HANS JÜRGEN OHLBACH & JANA KÖHLER

CHAPTER 3

HOW TO AUGMENT A FORMAL SYSTEM WITH A BOOLEAN ALGEBRA COMPONENT

1. INTRODUCTION

Reasoning with Boolean Algebras is just propositional reasoning. This is well investigated and a lot of good algorithms have been developed. Other formal systems, for example mathematical programming for reasoning about arithmetical equation systems, are equally well developed. Combining such a system with a Boolean component where the Boolean expressions are interpreted as sets, would allow one to use arithmetical algorithms to reason about numerical features of sets.

In this chapter we introduce the atomic decomposition technique as a means to combine computation and reasoning in a given formal system with reasoning about Boolean Algebras and features of the elements of the Boolean Algebra which make sense in the given basic system. Propositional reasoning is invoked in a kind of compilation phase which eliminates the Boolean Algebra part of the problem completely and shifts the main reasoning problem to the basic system. The decomposition method works for combinations of formal systems with a Boolean Algebra component, where the Boolean terms are embedded in *bridging functions* mapping the Boolean parts to objects the given formal system can understand.

Before we start developing the mathematical part, let us introduce the basic idea with a simple example.

1.1. *An Introductory Example*

If Henry has two sons and three daughters, he has five children. In a formal language we can state this as

$$|sons| = 2 \wedge |daughters| = 3 \rightarrow |children| = 5 \tag{3.1}$$

where $|\ldots|$ denotes the set cardinality function. Checking the validity of this formula involves a combination of Boolean reasoning with arithmetic equation solving. Without further information, sons, daughters and children can be arbitrary sets having arbitrary overlaps with each other. Therefore implication (3.1) is not valid in general.

W. Bibel, P. H. Schmitt (eds.), Automated Deduction. A basis for applications. Vol. III

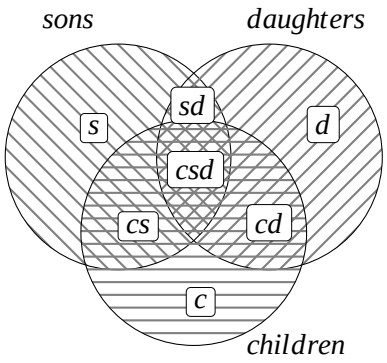


Figure 1. A general set structure

Figure 1 shows the most general way, three different sets can overlap with each other. As one can see, the three sets can be built up from seven mutually disjoint and unseparated areas. We gave these areas names with the following meaning:

c = children, not sons, not daughters.
s = sons, not children, not daughters.
d = daughters, not children, not sons.
cs = children, which are sons, not daughters.
cd = children, which are daughters, not sons.
sd = sons, which are daughters, not children.
csd = children, which are both sons and daughters.

The original sets can now be obtained from their "atomic" components:

$$\begin{aligned} children &= c \cup cs \cup cd \cup csd \\ sons &= s \cup cs \cup sd \cup csd \\ daughters &= d \cup cd \cup sd \cup csd. \end{aligned}$$

Moreover, since this decomposition is mutually disjoint and exhaustive, the cardinalities of the sets just add up:

$$\begin{aligned} |children| &= |c| + |cs| + |cd| + |csd| \\ |sons| &= |s| + |cs| + |sd| + |csd| \\ |daughters| &= |d| + |cd| + |sd| + |csd|. \end{aligned}$$

Formula (3.1) can now be rewritten into

$$(3.2) \qquad \begin{array}{c} |s|+|cs|+|sd|+|csd|=2 \wedge |d|+|cd|+|sd|+|csd|=3 \\ \rightarrow\ |c|+|cs|+|cd|+|csd|=5 \end{array}$$

or by dropping the cardinality function $|...|$

$$(3.3) \qquad \begin{array}{c} s+cs+sd+csd=2 \wedge d+cd+sd+csd=3 \\ \rightarrow\ c+cs+cd+csd=5. \end{array}$$

In (3.3) we interpret the symbols $c, s, \ldots$ directly as the numbers denoting the cardinality of the corresponding sets. This makes sense because the sets are finite and mutually disjoint. This way, problem (3.1) has been transformed into a pure non-negative linear Diophantine equation problem. Diophantine equations are equations with integer-valued variables. They are called *linear* if no products of different variables occur and *non-negative* if variables are constrained to non-negative integers. Formula (3.3), of course, is still not valid. Further information is necessary.

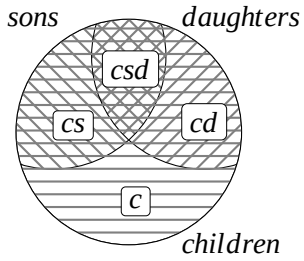


Figure 2. The correct subset relationships.

First of all we add the subset information. Both sons and daughters are children. This means the first picture was a bit too general. We are in a more specific situation which is depicted in figure 2. Here all sets not containing c are empty reflecting the subset information. That means $s=0$, $d=0$, $sd=0$ holds and we can simplify problem (3.3) to

$$(3.4) \qquad cs+csd=2 \wedge cd+csd=3 \quad \rightarrow \quad c+cs+cd+csd=5.$$

which is still not valid. The next piece of information, we need, is that there are no hermaphrodites. This means the intersection of sons and daughters is empty: $csd=0$. We obtain

$$cs=2 \wedge cd=3 \quad \rightarrow \quad c+cs+cd=5.$$

Finally, we exploit the fact that there are no other types of children besides sons and daughters, i.e. $c = 0$. We end up with

$$(3.5) \quad cs = 2 \wedge cd = 3 \quad \rightarrow \quad cs + cd = 5$$

and this is in fact valid.

