ contains terms of the form $x \cup y$, $x \cap y$, $x \setminus y$, $\{y_1, \ldots, y_k\}$ **do**
7. let t be any such term and let x_t be a newly introduced variable;
8. replace in ψ each occurrence of the term t by the variable x_t;
9. $\Phi := \Phi \wedge (x_t = t)$;
10. **end while**;
11. let $\psi_1 \vee \ldots \vee \psi_k$ be a disjunctive normal form of ψ;
– *At this point each* ψ_i *is a conjunction of literals*
– *of the form* $x \in y$, $x = y$, $x \subseteq y$, *or their negations.*
12. **for** $i = 1$ to k **do**
13. **for** each conjunct of type $\neg(x \subseteq y)$ in ψ_i **do**
14. let z_{xy} be a newly introduced variable;
15. replace in ψ_i each occurrence of $\neg(x \subseteq y)$ by $(z_{xy} \in x \wedge z_{xy} \notin x)$;
16. **end for**;
17. **for** each conjunct of type $x \subseteq y$ in ψ_i **do**
18. replace in ψ_i each occurrence of $x \subseteq y$ by $y = x \cup y$;
19. **end for**;
20. **end for**;
21. **return** $\langle \psi_1 \wedge \Phi, \ldots, \psi_k \wedge \Phi \rangle$
end procedure

Figure 1: The normalization procedure.

Let S be a finite collection of normalized HMLSS-literals. An *initial* HMLSS-tableau for S is a tree with just one branch whose nodes are labelled with the literals in S.

An HMLSS-tableau for S is a tableau labelled with normalized HMLSS-literals that can be constructed from the initial tableau for S by means of a finite number of applications of the rules illustrated in Table 2.

We assume that no literal can occur more than once on any given branch; that is we assume that a rule adding a literal which is already on the branch has no effect.

Let $\mathcal{T}$ be an HMLSS-tableau for a given finite collection S of normalized HMLSS-literals. A branch θ of $\mathcal{T}$ is:

- *strict*, if no rule has been applied more than once on θ to the same occurrence of literal;

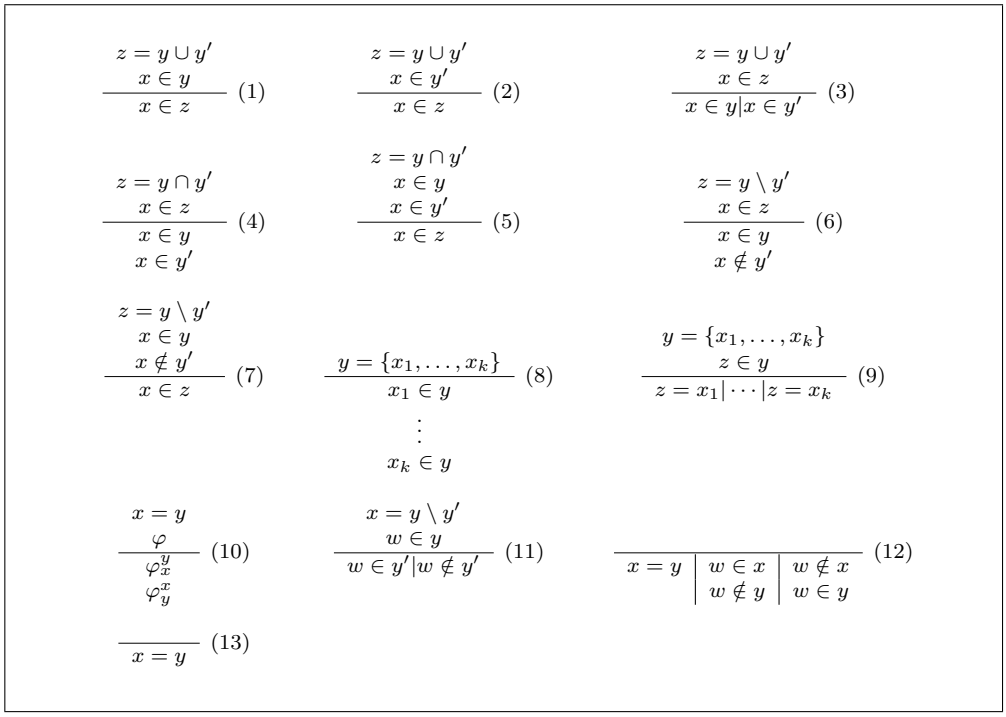

Table 2: The tableau calculus for HMLSS.

- *saturated with respect to a given rule*, if the rule has been applied at least once to each instance of its premises on θ;
- *saturated with respect to a collection of rules*, if θ is saturated with respect to each rule of the collection;
- *satisfiable*, if the collection of normalized literals labelling the nodes of θ is satisfiable in an assignment in the hyperset universe;
- *closed*, if either θ contains a contradictory literal of the form $x \neq x$ or it contains a pair of complementary literals;
- *open*, if it is not closed.

A tableau is:

- *annotated*, if some information is stored on its branches and/or its nodes;
- *strict*, or *saturated with respect to a set of rules*, or *closed* if so are all its branches;

- *satisfiable*, if at least one of its branches is satisfiable.

Notice that closed branches and closed tableaux are unsatisfiable.

It is convenient to recall some further terminology. Let θ be an open branch of a tableau $\mathcal{T}$ for a finite collection S of normalized HMLSS-literals, constructed during the execution of the procedure HMLSS_*tableau_test*(S).

