

Capitolo 1

Implementazione di procedure di decisione per frammenti Binding in Vampire

L'algoritmo di decisione, la classificazione, Il preprocessing

1.1 Preprocessing

Preprocess	«typedef» BindingFormulaMap: DHMap<Literal*, Formula*>
+prb: Problem +fragment: Fragment - _bindingFormulas: BindingFormulaMap - _booleanToLiteral: BooleanToLiteralBindingMap - _literalToBoolean: LiteralToBooleanBindingMap - _bindingClauses: BindingClauseMap - _sat2Fo: SAT2FO - _clauses: SATClauseStack - _literals: LiteralList*	«typedef» BooleanToLiteralBindingMap: DHMap<Literal*, LiteralList*>
	«typedef» LiteralToBooleanBindingMap: DHMap<Literal*, Literal*>
+Preprocess(prb: Problem) +ennf() +topBooleanFormula() +naming() +nnf() +satClausify() - _newBooleanBinding(): Literal* - _newBindingLiteral(lit: Literal*): Literal* - _addBindingFormula(formula: Formula*): Formula* - _getSingleLiteralSatClause(literal: Literal*): SATClauseStack* - _topBooleanFormula(formula: Formula*): Formula* +isBooleanBinding(literal: Literal*): bool +isBindingLiteral(literal: Literal*): bool +getLiteralBindings(booleanBinding: Literal*): LiteralList* +getBooleanBinding(literalBinding: Literal*): Literal* +getSatClauses(literal: Literal*): SATClauseStack* +literals(): LiteralList* +satClauses(): SATClauseStack* +toSAT(literal: Literal*): SATLiteral +maxSatVar(): unsigned	«typedef» BindingClauseMap: DHMap<Literal*, SAT::SATClauseStack*>

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

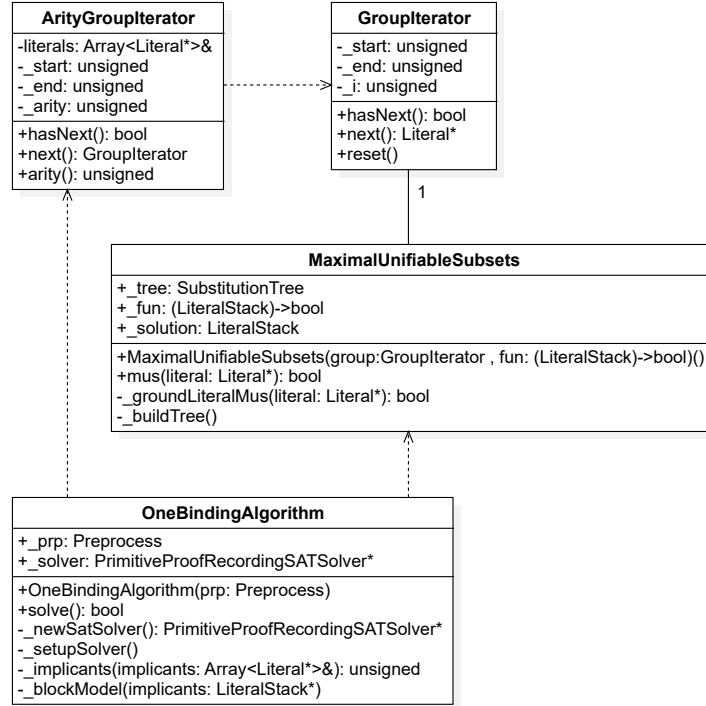


Figura 1.2: Struttura dell'algoritmo di decisione

1.2.1 Implicants Sorting

1.2.2 Maximal Unifiable Subsets

Algorithm 1: Maximal Unifiable Subsets

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

1.2.3 Algoritmo Finale

1.3 Algoritmo di Classificazione

(Input formula rettificata senza true e false)