We get a more abstract view of what we did, if we change the argumentation slightly. The information that *children* are partitioned into *sons* and *daughters* can be represented as propositional formulae:

$$(3.6) \quad \begin{array}{ll} sons \rightarrow children & daughters \rightarrow children \\ children \rightarrow sons \vee daughters & sons \rightarrow \neg daughters. \end{array}$$

There are only three propositional models of these formulae:
$\{children, sons, \neg daughters\}$, $\{children, daughters, \neg sons\}$ and
$\{\neg children, \neg sons, \neg daughters\}$.

The first one corresponds to the set of sons only, the second one to the set of daughters only and the third one to all other objects which are not children. This triggers the decomposition of $|children|$ into $|cs| + |cd|$, $|sons|$ into $|cs|$ and $|daughters|$ into $|cd|$ (the negated propositions in the models can be ignored). Built into the decomposition is the information contained in the propositional axioms (3.6). These axioms are therefore not needed any longer. The decomposed sets are completely independent of each other. Therefore terms like $|cs|$ can be treated like integer variables with the constraint $|cs| \geq 0$. This way even the cardinality function is not needed any longer. The whole problem has been reduced to a pure equational reasoning problem.

1.2. *Boolean Algebras*

The introductory example illustrates the basic idea of the method. Combining propositional reasoning with arithmetic equation solving, however, is only one application of the atomic decomposition. The method can be described in a much more abstract way leading to a wider range of applications. For an appropriate mathematical treatment of the abstract method we need some basic facts and results about Boolean Algebras which may not be familiar to the average reader. Therefore we summarise the material about Boolean Algebras which is relevant for the purposes of this paper. Details can be found in every modern textbook about lattice theory and Boolean Algebra.

A Boolean Algebra $A = (D_A, \bot_A, \top_A, \sqcap_A, \sqcup_A, {}'^A)$ consist of a non-empty *domain* D_A, two distinguished elements $\bot_A$ and $\top_A$, the two binary functions $\sqcap_A$ (meet) and $\sqcup_A$ (join) and the inverse function ${}'^A$. They observe the familiar laws commutativity, associativity, distributivity for $\sqcap_A$ and $\sqcup_A$, the absorption laws for $\sqcap_A$ and $\sqcup_A$ and the inverse law for ${}'^A$.

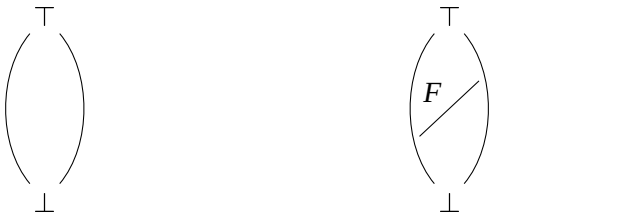


Figure 2.

Examples for Boolean Algebras are the two element Boolean Algebra **2** consisting of just $(\bot, \top)$, the powerset algebra 2^S of some basic set S, or the equivalence classes of propositional formulae (the Lindenbaum algebra for propositional logic).

A *partial order* $x \sqsubseteq_A y$ iff $x \sqcap_A y = x$ can be introduced. It makes $\bot_A$ the smallest element and $\top_A$ the largest element. A *difference function* $x \setminus_A y \stackrel{\text{def}}{=} x \sqcap_A y'_A$ is also very common.

Figure 1.2 shows a very intuitive picture of a Boolean Algebra with smallest element $\bot$ and largest element $\top$.

A Boolean Algebra is *generated* by a set P of objects if all its elements can be obtained by meets, joins and the inverse of elements of P. It is *finitely generated* if P is finite. The Lindenbaum algebra of propositional logic is generated from the set of Boolean variables.

A smallest element above $\bot$ is called an *atom* in a Boolean Algebra. (There also 'anti-atoms'. These are the biggest elements below $\top$.) A Boolean Algebra is *complete* iff it contains all (*infinite*) meets and joins. It is *atomic* iff all its elements x can be obtained as the join of all the atoms below x, i.e.

$$x = \bigsqcup_{a \sqsubseteq_A x,\ a \text{ atomic}} a. \tag{3.7}$$

Finite Boolean Algebras are always complete and atomic. The powerset algebra is also complete and atomic. Its atoms are the singleton sets.

A *filter* F in a Boolean Algebra A is a *proper* subset which is upwards closed and contains all meets. (The dual concept to a filter is an *ideal*). An *ultrafilter* is a maximal filter. An ultrafilters F is characterized by the following properties:

1. for each element x in A, either $x \in F$ or $x' \in F$.
2. $x \sqcap y \in F$ iff $x \in F$ and $y \in F$.
3. $x \sqcup y \in F$ iff $x \in F$ or $y \in F$.
4. $\top \in F$.

Because of condition 1, an ultrafilter F cuts a Boolean Algebra in two halfs, the set of elements which are in F and the set of elements which are not in F. This actually induces a homomorphism h_F from A into the binary algebra **2**.