With V_S, T, $\sim_\theta$, T', V', $\hat{\in}_\theta$, G_θ, R_θ we denote the following objects:

- V_S: the set of the variables occurring in S;
- T: the collection of the variables occurring in θ different from V_S;
- $\sim_\theta$: the equivalence relation induced over $V_S \cup T$ by means of the literals $x = y$ in θ;
- T': the set $\{t \in T : t \nsim_\theta x, \text{ for every } x \in V_S\}$;
- V': the set $(V_S \cup T) \setminus T'$;
- $\hat{\in}_\theta$: the binary relation over $V' \cup T'$ defined by $x \hat{\in}_\theta y$ if and only if $x \in y$ is in θ, for $x, y \in V'$;
- G_θ: the direct graph $(V' \cup T', \hat{\in}_\theta)$, called *dependency graph relative to* θ;
- R_θ: the realization of G_θ relative to the partition (V', T') and to pairwise disjoint sets u_t, for $t \in T'$ each of cardinality at least $|V' \cup T'|$.

Some comments on the rules of our tableau calculus and relative saturation strategy are now in order. Rules (1)-(11) in Table 2 coincide with the ones of the tableau calculus for MLSS presented in [3]; in particular, rules (1)-(9) capture the semantics of the operators of the HMLSS language. We recall that they allow to deduce new information about the membership and equality relations on the current branch. Specifically, as will be clarified below, they are applied in the saturation step of the decision procedure. Rule (12) has been introduced for the first time in [3]. It is a cut rule whose application is allowed during the model checking step (illustrated below) to variables of V_S only. Soundness proof of the rules (1)-(12) can be carried out as shown in [3], that is by proving that if a tableau $\mathcal{T}$ is satisfiable, then any tableau obtained from $\mathcal{T}$ by means of a finite number of their applications is still satisfiable.

Rule (13) is the new rule of the calculus. It can applied to a pair of variables $x, y \in V_S$ on a branch θ only in the following two cases:

Case A. The following conditions must hold simultaneously:

- θ is saturated with respect to rules (1)-(11).
- It is possible to construct two pointed flat systems of equations $\mathcal{E}_x = (X, A, e, x)$ and $\mathcal{E}'_y = (X', A', e', y)$ such that:

a. $X, X' \subseteq V_S$,
b. for every $u \in X$ and $v \in X'$,
   1. $\emptyset \neq e_u \subseteq X$ and $\emptyset \neq e'_v \subseteq X'$,
   2. $u = e_u$, $v = e'_v$ are literals occurring in θ,
c. $\mathcal{E}_x$ and $\mathcal{E}'_y$ are bisimilar.

Case B. The following conditions must hold simultaneously:

- θ is an open branch saturated with respect to rules (1)-(11), and saturated with respect to rule (12) up to the $\sim_\theta$ relation (i.e., if rule (12) has been applied to $x, y \in V_S$ and there are $z, w \in V_S$ such that $z \sim_\theta x$ and $w \sim_\theta y$, then rule (12) is not applied to z, w). Moreover, θ is not satisfied by R_θ.
- $x \sim_{R_\theta} y$, that is, $R_\theta x = R_\theta y$.

Let us spend some words on rule (13) and on its application constraints. Rule (13) has been designed to detect and derive (making explicit) existing bisimilarity relations between the variables of the input formula. Lifting to non-well-founded sets, the additional information provided by the application of this rule is used in specific situations (see Sections 5 and 6) to deal with the presence of cycles in the membership relation. During the tableau construction, however, the dependency graph G_θ and the realization R_θ are subject to evolution: G_θ is enriched by the discovery of new dependencies among the variables of the branch and, as a consequence, R_θ is modified. If rule (13) were applied too liberally, the introduction of the literal $x = y$ in θ might be a premature commitment causing an unneeded contradiction.

If rule (13) is applied according to the restrictions of **case A**, we derive new equalities from bisimilarity relations explicitly occurring on the branch. As it is shown in Section 5, these relations are not modified as the dependency graph evolves. On the other hand, if we apply rule (13) in **case B** to two variables x, y on a branch θ, it is shown (see next section) that the equality $R_{\bar\theta} x = R_{\bar\theta} y$ must hold for every prolongation $\bar\theta$ of θ. Then we can apply Theorem 2.6 and derive that the pointed flat systems of equations $\overline{\mathcal{E}}_x$ and $\overline{\mathcal{E}}_y$ introduced in Section 3.2.1 are bisimilar. Thus the literal $x = y$ can be safely added to the branch.

More details on rule (13) and on its proof of correctness with the restrictions introduced above and in the context of the procedure illustrated below are given in Section 5.

Observe that our tableau calculus could be extended with further natural rules as, for instance, a rule of the form

$$\frac{\begin{array}{c} z = y \cup y' \\ x \notin y \\ x \notin y' \end{array}}{x \notin z}$$

procedure *HSaturate*$(\mathcal{T}, V)$
1. **Iterate** till stability
2. strictly saturate the non-closed branches of $\mathcal{T}$ with respect to rules (1)-(11) and then to rule (13) (**case A**).
3. Let $\mathcal{T}$ be the resulting annotated tableau, where it is assumed that during saturation the attributes are inherited by the new branches.
4. **for** each branch θ of $\mathcal{T}$ **do**
5. $\mathit{To\_be\_diversified}_\theta := \{(x_1, x_2) \in V \times V :\ x_i \in z$ and $x_{3-i} \notin z$ in θ, for some $z \in V$ and for some $i \in \{1, 2\}$, or $x_1 \neq x_2$ is in $\theta\}$;
6. $\mathit{Diversified}_\theta := \{(x, y) \in V \times V :\ (x', y') \in \mathit{Diversified}_\theta$, for some $x' \sim_\theta x, y' \sim_\theta y\}$;
7. **end for**;
8. **return** $\mathcal{T}$

end procedure

Figure 2: The saturation procedure.