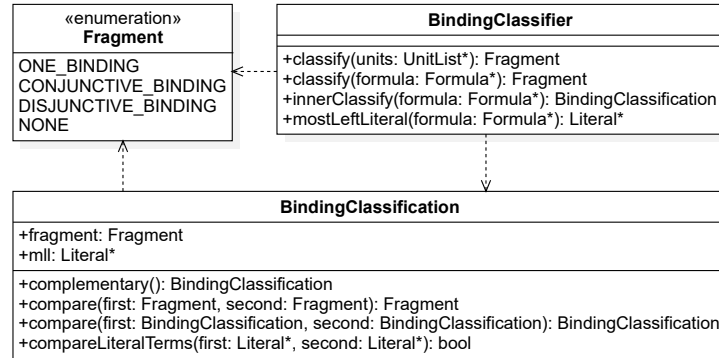


Figura 1.3: Classificatore

Algorithm 2: Classificatore esterno

Firma: $\text{classify}(\varphi)$ **Input:** φ Una formula rettificata

Output: Un elemento dell'enumerazione Fragment

```
switch  $\varphi$  do
  case Literal do
    | return ONE_BINDING;
  end
  case  $A[\wedge, \vee]B$  do
    | return  $\text{compare}(\text{classify}(A), \text{classify}(B))$ ;
  end
  case  $\neg A$  do
    | return  $\text{classify}(A).\text{complementary}()$ ;
  end
  case  $[\forall, \exists]A$  do
    |  $\text{sub} := \varphi$ ;
    |  $\text{connective} :=$  connective of  $\varphi$ ;
    | repeat
    |   |  $\text{sub} :=$  subformula of sub;
    |   |  $\text{connective} :=$  connective of sub;
    | until  $\text{connective} \notin \{\forall, \exists\}$ ;
    |  $(\text{fragment}, -) := \text{innerClassify}(\text{sub})$ ;
    | return  $\text{fragment}$ ;
  end
  case  $A \Leftrightarrow B$  do
    | return  $\text{compare}(\text{classify}(A \Rightarrow B), \text{classify}(B \Rightarrow A))$ ;
  end
  case  $A \oplus B$  do
    | return  $\text{classify}(A \Leftrightarrow B).\text{complementary}()$ ;
  end
  case  $A \Rightarrow B$  do
    | return  $\text{compare}(\text{classify}(\neg A), \text{classify}(B))$ ;
  end
end
```

Algorithm 3: Classificatore interno

Firma: $\text{innerClassify}(\varphi)$ **Input:** φ Una formula rettificata

Output: Una coppia (Fragment, Literal)

```
switch  $\varphi$  do
  case Literal  $l$  do
    | return (ONE_BINDING,  $l$ );
  end
  case  $A[\wedge, \vee]B$  do
    | return innerCompare(innerClassify( $A$ ), innerClassify( $B$ ), 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( $B$ ), connective of  $\varphi$ );
  end
  else
    | return (None, null);
  end
end
```

Algorithm 4: Compare esterno

Firma: $\text{compare}(A, B)$ **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: $\text{innerCompare}(A, B, \text{con})$ **Input:** A, B due coppie (Fragment, Literal), con un connettivo

Output: Una coppia (Fragment, Literal)

```
switch  $A.\text{first}, B.\text{first}, \text{con}$  do
  case  $\text{One\_Binding}, \text{One\_Binding}, \_$  do
    if  $A.\text{second}$  has same terms of  $B.\text{second}$  then
      | return  $A$ ;
    end
    else if  $\text{conn} = \wedge$  then
      | return  $(\text{Conjunctive\_Binding}, \text{null})$ ;
    end
    else if  $\text{conn} = \vee$  then
      | return  $(\text{Disjunctive\_Binding}, \text{null})$ ;
    end
  end
  case  $[\text{One\_Binding}, \text{Conjunctive\_Binding} \mid \text{Conjunctive\_Binding}, \text{One\_Binding}], \wedge$  do
    | return  $(\text{Conjunctive\_Binding}, \text{null})$ ;
  end
  case  $[\text{One\_Binding}, \text{Disjunctive\_Binding} \mid \text{Disjunctive\_Binding}, \text{One\_Binding}], \vee$  do
    | return  $(\text{Disjunctive\_Binding}, \text{null})$ ;
  end
  case  $\text{Conjunctive\_Binding}, \text{Conjunctive\_Binding}, \wedge$  do
    | return  $(\text{Conjunctive\_Binding}, \text{null})$ ;
  end
  case  $\text{Disjunctive\_Binding}, \text{Disjunctive\_Binding}, \vee$  do
    | return  $(\text{Disjunctive\_Binding}, \text{null})$ ;
  end
end
return  $(\text{None}, \text{null})$ ;
```