THEOREM 1.1. (Stone's Representation Theorem (Stone, 1936)).
For every Boolean Algebra A there is an isomorphism ι *between A and the Boolean Algebra of the set of sets of ultrafilters of A:*

$$\iota(x) \stackrel{\text{def}}{=} \{F \mid F \textit{ is an ultrafilter in A and } x \in F\}.$$

Let $\iota(A)$ *be the corresponding set algebra' isomorphic to A.* ■

If A is complete and atomic then ι maps A to the full powerset algebra of its ultrafilters, otherwise ι is only an embedding into it. The ultrafilters in A are actually the atoms in $\iota(A)$. If A is complete and atomic, and F is an ultrafilter then $\iota^{-1}(\{F\})$ is an atom in A.

2. Syntax of the Languages Involved

We need three components in the syntax. The first component of our syntax is the language L_E of some basic system E which we want to augment with a Boolean component. In order to cover as many systems as possible, we do not specify E in detail. The only requirement is that L_E has constant and function symbols, or free variables, which can be existentially quantified. The system E may be very concrete in the sense that everything except the free variables has a fixed meaning. Typical examples for E are systems of arithmetical equations and inequations, but E may be any other suitable formal system.

The second component is the Boolean Algebra component. It is specified in the usual way.

Given a set P of Boolean variables, the Boolean term language is:

$$L_B(P) ::= P \mid L_B(P) \sqcap L_B(P) \mid L_B(P) \sqcup L_B(P) \mid L_B(P)'.$$

Other functions, for example set difference $\setminus$, can be defined as abbreviations in the language.

Important: $L_B(P)$ *and* L_E *need not share any symbols!*

A *Boolean axiom* is of the form $c = \top$ where $c \in L_B(P)$.

$x \sqsubseteq y$ can be used as an abbreviation for $x' \sqcup y = \top$ and $x = y$ can be used for $x' \sqcup y = \top$ together with $y' \sqcup x = \top$.

EXAMPLE 2.1. *If* $P = \{children, sons, daughters\}$ *then* $\{sons \sqcap daughters, children \sqcup sons'\} \subseteq L_B(P)$, *'*$sons \sqsubseteq children$*' and '*$daughters \sqsubseteq children$*' are Boolean axioms.* ■

As a bridge between these two languages L_E and $L_B(P)$ we need a distinguished set F of *bridging functions*, different from all other symbols involved. A typical example for a bridging function is the cardinality function mapping sets to integers. A bridging function symbol may have any finite arity. Each argument position, however, can take either a Boolean term as argument, or an L_E-term. For convenience, we assume that a bridging function of arity $n+k$ reserves the first n arguments for Boolean terms and the remaining k arguments for L_E-terms.

The combined language is defined as follows:

DEFINITION 2.2. (The combined language $L_{BE}(P)$).
If $s[x] \in L_E$ *where* x *is some term occurring at some position in* s, $f \in F$ *with arity* $n+k, t_1, \ldots, t_n \in L_B(P), s_1, \ldots, s_m \in L_E$ *then* $s[x/f(t_1, \ldots, t_n, s_1, \ldots, s_k)] \in L_{BE}(P)$. *No other terms are in* $L_{BE}(P)$. ■

$L_{BE}(P)$ is essentially like L_E, but L_E-term positions can be occupied by $L_B(P)$-terms embedded in a bridging function. The combined language is such that the Boolean parts and the L_E-parts are separated by the functions in F. The Boolean part is the lower part in the term tree, then there comes some bridging function and the upper part belongs to L_E.

EXAMPLE 2.3. E *= arithmetic,* $F = \{|.|, f, a, c\}$ *with the informal meaning:* $|.|$ *is the set cardinality function,* f *means 'combined fortune', a means 'average income' and the binary function c means 'consumption of'. Well formed* $L_{BE}(P)$ *axioms are now*

$|sons \sqcup friends| \geq 5$

(there are more than 5 sons and friends.)

$f(children) \geq 100000$

(the combined fortune of the children exceeds 100000.)

$a(daughters) = 10000$

(the average income of the daughters is 10000.)

$c(children \sqcup friends, cigarettes) = 0$

(The children and friends do not smoke cigarettes.) ■

Bridging functions are instances of Dov Gabbay's fibring functions, he introduced for combining different logical systems (Gabbay, 1996; Gabbay, 1998). In Gabbay's fibring approach, these functions are objects in the semantics of the combined language. They link models of the component languages with each other. In our system these functions have a syntactic representation, which means that one can specify their properties and reason with them explicitly.

3. SEMANTICS OF THE LANGUAGES INVOLVED

The language L_E comes with its natural semantics. The only feature we need is that an interpretation $\Im_E$ for E maps the free variables and constant symbols to elements of E's domain and interprets function symbols as functions in the usual way. In an arithmetical language, $\Im_E$ may for example represent a solution of an equation system.

The language $L_B(P)$ is to be interpreted as a complete and atomic Boolean Algebra, usually, but not necessarily as a set algebra.

The interpretation is therefore a homomorphism $\Im_B : L_B(P) \mapsto A$ where A is a complete and atomic Boolean Algebra.[1]

$\Im_B$ satisfies a Boolean axiom $\varphi = \top$, i.e.
$\Im_B \models (\varphi = \top)$ iff $\Im_B(\varphi) = \top_A$.

Since the language $L_B(P)$ and L_E do not share any symbols, we can define a combined interpretation $\Im_{BE}$ as the union of the interpretations $\Im_B$ and $\Im_E$. The interpretation of the bridging function symbols also becomes part of $\Im_{BE}$.