However, as follows immediately from the total correctness of our decision test (which will be proved in Section 5), the tableau calculus in Table 2 is complete and needs no further rules. We preferred to choose rules which allow to deduce new *memberships*, rather than *nonmemberships*, since the former are used in the construction of candidate models.

The decision procedure we present in the following is similar to the one provided in [3] for the MLSS fragment. It relies upon a saturation strategy for the given tableau calculus and is based on an alternation of calls to an auxiliary procedure, *HSaturate*, with model checking steps attempting the construction of a hyperset model for a branch which is still not closed. Candidate models are constructed by means of the realization of the dependency graph obtained from the information given by the literals present on the branch. If the realization so obtained does not succeed in satisfying the branch, it will not satisfy some of its literals of the "negative" forms $x \notin y$, $x = y \setminus z$, or $x \neq y$.

Such negative information is used to select a pair of variables which, although modeled by the same set, may need to be diversified. If (x_1, x_2) still need to be diversified, rule (12) is applied. Once rule (12) has been applied to every pair of variables that need to be diversified, there may still be some pair of variables, already processed by an application of rule (12), that are still modeled with the same set. At this point, any further application of rules (1)-(12) to the branch would only bring redundant information to the dependency graph, leading neither to satisfiability nor to the closure of the branch. Since the constraints of **case B** are satisfied, we can safely apply rule (13) which, after saturation w.r.t. rule (10), leads to the closure of all branches from θ.

The procedure *HSaturate*$(\mathcal{T}, V)$ saturates in a systematic way and under strictness hypothesis the non-closed branches of the input tableau $\mathcal{T}$ with respect to rules (1)-(11) and rule (13) of Table 2 (**case A**). Each branch θ in the tableau $\mathcal{T}$, constructed by procedures HMLSS_*tableau_test*(S) and *HSaturate*$(\mathcal{T}, V)$, is annotated with the attributes *Diversified*$_\theta$ and *To_be_diversified*$_\theta$. Roughly speaking, the set *Diversified*$_\theta$ consists of all pairs (x_1, x_2) of variables in V_S for which there exists another variable $u \in V_s \cup T$ such that $u \in x_i$ and $u \notin x_{3-i}$ are in θ, for some $i \in \{1, 2\}$. On the other hand, the set *To_be_diversified*$_\theta$ consists of all pairs (x_1, x_2) of variables in V_S such that the branch θ contains either the literal $x_i \neq x_{3-i}$ or the literals $x_i \in y$ and $x_{3-i} \notin y$, for some $y \in V_S$.

In view of what has just been said, it follows that the set Δ_θ consists of all those pairs (x_1, x_2) of variables which are mapped into the same set by the realization R_θ but that need to be diversified in order for the branch to be satisfiable.

5. Correctness of the Procedure HMLSS_*tableau_test*

We show that the procedure HMLSS_*tableau_test* described in Section 4 is totally correct, by first proving its termination and then its partial correctness. This, in particular, entails immediately the completeness of our tableau calculus.

5.1. Termination

Termination of the procedure *HSaturate* is carried out as in [3]. In particular, to prove termination for HMLSS_*tableau_test*, we only need to show that its **while**-loop can only be executed finitely many times. We call $\mathcal{T}^*$ the tableau limit constructed by HMLSS_*tableau_test*. To begin with, we notice that $\mathcal{T}^*$ must be finite. Indeed, if $\mathcal{T}^*$ were infinite, then by König's lemma it would have an infinite branch θ^*. This is possible only if the branch θ^* is processed infinitely many times by instructions 9-21 with the consequence that the set *Diversified*$_{\theta^*}$ would be properly incremented infinitely many times. But this would lead to a contradiction, since *Diversified*$_\theta \subseteq V_S \times V_S$ and moreover for each branch θ the value of the attribute *Diversified*$_\theta$ is monotonic increasing, during execution of HMLSS_*tableau_test*.

Having proved that $\mathcal{T}^*$ is finite, to show termination of the procedure HMLSS_*tableau_test*, it is now enough to observe that at least one node is added to the tableau at each iteration of the **while**-loop of HMLSS_*tableau_test*.

5.2. Partial Correctness

We prove partial correctness of the procedure HMLSS_*tableau_test* under the following assumptions, which will be discharged later:

A1. The assert statement $\Delta_\theta \neq \emptyset$ of line 10 of the procedure HMLSS_*tableau_test* is always true when encountered.

A2. If rule (13) is applied either in **case A** or in **case B** to a pair of variables (x, y) on a branch θ, then $R_{\bar{\theta}}x = R_{\bar{\theta}}y$ each time the realization $R_{\bar{\theta}}$ is computed, for every prolongation $\bar{\theta}$ of θ.

Under assumptions A1 and A2, one can check that the procedure HMLSS_*tableau_test* can only return its control either

- when it finds a realization R_θ satisfying one of the tableau branches (line 7); in this case the realization R_θ must in particular satisfy the input collection S of HMLSS-literals, or
- when all tableau branches are closed (line 33); in this case, because of the soundness of rules (1)-(12) and of rule (13) (guaranteed under assumption A2), it follows immediately that the input set must be unsatisfiable.

