Capitolo 1

Implementazione di procedure di decisione per frammenti Binding in Vampire

L'algoritmo di decisione, la classificazione, Il preprocessing

1.1 Preprocessing

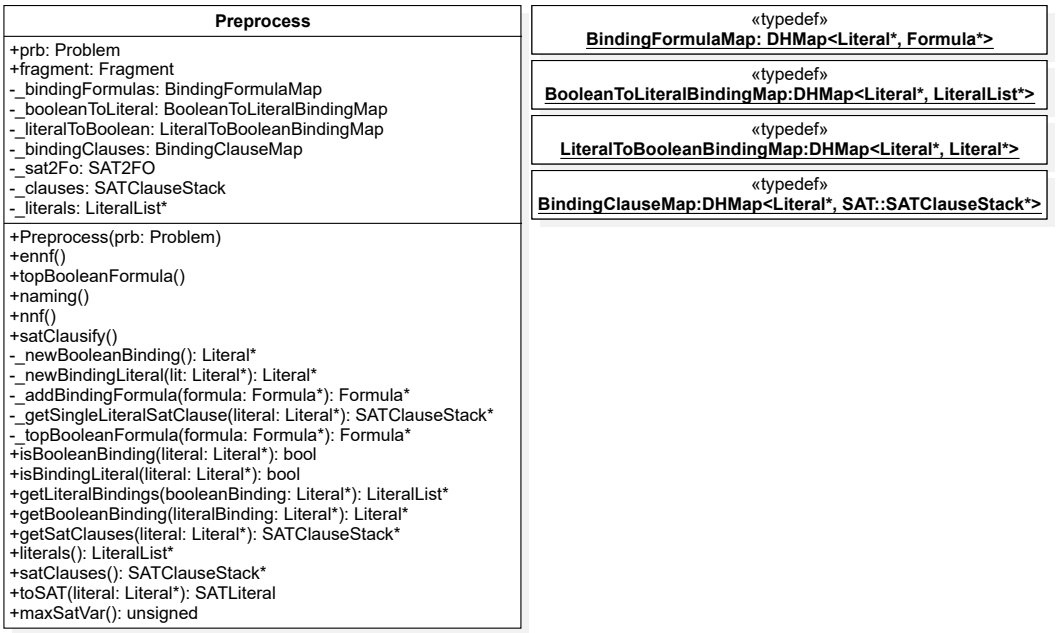

Figura 1.1: Struttura del Preprocessing

1.1.1 Boolean Top Formula

1.1.2 Forall-And

1.1.3 SAT-Clausification

1.2 Procedura di Decisione

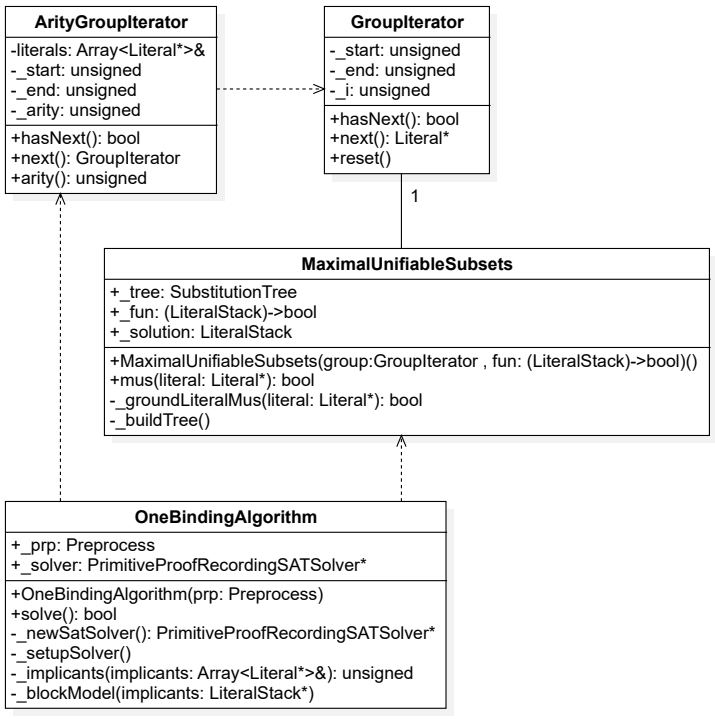

Figura 1.2: Struttura dell'algoritmo di decisione

1.2.1 Implicants Sorting

1.2.2 Maximal Unifiable Subsets

Algorithm 1: Maximal Unifiable Subsets

Firma: mus($literal$)
Input: $literal$ un puntatore ad un letterale
Output: $\top$ o $\bot$
GlobalData: S una mappa da letterali a bool
if $S[literal]$ **then**
  return $\top$;
end
if $literal$ *is ground* **then**
  return $groundLiteralMus(literal)$;
end
$S[literal] = \top$;
$l := \emptyset$;
$res := mus(literal, l)$;
$S[literal] = \bot$;
return res;

