Contextual Rewriting in SPASS

Christoph Weidenbach and Patrick Wischnewski

Max-Planck-Institut für Informatik,
Campus E 1.4, Saarbrücken, Germany,
{weidenb,wischnew}@mpi-inf.mpg.de

Abstract. Sophisticated reductions are an important means to achieve progress in automated theorem proving. We consider the powerful reduction rule *Contextual Rewriting* in the context of the superposition calculus. If the rule is considered in its abstract, most general form, the applicability of contextual rewriting is not decidable. We develop a decidable instance of the general contextual rewriting rule and implement it in SPASS. An experimental evaluation on the TPTP gives first insights into the application potential of the rule instance.

1 Introduction

In the superposition context, first-order theorem proving with equality deals with the problem of showing unsatisfiability of a (finite) set N of clauses. This problem is well-known to be undecidable, in general. It is semi-decidable in the sense that superposition is refutationally complete. The superposition calculus is composed of inference and reduction rules. Inference rules generate new clauses from N whereas reduction rules delete clauses from N or transform them into simpler ones. If, in particular, powerful reduction rules are available, decidability of certain subclasses of first-order logic can be shown and explored in practice [1–4]. Hence, sophisticated reductions are an important means for progress in automated theorem proving. In this paper the reduction rule *Contextual Rewriting* is considered in the context of the superposition calculus [5]. Contextual rewriting extends rewriting with unit equations to rewriting with full clauses containing a positive orientable equation. In order to apply such a clause for rewriting, all other literals of that clause have to be entailed by the context of the clause to be rewritten and potentially further clauses from a given clause set. Hence, the name contextual rewriting.

For a first, simplified example consider the two clauses

$$P(x) \rightarrow f(x) \approx x \qquad S(g(a)), a \approx b, P(b) \rightarrow R(f(a))$$

where we write clauses in implication form [6]. Now in order to rewrite $R(f(a))$ in the second clause to $R(a)$ using the equation $f(x) \approx x$ of the first clause with matcher $\sigma = \{x \mapsto a\}$, we have to show that $P(x)\sigma$ holds in the context of the second clause $S(g(a)), a \approx b, P(b)$, i.e., $\models S(g(a)), a \approx b, P(b) \rightarrow P(x)\sigma$. This obviously holds, so we can replace $S(g(a)), a \approx b, P(b) \rightarrow R(f(a))$ by

$S(g(a)), a \approx b, P(b) \to R(a)$ via a contextual rewriting application of $P(x) \to f(x) \approx x$.

More general, contextual rewriting is the following rule:

$$\mathcal{R}\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C = (\Gamma_2 \to \Delta_2)[u[s\sigma] \approx v]}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ (\Gamma_2 \to \Delta_2)[u[t\sigma] \approx v]\end{array}}$$

where $(\Gamma_2 \to \Delta_2)[u[s\sigma] \approx v]$ expresses that $u[s\sigma] \approx v$ is an atom occurring in Γ_2 or Δ_2 and u contains the subterm $s\sigma$. Contextual rewriting reduces the subterm $s\sigma$ of u to $t\sigma$ if, among others, the following conditions are satisfied

$$N_C \models \Gamma_2 \to A \text{ for all } A \text{ in } \Gamma_1\sigma$$
$$N_C \models A \to \Delta_2 \text{ for all } A \text{ in } \Delta_1\sigma$$

where N is the current clause set, $C, D \in N$, and N_C denotes the set of clauses from N smaller than C with respect to a reduction ordering $\prec$, total on ground terms. Reduction rules are labeled with an $\mathcal{R}$ and are meant to replace the clauses above the bar by the clauses below the bar. Both side conditions are undecidable, in general. Therefore, in order to make the rule applicable in practice, it must be instantiated such that eventually these two conditions become effective. This is the topic of this paper.

For a more sophisticated, further motivating example, consider the following clause set. It can be finitely saturated using contextual rewriting but not solely with less sophisticated reduction mechanisms such as unit rewriting or subsumption.

Let i, q, r, c be functions, a, b be constants and x_1, x_2, x_3, x_4, y_1 be variables and let $r \succ c \succ q \succ i \succ b \succ a \succ nil$ using the KBO with weight 1 for all function symbols and variables.

1: $\quad \to \quad q(nil) \approx b$

2: $\quad i(x_1) \approx b, q(y_1) \approx b \quad \to \quad q(r(x_1, y_1)) \approx b$

3: $\quad i(x_1) \approx b, q(y_1) \approx b \quad \to \quad q(c(x_1, y_1)) \approx a$

4: $\quad i(x_1) \approx b, q(y_1) \approx b, i(x_3) \approx b \quad \to \quad r(x_3, c(x_1, y_1)) \approx c(x_1, r(x_3, y_1))$

5: $\quad i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \quad \to \quad y_1 \approx nil, q(c(x_1, c(x_2, r(x_3, y_1)))) \approx b$

If we apply superposition right between clause 4 and clause 5 on the term $q(c(x_1, c(x_2, r(x_3, y_1))))$ we obtain the clause