Indeed, by Lemma 5.5 below, if there is a non-closed branch θ in $\mathcal{T}$ not satisfied by R_θ (line 9), some of the literals of types $x \notin y$, $x = y \setminus z$, or $x \neq y$ are not satisfied. As illustrated by Lemma 5.8, literals not satisfied by R_θ are used to determine the pairs of variables $(x_1, x_2) \in V_S \times V_S$ that need to be diversified in order to make θ satisfiable (these pairs belong to Δ_θ).

For every pair $(x_1, x_2) \in \Delta_\theta$, at most one attempt of diversification is carried out with rule (12) (lines 11-12). If all the pairs of Δ_θ have already been processed by rule (12), namely all of them belong to $Diversified_\theta$ as well, then θ must be closed by an application of rule (13) **case B** (lines 22-27).

Assumptions A1 and A2 are discharged in Sections 5.2.1 and 5.2.2, respectively. In Section 5.2.3 we provide some further information on the decision procedure HMLSS_*tableau_test* in Fig. 3. In particular we show how to characterize the pairs of variables that will never be diversified on the current branch θ and that, therefore, cause its closure by an application of rule (13) **case B** (Lemma 5.9). We also give a characterization of the pairs of variables that can be diversified by rule (12) (Lemma 5.9). If all the pairs of variables in Δ_θ are of the latter type, the procedure behaves in the same way as the one in [3].

5.2.1. Proof of Assumption A1

We recall some notions and lemmas introduced in [3], which will be useful later.

LEMMA 5.1. *At any step during the construction of a tableau by means of procedures* HMLSS_*tableau_test and HSaturate, the following facts hold:*

(a) if a literal $x \in y$ is added to a branch θ, then $y \sim_\theta y'$, for some y' in V_S.

(b) If $R_\theta x = R_\theta y$, with x and y distinct variables, then there exist x', y' in V_S such that $x \sim_\theta x'$ and $y \sim_\theta y'$. □

LEMMA 5.2. *Let θ be a non-closed branch saturated w.r.t. to rules (1)-(11). Then the realization R_θ defined as before satisfies all literals occurring in θ of the following types:*

$$x = y \cup z, \quad x = y \cap z, \quad x = \{y_1, \ldots, y_k\},$$
$$x = y, \qquad x \in y.$$

Moreover, for each literal $x = y \setminus z$ in θ, we have $R_\theta y \setminus R_\theta z \subseteq R_\theta x$. □

Let θ be a non-closed branch of the tableau $\mathcal{T}$ returned by some call to procedure *HSaturate* during the execution of HMLSS_*tableau_test*(S), and let V_S, T, $\sim_\theta$, T', V', G_θ, and R_θ be defined as before. Let also *To_be_diversified*$_\theta$, *Diversified*$_\theta$ be the attributes of θ computed by procedures HMLSS_*tableau_test* and *HSaturate*. Furthermore, for $(u_1, u_2), (w_1, w_2) \in \Delta_\theta$, let us put $(u_1, u_2) \prec_\theta (w_1, w_2)$ if there exist $i, j \in \{1, 2\}$ such that the literals $u_i \in w_j$ and $u_{3-i} \notin w_j$ are in θ. We have the following properties.

LEMMA 5.3. *If $(w, w') \in \Delta_\theta \cap$ Diversified$_\theta$, then there exists $(u, u') \in \Delta_\theta$ such that $(u, u') \prec_\theta (w, w')$.* □

Proofs of Lemmas 5.1, 5.2, and 5.3 can be found in [3]. The next lemma guarantees the correctness of rule (13) when applied in **case A**.

LEMMA 5.4. *Let θ be a non-closed branch and let x_1, x_2 be two variables of V_S satisfying the constraints of **case A** of rule (13). Then, for every non-closed prolongation $\bar{\theta}$ of θ obtained from θ by saturating it w.r.t. to rules (1)-(11), it holds that $R_{\bar{\theta}} x_1 = R_{\bar{\theta}} x_2$.*

Proof. We recall that x_1 and x_2 satisfy **case A** of rule (13) if it is possible to construct two pointed flat systems of equations $\mathcal{E}_{x_1} = (X, A, e, x_1)$ and $\mathcal{E}'_{x_2} = (X', A', e', x_2)$ such that:

a. $X, X' \subseteq V_S$,

b. for every $u \in X$ and $v \in X'$

   1. $\emptyset \neq e_u \subseteq X$ and $\emptyset \neq e'_v \subseteq X'$,
   2. $u = e_u$, $v = e'_v$ are literals occurring in θ,

c. $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are bisimilar.

Let $\bar{\theta}$ be any prolongation of θ obtained according to the hypothesis. Then, by Lemma 5.2, $R_{\bar{\theta}}$ satisfies all literals of type $x = \{y_1, \ldots, y_k\}$ occurring on $\bar{\theta}$. Among these it satisfies the literals $u = e_u$, $v = e'_v$, for every $u \in X$ and $v \in X'$.

By saturation of $\bar{\theta}$ w.r.t. rules (8) and (9), for every $u \in X$ and $v \in X'$ it holds

$$\begin{aligned} G_{\bar{\theta}}(u) &= \{\bar{x} : \bar{x} \sim_{\bar{\theta}} x, x \in e_u\} \\ G_{\bar{\theta}}(v) &= \{\bar{y} : \bar{y} \sim_{\bar{\theta}} y, y \in e'_v\}. \end{aligned}$$

We recall that for every $x \in V_S$, $G_{\bar{\theta}}(x) = G^e_{\bar{\theta}}(x)$ holds, where G^e is the function of the flat system of equations $\overline{\mathcal{E}} = (V \cup T, \{v_t : t \in T\}, G^e)$ described in Section 3.2.1. Then, under the hypothesis that $\mathcal{E}_{x_1}$ and $\mathcal{E}'_{x_2}$ are bisimilar, the pointed flat systems $\overline{\mathcal{E}}_{x_1}$ and $\overline{\mathcal{E}}_{x_2}$ must be bisimilar too and, by Theorem 2.6, $R_{\bar{\theta}}x_1 = R_{\bar{\theta}}x_2$. □

The following variant of Lemma 5.2 can easily be proved using the same procedure adopted in [3] to prove Lemma 5.2, and applying Lemma 5.4 to treat literals of type $x = y$ derived by the application of rule (13) **case A**.

