Bounded Connection Tableau

November 21, 2018

1 Basic Definitions

Flat equation A *flat equation* is an equation of the form $a = b$.

Functional Equation A *functional equation* (f-equation) is a equation of the form $f(\bar{a}) = b$.

Literal A *literal* is a (negated) propositional variable, or a (negated) flat equation.

2 Complex Definitions

2.1 Pseudo-Literal

3 Flattening

If we have the formula $\forall x.R(f(x)) = b$, we must flatten it, we therefore have $\forall x \exists c_1(R(c_1) = b \wedge f(x) = c_1)$.

4 Algorithms

4.1 Prove

Input: Set of clauses Output: Proof OR Invalid

Algorithm 1: Closable

Input: Input Clauses

```
1 eqs ←equations(branch) for l ∈ branch do
2 |  if complementary(head(branch), l) then
3 |  |  if unifiable_eqs(head(branch), l) then
4 |  |  |  return True;
5 |  |  end
6 |  end
7 end
8 return False;
```

4.2 Closable

Algorithm 2: Closable

Input: branch

```
1 eqs ←equations(branch) for l ∈ branch do
2 |  if complementary(head(branch), l) then
3 |  |  if unifiable_eqs(head(branch), l) then
4 |  |  |  return True;
5 |  |  end
6 |  end
7 end
8 return False;
```

Closable checks whether a branch can be closed. The predicate $unifiable_E(p, q)$ is true if literals p and q are unifiable modulo the set of equations E. This is were we use our BREU-procedure (though in the singular case, one can use simple congruence closure).

4.2.1 Todo

- Do we need local or global closability?
- Maybe have a congurence closure procedure here instead?

4.3 Closing Branch w. Clause

Algorithm 3: Close Branch w. Clause

Input: branch, startStep the first step to try, φ a clause
Result: Result after an extension step closing branch using φ with index or None

```
1  maxStep ← |φ|;
2  for i ∈ [startStep..maxStep] do
3  |  newBranch ← φ(i) :: branch;
4  |  if closable(newBranch) then
5  |  |  newClause ←rearrange(φ, i);
6  |  |  newBranches ← extend(branch,newClause);
7  |  |  return (newBranches, i);
8  |  end
9  end
10 return (None, maxStep);
```

The purpose of *Close Branch w. Clause* (see Alg. 2) is to try and use a clause to close a branch. The result will often be a new subtree, since the extension step will create one branch per literal of the clause used. If the clause is a unit-clause, a singleton branch will be returned however.

The function call *closable* can either be used in the *local* sense, where it only considers if this branch can be closed in isolation, or in a *global* sense where the current context constrains possible substitutions.

4.3.1 Todo

- Do we need to store the substitution received from *closable*.
- Do we care about what pair of literals are closing the branch? Since we are using connection tableaux we can assume that the leaf of a branch is always used for closing a branch.

4.4 Close Table

Algorithm 4: Search

Input: table a table, Γ a set of clauses
Result: table extended to a closed tableau using clauses $\in \Gamma$

```
1  openBranches ← openBranches(table);
2  if openBranches = 0 then
3  |  return table
4  end
5  branch ←pickBranch(table);
6  for clause ∈ Γ do
7  |  for i ∈ 1..|clause| do
8  |  |  ret ← CloseBranchWClause(branch, clause,i);
9  |  |  if ret ≠ None then
10 |  |  |  newTable ← replaceBranch(table, branch, ret);
11 |  |  |  return CloseTable(newTable, Γ);
12 |  |  end
13 |  end
14 end
15 return None
```

4.4.1 Todo

- Change index for literals in clause to subroutine returning a list instead?