The interpretation of the bridging function symbols in F can, but need not be fixed. Certain structural conditions, however must be satisfied. First of all, bridging functions must be type conform. That means for a bridging function symbol f with n Boolean arguments and m L_E-arguments, a combined interpretation $\Im_{BE}(f)$ must map tuples consisting of n elements of the Boolean Algebra and m elements of E's domain to an element of E's domain.

But there is the extra requirement of *additivity,* which is essential for making the atomic decomposition work. If a bridging function is applied to the join of two disjoint objects x and y of a Boolean Algebra ($x \sqcap_A y = \bot_A$), then it must be possible to apply the bridging function to x and y separately and then combine the result. For the set cardinality function for example, this is a very natural property: $|x \cup y| = |x| + |y|$, provided x and y are disjoint.

To ensure additivity, we require that the *additivity axioms* are satisfied, either by building them into the system, or by making them an explicit part

[1] In many practical applications, A is even finite. We have not investigated the case where A is not complete or not atomic.

of a specification. The additivity axioms are special interaction axioms for fibring functions in the sense of Gabbay (Gabbay, 1996; Gabbay, 1998).

DEFINITION 3.1. (Additivity Axioms).
The additivity axioms for a bridging function $f \in F$ *with arity* $n+k$ *are:*
$x \sqcap y = \bot \rightarrow f(\ldots, t_{i-1}, x \sqcup y, t_{i+1}, \ldots)$
$* = g_i(f(\ldots, t_{i-1}, x, t_{i+1}, \ldots), f(\ldots, t_{i-1}, y, t_{i+1}, \ldots))$
for each $i \in \{1, \ldots, n\}$ *of the Boolean argument positions, where* $g_i(x,y)$ *is some term in* L_E. ■

EXAMPLES 3.2. *If* $x \cap y = \emptyset$ *then* $|x \cup y| = |x| + |y|$. *Therefore the bridging function 'cardinality' is additive for set algebras and the function* g_1 *in the definition above is the addition of numbers.*

The same holds if the function 'combined fortune' f *in the example above is interpreted in the intended way. The situation for the bridging function 'average income' 'a' mentioned in the same example is a bit more complicated. To get the intended behaviour, we have to require*

$$a(x \sqcup y) = \frac{|x|a(x) + |y|a(y)}{|x| + |y|}.$$

In a different setting we may map the Boolean Algebra terms to, say, sets of axioms of some logic, and the algebra E consists of the set of models of this logic. A bridging function may now be a function M mapping axiom sets to sets of models. This function is additive as well because

$$M(x \sqcup y) = M(x) \cap M(y)$$

holds. Here the function g is set intersection. ■

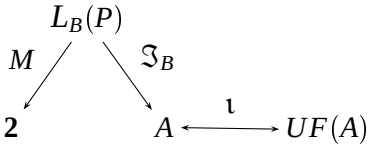
DEFINITION 3.3. (Problem specification).
A problem specification *now consists of three parts*

1. *a set* A *of Boolean axioms,*
2. *a set* E *of* $L_{BE}(P)$ *formulae,*
3. *the bridging function additivity axioms* B *for each bridging function (Def. 3.1).*

The satisfiability *problem is to find out whether such a specification is consistent, i.e. it has an interpretation satisfying all three parts.* ■

4. ATOMIC DECOMPOSITION

The Boolean language $L_B(P)$ is isomorphic to the propositional formula language. The correspondences are $\sqcap \sim \wedge$, $\sqcup \sim \vee$, $' \sim \neg$, $\sqsubseteq \sim \rightarrow$, $= \sim \leftrightarrow$. Therefore we can also have interpretations into binary truth values. As algebraic objects, these binary truth value interpretations are homomorphisms, $M : L_B(P) \mapsto \mathbf{2}$. Therefore we can summarise the situation in the following picture.



PROPOSITION 4.1. (Models and ultrafilters).
Let A be a consistent set of Boolean axioms and $\Im_B : L_B(P) \mapsto A$ be an interpretation satisfying A. There is a (surjective) on-to mapping μ from the set of propositional models of A to the set of ultrafilters of A.

Proof: A induces an equivalence relation on $L_B(P)$ terms such that $\Im_B$ as a model of A maps equivalent formulae to the same element of A. Each element x in A is therefore the image $\Im_B(c_x)$ of some Boolean A-equivalent terms c_x. For each x in A we therefore choose a Boolean term c_x with $\Im_B(c_x) = x$. $\Im_B(c'_x) = x'^A$ is automatically guaranteed.

a) Let m be a propositional model of A. We define

$$\mu(m) \stackrel{\text{def}}{=} \{\Im_B(c) \mid m \models c\}$$

and check the properties 1 – 4 of ultrafilters (page 61).
1.) for $x \in A$, either $m \models c_x$ or $m \models c'_x$. Therefore either $x \in \mu(m)$ or $x'^A \in \mu(m)$.
2.) $x \sqcap_A y \in \mu(m)$ iff $m \models c_x \sqcap c_y$ iff $m \models c_x$ and $m \models c_y$ iff $x \in \mu(m)$ and $y \in \mu(m)$. The proof for the third condition is analogous and the fourth condition is trivial.

b) Let F be an ultrafilter in A. We define $m = \mu^{-1}(F)$ to be a propositional interpretation such that

$$m \models c \text{ iff } \Im_B(c) \in F.$$

Using the conditions 1 – 4 for ultrafilters again, it is straightforward to prove that m is an interpretation. m satisfies A because $\Im_B$ satisfies A and therefore it maps all terms ψ with '$\psi = \top$' $\in A$ to $\top_A$. Thus, $\Im_B(\psi) \in F$ and therefore $m \models \psi$.