LEMMA 5.5. *Let θ be a non-closed branch saturated by an application of the procedure HSaturate. Then, the realization R_θ defined as before satisfies all literals occurring in θ of the following types:*

$$\begin{aligned} &x = y \cup z, \quad x = y \cap z, \quad x = \{y_1, \ldots, y_k\}, \\ &x = y, \qquad\quad x \in y. \end{aligned}$$

Moreover, for each literal $x = y \setminus z$ in θ, we have $R_\theta y \setminus R_\theta z \subseteq R_\theta x$. □

The next lemma proves assumption A1.

LEMMA 5.6. *If θ is a branch saturated w.r.t. the procedure HSaturate but not satisfied by R_θ, then $\Delta_\theta \neq \emptyset$.*

Proof. If θ is a branch saturated w.r.t. the procedure *HSaturate* but not satisfied by R_θ, we are in the "**else**" block of the first "**if**" condition of procedure HMLSS_*tableau_test*. Let l be a literal on θ not satisfied by R_θ. Then, by the preceding lemma, the literal l can only be of type $x \notin y$, or $x = y \setminus z$, or $x \neq y$. Thus we have the following cases.

Case: $x \notin y$. If l is the literal $x \notin y$, then our assumption that R_θ does not satisfy l implies that $R_\theta x \in R_\theta y$. Thus either $x \in y$ is in θ, or the literal $x' \in y$ is in θ, for some variable x' distinct from x and such that $R_\theta x' = R_\theta x$. The first case cannot occur, since by hypothesis the branch is not closed. Thus the latter case must hold. By Lemma 5.1 we can assume w.l.o.g. that x, x', y are in V_S. Thus $(x, x') \in \Delta_\theta$ and therefore $\Delta_\theta \neq \emptyset$.

Case: $x = y \setminus z$. R_θ does not satisfy a literal of type $x = y \setminus z$, where we can assume w.l.o.g. that $x, y, z \in V_S$. By the preceding lemma $R_\theta x \subsetneq R_\theta y \setminus R_\theta z$. Hence there must exist an $s \in R_\theta x$ such that $s \notin R_\theta y \setminus R_\theta z$. We have that $s = R_\theta w$, for some variable w for which the literal $w \in x$ is in θ. But then, by saturation w.r.t. rule (6), it would follow that $w \in y$ and $w \notin z$ are in θ, so that $s = R_\theta w \in R_\theta y$. Thus we must also have $s \in R_\theta z$, which yields the existence of a variable w' distinct from w such that the literal $w' \in z$ is in θ and such that $R_\theta w = R_\theta w'$. Thus $(w, w') \in \Delta_\theta$ and therefore $\Delta_\theta \neq \emptyset$.

Case: $x \neq y$. Assume that R_θ does not satisfy a literal of type $x \neq y$. Without loss of generality, we may assume that $x, y \in V_s$. Therefore, since $R_\theta x = R_\theta y$, it follows at once that $(x, y) \in \Delta_\theta$. □

5.2.2. Proof of Assumption A2

Assumption A2 is proved by the following lemma.

LEMMA 5.7. *If rule (13) is applied either in **case A** or in **case B** to a pair of variables (x, y) on a branch θ, then $R_{\bar\theta} x = R_{\bar\theta} y$ each time the realization $R_{\bar\theta}$ is computed, for every prolongation $\bar\theta$ of θ.*

Proof. If rule (13) is applied with the restrictions described in **case A**, then the thesis follows by Lemma 5.5 and from the fact that since θ is an open branch the pair $(x, y) \notin \Delta_\theta$ and (x, y) will never be processed by rule (12). If rule (13) is applied to (x, y) in **case B**, all prolongations of θ will be closed without computing the realization anymore. Thus the thesis follows. □

5.2.3. Some Additional Results on the Behavior of the Procedure

In this section we introduce two lemmas showing in which cases a pair of variables $(x_1, x_2) \in \Delta_\theta$ can be diversified by rule (12) only. Such results are interesting because one may notice that if all the pairs of variables in Δ_θ can be diversified by rule (12), then the procedure described in this paper behaves like the one in [3]. On the other hand, if Δ_θ contains at least one pair of variables (x_1, x_2) that cannot be diversified by rule (12), rule (13) **case B** must be applied in order to close the branch.

In the following lemma we show how effective a diversification step can be.

LEMMA 5.8. *Let (w_1, w_2) be a pair of variables chosen from the set $\Delta_\theta \setminus Diversified_\theta$ when executing line 11 of the procedure* HMLSS_*tableau_test*(S). *Then the following implications hold:*

(a) If a literal of the form $w_i = \{y_1, \ldots, y_k\}$ occurs in θ, for some $i \in \{1, 2\}$, then there are $m \geq k$ prolongations $\theta_1, \theta_2, \ldots, \theta_m$ of θ, obtained by ap-

plying rule (12) and procedure HSaturate to the branch θ, such that $R_{\theta_j} w_1 = R_{\theta_j} w_2$, for $j = 1, 2, \ldots, m$.