1.2.3 Algoritmo Finale

1.3 Algoritmo di Classificazione

(Input formula rettificata senza true e false)

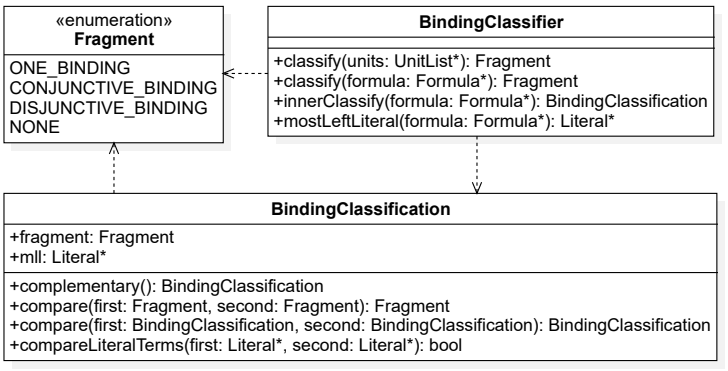

Figura 1.3: Classificatore

Algorithm 2: Classificatore esterno

Firma: classify(φ) **Input:** φ Una formula rettificata
Output: Un elemento dell'enumerazione Fragment
switch φ **do**
    case *Literal* **do**
        return ONE_BINDING;
    end
    case $A[\wedge, \vee]B$ **do**
        return $compare(classify(A), classify(B))$;
    end
    case $\neg A$ **do**
        return $classify(A).complementary()$;
    end
    case $[\forall, \exists]A$ **do**
        $sub := \varphi$;
        connective := connective of φ;
        repeat
            sub := subformula of sub;
            connective := connective of sub;
        until *connective* $\notin \{\forall, \exists\}$;
        $(fragment, \_) := innerClassify(sub)$;
        return $fragment$;
    end
    case $A \Leftrightarrow B$ **do**
        return $compare(classify(A \Rightarrow B), classify(B \Rightarrow A))$;
    end
    case $A \oplus B$ **do**
        return $classify(A \Leftrightarrow B).complementary()$;
    end
    case $A \Rightarrow B$ **do**
        return $compare(classify(\neg A), classify(B))$;
    end
end

Algorithm 3: Classificatore interno

Firma: innerClassify(φ) **Input:** φ Una formula rettificata
Output: Una coppia (Fragment, Literal)
switch φ **do**
  case *Literal* l **do**
    return $(ONE\_BINDING, l)$;
  end
  case $A[\wedge, \vee]B$ **do**
    return $innerCompare(innerClassify(A), innerClassify(B), \text{connective of } \varphi)$;
  end
  case $\neg A$ **do**
    return $innerClassify(A).complementary()$;
  end
  case $A[\Rightarrow, \Leftrightarrow, \oplus]B$ **do**
    return $innerCompare(innerClassify(A), innerClassify(B), \text{connective of } \varphi)$;
  end
  else
    return $(None, null)$;
  end
end

Algorithm 4: Compare esterno

Firma: compare(A, B) **Input:** A, B due elementi dell'enumerazione Fragment
Output: Un elemento dell'enumerazione Fragment
if $A = B$ **then**
  return A;
end
if $One\_Binding \notin \{A, B\}$ **then**
  return $None$;
end
return $max(A, B)$;

Algorithm 5: Compare interno

Firma: innerCompare(A, B, con) **Input:** A, B due coppie (Fragment, Literal), *con* un connettivo
Output: Una coppia (Fragment, Literal)

```
switch A.first, B.first, con do
    case One_Binding, One_Binding, _ do
        if A.second has same terms of B.second then
            return A;
        end
        else if conn = ∧ then
            return (Conjunctive_Binding, null);
        end
        else if conn = ∨ then
            return (Disjunctive_Binding, null);
        end
    end
    case [One_Binding, Conjunctive_Binding | Conjunctive_Binding, One_Binding], ∧ do
        return (Conjunctive_Binding, null);
    end
    case [One_Binding, Disjunctive_Binding | Disjunctive_Binding, One_Binding], ∨ do
        return (Disjunctive_Binding, null);
    end
    case Conjunctive_Binding, Conjunctive_Binding, ∧ do
        return (Conjunctive_Binding, null);
    end
    case Disjunctive_Binding, Disjunctive_Binding, ∨ do
        return (Disjunctive_Binding, null);
    end
end
return (None, null);
```