The definitions in a) and b) guarantee $\mu(\mu^{-1}(F)) = F$. The mapping is not injective. For example in the extreme case where A is the binary Boolean Algebra **2**, there is only one ultrafilter consisting of $\{\top\}$ only. All propositional models are mapped to this single ultrafilter. ■

As a consequence of Stone's representation theorem we also get an on-to mapping from the propositional models of A to the atoms in A, provided A is complete and atomic. In this case the isomorphism ι yields a one-to-one mapping between A's atoms and A's ultrafilters, which are the atoms of $\iota(A)$. Therefore $\nu(m) \stackrel{\text{def}}{=} \iota(\mu(m))$ yields for a model m a corresponding atom in A.

COROLLARY 4.2. (Propositional Models and Atoms).
If A is complete and atomic there is an on-to mapping ν from the set of propositional models of A to the set of atoms in A.

Furthermore we have for each $x = \Im_B(c)$ in A, atom a in A and for each propositional model m with $\nu(m) = a$:
$a \sqsubseteq_A x$ iff $m \models c$. ■

EXAMPLE 4.3. *Let us illustrate this with a fragment of the introductory example. Suppose the Boolean axiomatization is*

$$A = \{sons \sqsubseteq children, daughters \sqsubseteq children\}.$$

This corresponds to the propositional clause set

$$\{\{\neg sons, children\}, \{\neg daughters, children\}\},$$

which has five models:

1. $\{\neg sons, \neg daughters, \neg children\}$
2. $\{\neg sons, \neg daughters, children\}$
3. $\{\neg sons, daughters, children\}$
4. $\{sons, \neg daughters, children\}$
5. $\{sons, daughters, children\}$.

Each of these models corresponds to an area in Figure 2. The first one corresponds to the area surrounding the circle, no children at all. The second one corresponds to c, the third one to cd, the fourth one to cs and the fifth one to csd. These are the atoms of the Boolean Algebra consisting of the set of sets which can be obtained by forming their set-union. ■

We can now introduce a syntactic representation for the propositional models of A and extend the interpretation $\Im_B$ to map the syntactic representation of the models, let us call them *syntactic atoms*, to the atoms of A:

$$\text{If } m \models A \text{ then } \Im_B(m) \stackrel{\text{def}}{=} \nu(m) \in A.$$

The precise structure of the syntactic representation of A's models can be left to an implementation. Quite convenient is the representation we used in the introductory example, as a set or string of Boolean variables.

Corollary 4.2 and (3.7) now yields the basis for our atomic decomposition calculus

$$\mathfrak{I}_B(c) = \bigsqcup_{m \models c, m \models A} \mathfrak{I}_B(m)$$

for each Boolean term c.

This allows us to map Boolean terms to sets of syntactic atoms. A set of such atoms denotes the join of the interpretations of its members. Therefore we define for sets $\{m_1, \ldots, m_n\}$ of atoms

$$\mathfrak{I}_B(\{m_1, \ldots, m_n\}) \stackrel{\text{def}}{=} \mathfrak{I}_B(m_1) \sqcup_A \ldots \sqcup_A \mathfrak{I}_B(m_n).$$

DEFINITION 4.4. (Atomic Decomposition). *We define a decomposition function α_A which maps Boolean terms to sets of syntactic atoms (propositional models of A).*

$$\begin{array}{lcll} \alpha_A(p) & = & \{m_1, \ldots, m_n\} & \textit{where } p \textit{ is a Boolean variable} \\ \alpha_A(x \sqcup y) & = & \alpha_A(x) \cap \alpha_A(y) & \textit{and } m_i \models p \\ \alpha_A(x \sqcap y) & = & \alpha_A(x) \cup \alpha_A(y) & \\ \alpha_A(x') & = & M \setminus \alpha_A(x) & M \textit{ is the set of all models of } A. \end{array}$$

■

Notice that α_A yields sets of syntactic atoms by performing set operations on sets of syntactic atoms. This is no longer an operation within one of the languages, but with datastructures of the corresponding implementation.

Using that $\mathfrak{I}_B(m_1) \sqcap_A \mathfrak{I}_B(m_2) = \bot_A$ for two different atoms m_1 and m_2 it is now straightforward to show with structural induction

(3.8) $\quad \mathfrak{I}_B(c) = \mathfrak{I}_B(\alpha_A(c))$

for all Boolean terms c. For each '$\psi = \top$' $\in A$ this means in particular $\top_A = \mathfrak{I}_B(\psi) = \mathfrak{I}_B(\alpha_A(\psi))$ and therefore

(3.9) $\quad \alpha_A(\psi) =$ set of all atoms in A.

The next proposition confirms that the Boolean axioms A are no longer needed after the decomposition.

LEMMA 4.5. (Elimination of the Boolean axioms).
A specification $\{A, B, E\}$ consisting of the Boolean axioms A, the additivity axioms B and some $L_{BE}(P)$-formulae E is satisfiable if and only if $\{B, \alpha_A(E)\}$ is satisfiable.