(b) If no literal of the form $w_i = \{y_1, \ldots, y_k\}$ occurs in θ, for $i \in \{1, 2\}$, then for every open prolongation $\bar{\theta}$ of θ, obtained by applying rule (12) and procedure HSaturate to the branch θ, it holds that $R_{\bar{\theta}} w_1 \neq R_{\bar{\theta}} w_2$.

Proof. Let $(w_1, w_2) \in \Delta_\theta \setminus \mathit{Diversified}_\theta$ as in the hypothesis.

Concerning (a), let $w_i = \{y_1, \ldots, y_k\}$ occur in θ, for some $i \in \{1, 2\}$. Then the pair (w_1, w_2) still has to be processed by rule (12). It holds that $R_\theta w_1 = R_\theta w_2$, and either the literal $w_1 \neq w_2$ or the literals $w_i \in z$ and $w_{3-i} \in z$ occur in θ, for some $z \in V_S$.

Applying rule (12) as shown by lines 13-21 of the procedure HMLSS_*tableau_test*, three branches are constructed. Among them let us consider the one to which the literals $u \in w_i$ and $u \notin w_{3-i}$ are added, where u is a newly introduced variable.

By the application of rule (9) with premises $w_i = \{y_1, \ldots, y_k\}$ and $u \in w_i$, taking place during the execution of the procedure *HSaturate*, $m \geq k$ prolongations $\theta_1, \theta_2, \ldots, \theta_m$ of θ are created such that $R_{\theta_j} w_1 = R_{\theta_j} w_2$, for $j = 1, 2, \ldots, m$.

Concerning (b), let us assume that we apply rule (12) to the pair (w_1, w_2) and without loss of generality let us consider the prolongation of θ obtained by adding literals $u \in w_i$ and $u \notin w_{3-i}$ (we could have also considered the prolongation of θ obtained by adding literals $u \notin w_i$ and $u \in w_{3-i}$). Since no literal of type $w_i = \{y_1, \ldots y_k\}$ is on the branch and $u \notin V_S$, we can apply neither rule (9) nor rule (13) to put u in relation $\sim_\theta$ with some variable of V_S. Then, according to the definitions given in Section 4, $u \in T'$ and therefore $R_{\bar{\theta}} w_1 \neq R_{\bar{\theta}} w_2$. In fact $R_{\bar{\theta}} w_{3-i}$ cannot contain any a such that $a = R_{\bar{\theta}} u$. Moreover if the literal $u \in w'$ such that $w' \sim_\theta w_{3-i}$ is on the branch we can derive a contradiction. □

In the following lemma, pairs of variables (w_1, w_2) in Δ_θ that cannot be diversified by rule (12) are characterized as the ones for which a pointed flat system $\mathcal{E}_{w_i}$, for some $i \in \{1, 2\}$, is entirely contained in θ.

LEMMA 5.9. *If $\emptyset \neq \Delta_\theta \subseteq \mathit{Diversified}_\theta$, then there exists a $\prec_\theta$-cycle containing (w_1, w_2), for every $(w_1, w_2) \in \Delta_\theta$. Moreover, elements of Δ_θ can be characterized as being the pairs of variables (w_1, w_2) in V_S such that for some $i \in \{1, 2\}$ it is possible to construct a pointed flat system $\mathcal{E}_{w_i} = (X, A, e, w_i)$ satisfying*

1. *$X \subseteq V_S$,*
2. *for every $x \in X$, $x = e_x$ is a literal occurring in θ.*

Proof. Let (w_1, w_2) be an element of Δ_θ. Since $\Delta_\theta \subseteq$ *Diversified*$_\theta$, then $(w_1, w_2) \in \Delta_\theta \cap$ *Diversified*$_\theta$ and, by Lemma 5.3, there exists an infinite descending $\prec_\theta$-chain starting with (w_1, w_2). But $\Delta_\theta \cap$ *Diversified*$_\theta$ is finite, and thus there must exist a $\prec_\theta$-cycle containing (w_1, w_2).

To prove that when $\Delta_\theta \subseteq$ *Diversified*$_\theta$ the elements of Δ_θ are characterized as specified above, take a pair (w_1, w_2) occurring in $\Delta_\theta \subseteq$ *Diversified*$_\theta$. (w_1, w_2) has been processed by rule (12) and, after its application on branch θ, $R_\theta w_1 = R_\theta w_2$. Since $\Delta_\theta \subseteq$ *Diversified*$_\theta$, by Lemmas 5.3 and 5.8, (w_1, w_2) fulfills the conditions of the lemma. On the other hand, let (w_1, w_2) be a pair fulfilling the conditions of the lemma on a branch θ' and still not processed by rule (12). Then it can easily be checked (by iterated applications of Lemmas 5.3 and 5.8) that there is a prolongation θ'' of θ' such that $\emptyset \neq \Delta_{\theta''} \subseteq$ *Diversified*$_{\theta''}$ and $(w_1, w_2) \in \Delta_{\theta''}$. □

Let us now make some considerations on the above two lemmas. Lemma 5.8(a) says that if for one of the two variables w_1 and w_2 chosen for the application of rule (12) a literal of the form $w_i = \{y_1, \ldots, y_k\}$ is in θ, after the application of rule (12) and subsequent saturation of the branches carried out by the procedure *HSaturate* there will be some prolongation $\bar{\theta}$ of θ such that $R_{\bar{\theta}} w_1 = R_{\bar{\theta}} w_2$.

This situation is not problematic, as w_1 and w_2 can still be diversified by the diversification of some of their elements, namely of some pair (u_1, u_2) such that $(u_1, u_2) \prec_{\bar{\theta}} (w_1, w_2)$. Diversification of (u_1, u_2) is possible only if the pair satisfies the condition in Lemma 5.8(b), that is if for every $i \in \{1, 2\}$ no literal of the form $u_i = \{v_1, \ldots, v_m\}$ occurs on the current branch $\bar{\theta}$. Clearly, if $R_{\bar{\theta}} u_1 \neq R_{\bar{\theta}} u_2$, it also holds that $R_{\bar{\theta}} w_1 \neq R_{\bar{\theta}} w_2$. Cases like this are handled without using rule (13) and thus the procedure behaves in the same way as the one described in [3]. If the pair (w_1, w_2) chosen for the application of rule (12) fulfills the conditions specified by Lemma 5.9, then w_1 and w_2 cannot be diversified by any number of applications of rule (12), and thus rule (13) must be applied.

6. Some Examples

In light of what has been proved and discussed in the previous sections, we examplify some applications of the procedure HMLSS_*tableau_test* in which, for conciseness, the tableaux illustrated in the figures below represent only the relevant parts of the proofs produced by the procedure. Let φ_1 be the HMLSS-conjunction $x = \{x\} \wedge y = \{y\} \wedge x \neq y$. A closed tableau for φ_1 is illustrated in Fig. 4. Notice that the literal $x = y$ could be added to the branch because x and y satisfy the restrictions of rule (13), **case A**.

Next, let φ_2 be the HMLSS-conjunction $x \in x \wedge y \in y \wedge x \neq y$. φ_2 differs from φ_1 because while in φ_1 the structure of the hypersets x and y is completely

determined by the literals $x = \{x\}$ and $y = \{y\}$, such constraints are not present in φ_2. The formula φ_2 turns out to be satisfiable, as shown by the tableau described in Fig. 5. The first branch of the tableau is closed while the second and third ones are open and satisfiable. The second branch θ_2 is satisfied by the realization $R_{\theta_2}u = \{v_t\}$, for some suitable set v_t, $R_{\theta_2}x = \{\mathbf{\Omega}, \{v_t\}\}$, $R_{\theta_2}y = \mathbf{\Omega}$. The third branch is satisfied by $R_{\theta_3}u = \{v_t\}$, $R_{\theta_3}x = \mathbf{\Omega}$, $R_{\theta_3}y = \{\mathbf{\Omega}, \{v_t\}\}$. Notice that the tableau has been constructed without applying rule (13). In cases like this the procedure behaves just as the one presented in [3].

Consider again φ_1 and construct a closed tableau for it avoiding to use rule (13) **case A**. The resulting tableau is the one depicted in Fig. 6. As one may notice, the tableau in Fig. 4 is quite smaller.

7. Conclusions

We have presented a decision procedure for the fragment HMLSS of the hyperset theory. The algorithm proposed is similar to a tableau saturation strategy introduced in [3] in the context of the MLSS fragment. Switching from the well-founded to the non-well-founded setting required the construction of some means to deal with membership cycles, namely rule (13), which has been used to derive bisimilarity relations between the variables of the given formula.

To guarantee correct application of the rule on a branch to a pair of variables (x_1, x_2), one must check that the branch considered, which is open and not satisfied by the realization, is saturated with respect to the rules (1)-(11), (13) **case A** and to the rule (12), up to the relation $\sim_\theta$. In fact, in this case the dependency graph will not be enriched by new relevant information after further applications of the rules (1)-(12) and the variables x_1, x_2 of the pair (x_1, x_2) can be therefore guaranteed to remain bisimilar.

Several fragments of well-founded set theory have already been shown to possess a decision procedure [4, 5]. It would be interesting to extend to the hyperset context some of such most basic decidability results.

```
procedure HMLSS_tableau_test(S)
      – V_S the collection of variables occurring in S and
      – 𝒯 the initial tableau induced by S.
 1.   for θ ∈ 𝒯 do
 2.        Diversified_θ := ∅;
 3.   end for;
 4.   𝒯 := HSaturate(𝒯, V_S);
 5.   while there exists a non-closed branch θ in 𝒯 do
 6.        if R_θ satisfies θ then
 7.             return "input formula is satisfied by R_θ"
 8.        else
 9.             put Δ_θ = {(w, w') ∈ V_S × V_S : R_θ w = R_θ w'} ∩ To_be_diversified_θ;
10.             assert: Δ_θ ≠ ∅
11.             if Δ_θ \ Diversified_θ ≠ ∅ then
12.                pick (x, y) ∈ Δ_θ \ Diversified_θ;
13.                apply rule (12) to (x, y):
14.                  let u be the next unused variable;
15.                  split the branch θ into three branches θ1, θ2, and θ3, where
                       θ1 = θ; u ∈ x, u ∉ y;
                       θ2 = θ; u ∉ x, u ∈ y;
                       θ3 = θ; x = y;
16.                  and put
17.                    To_be_diversified_θ1 := To_be_diversified_θ;
18.                    To_be_diversified_θ2 := To_be_diversified_θ;
19.                    To_be_diversified_θ3 := To_be_diversified_θ;
20.                    Diversified_θ1 := Diversified_θ2 :=
                         Diversified_θ ∪ {(x', y')(y', x') ∈ V_S × V_S : x' ∼_θ x, y' ∼_θ y};
21.                    Diversified_θ3 := Diversified_θ;
                   – Notice that after saturation w.r.t. rule (10), all branches
                   – from θ3 will be closed.
22.             else
23.                choose (x1, x2) ∈ Δ_θ ∩ Diversified_θ;
24.                apply rule (13) (case B) to (x1, x2):
25.                  θ1 := θ; x1 = x2;
26.                  To_be_diversified_θ1 := To_be_diversified_θ;
27.                  Diversified_θ1 := Diversified_θ;
                   – Notice that after saturation w.r.t. rule (10), all branches
                   – from θ1 will be closed.
28.             end if;
29.             assign to 𝒯 the resulting tableau;
30.             𝒯 := HSaturate(𝒯, V_S);
31.        end if;
32   end while;
33.  return "input formula is unsatisfiable, as proved by the closed tableau 𝒯"
end procedure
```

Figure 3: The HMLSS-tableau test procedure.