Proof: If $\{A, B, E\}$ is satisfied by some interpretation $\Im$, then because of (3.8), the Boolean terms c and $\alpha_A(c)$ are mapped to the same objects. Therefore $\alpha_A(E)$ is satisfied as well.

Now suppose there is some model $\Im$ for $\{B, \alpha_A(E)\}$. $\Im$ maps the syntactic atoms m_i occurring in $\alpha_A(E)$ to some objects in some domain. Let $M \stackrel{\text{def}}{=} \{k_1, \ldots, k_l\} \stackrel{\text{def}}{=} \{\Im(m_1), \ldots, \Im(m_n)\}$ where $m_1, \ldots, m_n$ are all syntactic atoms. We define the Boolean Algebra A to be the powerset algebra of M. A is complete and atomic. The atoms in A are just the objects $\Im(m)$ where the m are the syntactic atoms.

$\Im$ is extended to $\Im_B$ such that $\Im_B$ maps Boolean variables p to $\{\Im(m) \mid m \in \alpha_A(p)\}$. Because of (3.9), it maps each ψ with '$\psi = \top$' $\in A$ to the interpretation of the set of all atoms, which is just the top element in A. Thus, $\Im_B$ satisfies A.

Structural induction using definition 4.4 yields $\Im_B(c) = \Im_B(\alpha_A(c))$. Since $\Im$ satisfies $\alpha_A(E)$, $\Im_B$ now must satisfy E. Thus, the original specification is satisfiable. ■

Having eliminated the Boolean part, we use now the additivity axioms B to eliminate the bridging functions together with the atoms introduced by α_A. This is done in two steps. In the first step we eliminate the additivity axioms.

LEMMA 4.6. (Elimination of Additivity Axioms).
Let $\{B, E'\}$ be a decomposed specification and let $E'' \stackrel{\text{def}}{=} \rho(E')$ be the result of an exhaustive left to right application of the equations in B to E, taking the sets $\{m_1, \ldots, m_k\}$ as joins $m_1 \sqcup \ldots \sqcup m_k$. (The bridging functions in E' then occur only with a single atom m as argument, no longer with a set of atoms).
$\{B, E'\}$ is satisfiable iff E'' is satisfiable.

Proof: The additivity axioms in B are conditioned equations. The conditions $x \sqcap y$ are satisfied for atoms x and y. Thus, the equation itself can be used as rewrite rules. Using them as rewrite rules is an equivalence transformation. Thus, E' and E'' are equivalent, given B. What remains to be shown is that whenever there is a model for E'', there is also one satisfying B. Let $\Im$ satisfy E''. As in the proof of proposition 4.5 we construct the algebra A as the powerset algebra of the interpretations of the m_i. Starting with the interpretations of the m_i as atoms of the powerset algebra, each of its elements can now be constructed as the join of some smaller elements. Therefore we can take the additivity axioms for the bridging functions f themselves as a recursive definition for the values of $f(x_1, \ldots, x_n, s_1, \ldots, s_k)$. The basis of the recursion is the interpretation of the terms $f(m_1, \ldots, m_n, s_1, \ldots, s_k)$ for the atoms m_i, which is provided by $\Im$. This way we get an interpretation satisfying B and E. ■

The last step is now to eliminate the bridging terms $f(m_1,\ldots,m_n,s_1,\ldots,s_k)$ in the transformed specification E''. If $k=0$, each such term can just be replaced by a new (existentially quantified) L_E-variable or constant symbol. If $k>0$, $f(m_1,\ldots,m_n,s_1,\ldots,s_k)$ is translated into a term $f'_{m_1,\ldots,m_n}(s_1,\ldots,s_k)$ where $f'_{m_1,\ldots,m_n}$ is a fresh new L_E-function symbol.

LEMMA 4.7. (Elimination of the Bridging Functions).
We define a replacement operation β *which replaces all bridging function symbols* f *with n Boolean and k* L_E*-arguments by corresponding* L_E*-terms.* β *introduces for each term* $f(m_1,\ldots,m_n,\ldots)$ *a new* L_E*-function (or constant) symbol* $f'_{m_1,\ldots,m_n}$ *and replaces terms* $f(m_1,\ldots,m_n,s_1,\ldots,s_k)$ *with* $f'_{m_1,\ldots,m_n}(s_1,\ldots,s_k)$.
E'' *is satisfiable if and only if* $\beta(E'')$ *is satisfiable.*

Proof: If E'' is satisfied by some interpretation $\Im$ we just choose $\Im(f'_{m_1,\ldots,m_n})(x_1,\ldots,x_k) \stackrel{\text{def}}{=} \Im(f(m_1,\ldots,m_n,x_1,\ldots,x_k))$.
This guarantees that $\beta(E'')$ is satisfied as well.
For the other direction, suppose $\Im$ satisfies $\beta(E'')$. For the interpretation of the terms $f(m_1,\ldots,m_n,s_1,\ldots,s_k)$ we just map the atoms m_i to themselves and define the interpretation of the bridging functions f to be

$$\Im(f)(m_1,\ldots,m_k,x_1,\ldots,x_k) \stackrel{\text{def}}{=} \Im(f'_{m_1,\ldots,m_n})(x_1,\ldots,x_k).$$

This is possible because neither the atoms m_i nor the bridging functions f occur in $\beta(E'')$. The terms $f(m_1,\ldots,m_n,s_1,\ldots,s_k)$ in E'' are now interpreted exactly like the terms $f'_{m_1,\ldots,m_n}(s_1,\ldots,s_k)$ in $\beta(E'')$. Nothing else is changed and therefore E'' is satisfied as well. ∎

As a consequence of the lemmas 4.5, 4.6 and 4.7 we get a general soundness and completeness result.