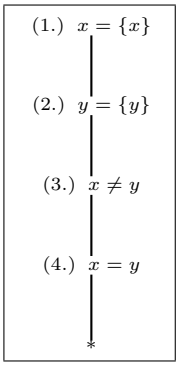

Figure 4: A closed tableau for φ_1.

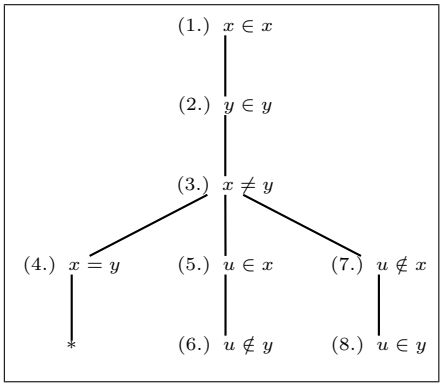

Figure 5: A tableau for φ_2.

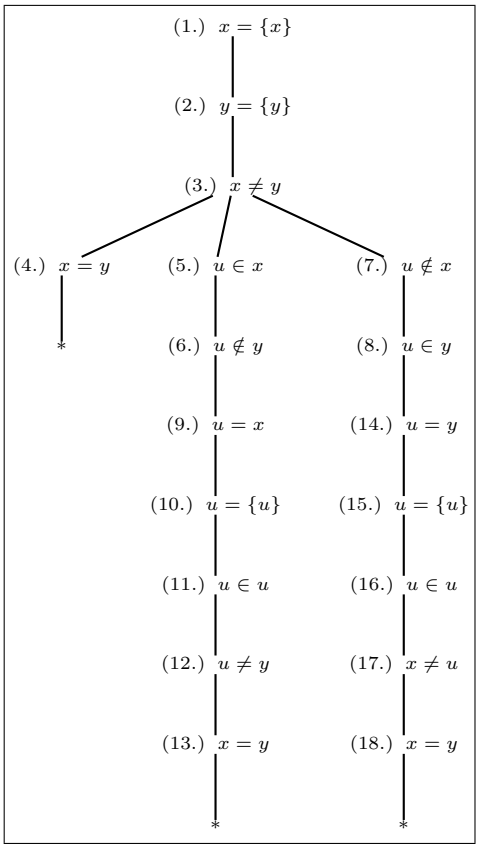

Figure 6: A closed tableau for φ_1 without the application of rule (13) case **B**.

References

[1] P. Aczel, *Non-well-founded sets*, CSLI Lecture Notes volume 14, Stanford University Press, Stanford (1988).

[2] J. Barwise and L.S. Moss, *Vicious circles*, CSLI Lecture Notes volume 60, Stanford University Press, Stanford (1996).

[3] D. Cantone, *A fast saturation strategy for set-theoretic tableaux*, LNCS volume 1227, Springer, Berlin (1997), 122–137.

[4] D. Cantone, A. Ferro and E.G. Omodeo, *Computable set theory*, Clarendon Press, Oxford (1989).

[5] D. Cantone, E.G. Omodeo and A. Policriti, *Set theory for computing — From decision procedures to declarative programming with sets*, Springer,

Berlin (2001).

[6] A. Dovier, E.G. Omodeo and A. Policriti, *Solvable set/hyperset contexts: II. A goal-driven unification algorithm for the blended case*, Appl. Algebra Engrg. Comm. Comput. **9** (1999), 1–48.

[7] E. Ferro, E.G. Omodeo and J.T. Schwartz, *Decision procedures for elementary sublanguages of set theory. I. Multi-level syllogistic and some extensions*, Commun. Pure Appl. Math. **33** (1980), 599–608.

[8] E.G. Omodeo and A. Policriti, *Solvable set/hyperset contexts: I. Some decision procedures for the pure, finite case*, Commun. Pure Appl. Math. **48** (1995), 1123–1155.

[9] C. Piazza and A. Policriti, *Towards tableau-based decision procedures for non-well-founded fragments of set theory*, LNCS volume 1847, Springer, Berlin (2000), 368–382.

[10] C. Piazza and A. Policriti, *Deciding modal logic using tableaux and Set Theory*, in the proceedings of the *AGP*2001*: Appia Gulp Prode*, Évora, Portugal (2001).

Authors' addresses:

Domenico Cantone
Dipartimento di Matematica e Informatica
Università di Catania
viale A. Doria, n. 6, Catania 95125, Italy
E-mail: cantone@dmi.unict.it

Massimo Felici
School of Informatics
The University of Edinburgh
10 Crichton Street, EH8 9AB Edinburgh, U.K.
E-mail: mfelici@staffmail.ed.ac.ek

Marianna Nicolosi Asmundo
Dipartimento di Matematica e Informatica
Università di Catania
viale A. Doria, n. 6, Catania 95125, Italy
E-mail: nicolosi@dmi.unict.it

Received September 15, 2010
Revised October 30, 2010