THEOREM 4.8. (Soundness and Completeness).
A problem specification (A,E,B) *is satisfiable iff and only if* A *is satisfiable and the transformed specification* $\beta(\rho(\alpha_A(E)))$ *is satisfiable.* ∎

The inference procedure derived from this theorem comprises the following steps: in order to check satisfiability of a combined specification (A,E,B), first compute the syntactic atoms derived from the propositional models of A. If there are no models then the specification is unsatisfiable. If there are models, decompose the Boolean terms occurring in E into sets of syntactic atoms. Use the additivity axioms in E to push the bridging functions down to the level of single atoms. Then replace the resulting 'bridging terms' with

variables or composed L_E-terms, and check the result with an E-satisfiability checker.

If satisfiability in E is decidable we get a decision procedure for the combination with the Boolean language. Satisfiability for this combination is then decidable as well. However, if the decision problem in E is exponential, the complexity of the combined procedure does may get double exponential because exponentially many new variables may be introduced. In section 5 below this problem is discussed in more detail.

EXAMPLE 4.9. *Using our method, it is quite straightforward to enrich classical propositional logic with a component for reasoning about sets. The bridging functions are just predicates on sets.*

As an example, consider again the specification (3.6) of children partitioned into sons and daughters. As bridging function we take the predicate $beautiful$. $beautiful(x)$ *is intended to mean 'every element of x is beautiful'. The additivity axiom is now*

$$beautiful(x \sqcup y) = beautiful(x) \wedge beautiful(y).$$

This holds even without the condition of x and y being disjoint.

The formulae in the combined language are for example

1) $beautiful(children)$
2) $beautiful(daughters) \rightarrow happy\text{-}father.$

The decomposition turns 1) into $beautiful(sons) \wedge beautiful(daughters)$ *and leaves 2) essentially the same. Replacing the bridging function terms with new predicate symbols yields the pure propositional formulae*

$$beautiful\text{-}sons$$
$$beautiful\text{-}daughters$$
$$beautiful\text{-}daughters \rightarrow happy\text{-}father$$

from which $happy\text{-}father$ *can be derived with Modus Ponens.*

EXAMPLE 4.10. *If the basic logic E is first-order predicate logic, we can enrich this logic with Boolean terms as arguments of the predicates, provided the additivity axioms hold. Typically they would again look like*

$$p(x \sqcup y, \ldots) = p(x, \ldots) \wedge p(y, \ldots).$$

The first steps of the procedure transforms each literal p with a Boolean term as argument into $p(\{m_1, \ldots, m_k\}, \ldots)$. *Instead of applying the additivity axioms now, which might yield a big conjunction for the positive literals and a*

big disjunction for the negative literals, one can build them into a new theory resolution rule:

$$\frac{p(\{m_1,\ldots,m_k\},s),C \quad \neg p(\{u_1,\ldots,u_l\},s'),D \qquad s\sigma = s'\sigma}{(p(\{m_1,\ldots,m_k\}\setminus\{u_1,\ldots,u_l\},s),C)\sigma.}$$

5. OPTIMISATIONS

A formula with l Boolean variables may in the worst case have 2^l models. For all of them one has to generate syntactic atoms. This makes the whole approach questionable. Fortunately there are some optimisations which can reduce the number of syntactic atoms considerably.

5.1. *Relevancy Principle*

A Boolean variable p occurring in the Boolean axioms A of some problem specification (A,B,E), but not in the E-formulae E does not contribute much to the problem solution. Boolean variables are implicitly existentially quantified. That means A is in fact short for $\exists p\, A$ if p does not occur in E. A is a propositional formula, and therefore the existentially quantified p can be eliminated using a quantifier elimination procedure (Gabbay and Ohlbach, 1992). The result is some formula A' which is equivalent to $\exists p\, A$, but does not contain p. In the propositional case, elimination of the existentially quantified p amounts to generating all resolvents with p in the clause form of A. The resolvents represent all consequences of p and therefore p is no longer necessary (Ackermann, 1935).

This way one can have large databases of Boolean axioms, but for the actual problem at hand, the atomic decomposition takes into account only the relevant Boolean variables.

5.2. *Factoring Principle*

A Boolean axiomatization A which can be split into separate parts $A_1,\ldots,A_r$ such that the parts mutually do not share Boolean variables, simplifies the atomic decomposition as well. The set of propositional models for A can be factored into the product $M_1\times,\ldots,\times M_r$ of the set of models for the A_i. The mapping μ from the set of propositional models to the set of ultrafilters of the algebra A, we established in proposition 4.1, implies that the ultrafilters, and thus the whole algebra A can be factored into the product

$A_1 \times \ldots \times A_r$ of its components. The atoms of such a product have the form $(\ldots, \bot_{i-1}, a_i, \bot_{i+1}, \ldots)$ where a_i is an atom of A_i and all other components are the bottom elements of the other algebras.

This can be exploited for the representation of the syntactic atoms. They can have the form $(\ldots, \bot, m_i, \bot, \ldots)$ where m_i is a syntactic atom of the component M_i. A further simplification is possible by just storing m_i and labelling it with the information 'belongs to M_i'.

This reduces the overall number of syntactic atoms from $|M_1| \cdot \ldots \cdot |M_r|$ to $|M_1| + \ldots + |M_r|$, which is an exponential improvement.

The meaning of this simplification makes sense also from an application point of view. As an illustration, consider some A axiomatizing, say family relationships as in the introductory example, and in addition relationships between the makes of cars. If there are no axioms saying something about the union or intersection of people and cars, then the factoring operation implicitly imposes that there is no object which is at the same time a person and a car. Therefore the whole Boolean Algebra is split into the part with sets of people and the part with sets of cars. People and cars together are represented by tuples in the product algebra. On the calculus side we therefore get syntactic atoms which represent either people or cars, but none for the intersection of both.

6. Extensions to Other Algebras

Many other algebras are extensions of Boolean Algebras. Relational Algebras provide composition and inverse of binary relations. Boolean Algebras with operators are the algebraic counterparts of modal logics (Jónsson and Tarski, 1951). Both have useful applications in computer science and the knowledge representation area. In particular Boolean Algebras with operators bring extra dimensions into a logical system, for example time dependency.

The elements of these algebras are still elements of the underlying Boolean Algebra, and all terms in the syntactic representation denote elements of this Boolean Algebra. Therefore in principle the mechanism of atomic decomposition still should work. Unfortunately there are in general infinitely many terms, which means that the syntactic representation of the atoms may become infinite. In many cases, however, the relevancy principle can help. We must make sure that only the terms occurring in the problem at hand, or at least a finite superset of them is needed, and all others can be ignored. In this case the representation of the atoms becomes finite again. For the finitely many relevant terms one can first compute the atoms as ordinary propositional models where all the terms in the extension of the Boolean Algebra are

treated as Boolean variables. These atoms are conjunctions of terms in the language of the extension of the Boolean Algebra. For each of these conjunctions one has to check whether they are consistent in the extended algebra, using a decision procedure for this algebra. The inconsistent conjunctions are deleted from the atomic decomposition. If satisfiability in the extended algebra is decidable then there are only finitely many tests to be done, and the decomposition method works.

The problem to ensure that only a finite superset of the terms occurring in the problem specification is needed, is in many algebras related to a finite model property. For example in the algebra corresponding to the modal logic K, the finite model property yields that there is always a finite tree model as deep as the maximal number of nestings of modal operators occurring in the problem (Chellas, 1980). For all modal terms with deeper nestings, the $\Box$ versions become $\top$ and the $\Diamond$ versions become $\bot$.

The finiteness condition has to be analyzed for each class of algebras, but for may cases well known results can be exploited for this purpose.

7. SUMMARY

Sophisticated inference procedures as for example mathematical programming techniques are usually restricted to the language they have been developed for. Any extension of the language is extremely difficult to incorporate into the procedure.

In this contribution we have developed a technique for incorporating a Boolean component into such a language without affecting the calculus. With the atomic decomposition of the Boolean terms we were able to eliminate the Boolean component completely from a mixed problem specification and reduce the given problem to the language of the basic system.

In the Boolean component one can axiomatize sets and set-theoretic relationships between them. The interface between the two languages is provided by a set of 'bridging functions'. They map the sets to objects of the basic system. Different bridging functions can be used simultaneously to deal with different features of the sets. A typical such bridging function is the cardinality function mapping sets to non-negative integers. Any inference system that can deal with arithmetic can therefore be used to reason about cardinalities and other numeric attributes of sets.

An interesting feature of our approach is the fact that more information at the Boolean side, i.e. more Boolean axioms, in general simplifies matters. More Boolean axioms usually means less propositional models, which in turn

means less syntactic atoms. Less syntactic atoms eventually means less variables to deal with in the arithmetic part.

We have sketched an extension of the method to more complex algebras such as Relational Algebras or Boolean Algebras with operators. But the details for these extensions still have to be worked out.

REFERENCES

Ackermann, W.: 1935, 'Untersuchung über das Eliminationsproblem der mathematischen Logik'. *Mathematische Annalen* **110**, 390–413.

Chellas, B. F.: 1980, *Modal Logic: An Introduction.* Cambridge: Cambridge University Press.

Gabbay, D. M.: 1996, 'An Overview of Fibred Semantics and The Combination of Logics'. In: F. B. K. Schulz (ed.): *Proceedings of FroCoS'96, Frontiers of Combining Systems*. pp. 1–56, Kluwer.

Gabbay, D. M.: 1998, *Fibring Logics*. Oxford University Press.

Gabbay, D. M. and H. J. Ohlbach: 1992, 'Quantifier elimination in second–order predicate logic'. In: B. Nebel, C. Rich, and W. Swartout (eds.): *Principles of Knowledge Representation and Reasoning (KR92)*. pp. 425–435, Morgan Kaufmann.

Jónsson, B. and A. Tarski: 1951, 'Boolean Algebras with Operators, Part I'. *American Journal of Mathematics* **73**, 891–939.

Stone, M. H.: 1936, 'The Theory of Representations for Boolean Algebras'. *Transactions of American Mathematical Society* **40**, 37–111.