6: $i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, i(x_4) \approx b, q(y_1) \approx b, q(c(x_4, y_1)) \approx b, b \approx a \to c(x_4, y_1) \approx nil,\ q(c(x_1, c(x_2, c(x_4, r(x_3, y_1))))) \approx b$

which is larger (both in the ordering and the number of symbols) than clause 5. Applying superposition between clause 4 and clause 6 yields an even larger clause. Repeating the superposition inference between clause 4 and these clauses creates larger and larger clauses. Hence, the exhaustive application of the superposition calculus does not terminate on this clause set. Furthermore, none of the reductions which have been implemented so far in SPASS and in any

other system we are aware of, can reduce clause 5.[1] However, with contextual rewriting we can reduce clause 5 using clause 3 to

$$7\text{: } i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \rightarrow y_1 \approx nil, a \approx b.$$

Clause 7 is a tautology and can be reduced to true. Then the set is saturated since no further superposition inference is possible. In order to apply contextual rewriting we have to verify the side conditions

$$N_C \models i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \rightarrow i(x_1) \approx b$$

and

$$N_C \models i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \rightarrow q(c(x_2, r(x_3, y_1))) \approx b.$$

The first condition holds trivially and the latter follows from clause 3 and clause 2.

In this work we presents an instance of contextual rewriting that reduces the above clause set, is decidable and feasible for practical problem instances. We tested our implementation on all problems of the *TPTP* library version 3.2.0 [7]. In Section 2 we develop a practically useful instance of contextual rewriting called *Approximated Contextual Rewriting*. Section 3 presents the implementation of approximated contextual rewriting in SPASS. The final section discusses experimental results.

2 Contextual Rewriting

We consider first-order logic with equality using notation from [6]. We write clauses in the form $\Gamma \rightarrow \Delta$ where Γ and Δ are multi-sets of atoms. The atoms of Γ denote negative literals while the atoms of Δ denote the positive literals. A substitution σ is a mapping from the set of variables to the set of terms such that $x\sigma \neq x$ for only finitely many variables x. The reduction rules, in particular the contextual rewriting rule, are defined with respect to a well-founded reduction ordering $\prec$ on terms that is total on ground terms. This ordering is then lifted to literals and clauses in the usual way [6]. A term s is called *strictly maximal* in $\Gamma \rightarrow \Delta$ if there is no different occurrence of a term in $\Gamma \rightarrow \Delta$ that is greater or equal than s with respect to $\prec$.

Contextual rewriting is a sophisticated reduction rule originally introduced in [5] that generalizes unit rewriting and non-unit rewriting [6]. It is an instance of the standard redundancy notion of superposition. A clause C is called *redundant* in a clause set N if there exist clauses $C_1, \ldots, C_n \in N$ with $C_i \prec C$ for $i \in \{1, \ldots, n\}$, written $C_i \in N_C$, such that $C_1, \ldots, C_n \models C$. The clause C is implied by smaller clauses from N. This condition can actually be refined to grounding substitutions: C is redundant if for all grounding substitutions σ for C there are ground instances $C_i\sigma_i$ of clauses $C_i \in N$ such that $C_i\sigma_i \prec C\sigma$, written $C_i\sigma_i \in N_{C\sigma}$, and $C_1\sigma_1, \ldots, C_n\sigma_n \models C\sigma$. Reduction rules are marked with an $\mathcal{R}$ and their application replaces the clauses above the bar with the clauses below the bar.

[1] Actually, the SATURATE system contained the first implementation of contextual rewriting, but it never matured to be widely usable.

Definition 1 (Contextual Rewriting [5]). *Let N be a clause set, $C, D \in N$, σ be a substitution then the reductions*

$$\mathcal{R}\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C = \Gamma_2, u[s\sigma] \approx v \to \Delta_2}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ C'' = \Gamma_2, u[t\sigma] \approx v \to \Delta_2\end{array}}$$

$$\mathcal{R}\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C = \Gamma_2 \to \Delta_2, u[s\sigma] \approx v}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ C'' = \Gamma_2 \to \Delta_2, u[t\sigma] \approx v\end{array}}$$

where the following conditions are satisfied

1. $s\sigma \succ t\sigma$
2. $C \succ D\sigma$
3. $N_C \models \Gamma_2 \to A$ *for all* A *in* $\Gamma_1\sigma$
4. $N_C \models A \to \Delta_2$ *for all* A *in* $\Delta_1\sigma$

are called contextual rewriting.

Due to condition 1-1 and condition 1-2 we have $C'' \prec C$ and $D\sigma \prec C$. Then from condition 1-3 and condition 1-4 we obtain that there exist clauses $C_1, \ldots, C_n \in N_C$ and $C_1, \ldots, C_n, C'', D\sigma \models C$. Therefore, the clause C is redundant in $N \cup \{C''\}$ and can be eliminated. The rule is an instance of the abstract superposition redundancy notion.

The side conditions 1-3 and 1-4 having both the form $N_C \models \Gamma \to \Delta$ are undecidable, in general. There are two sources for the undecidability. First, there are infinitely possible grounding substitutions σ' for the clause $\Gamma \to \Delta$ and C. Second, for a given σ' there may be infinitely many δ with $C_i\delta \prec C\sigma'$, $C_i \in N$. Therefore, in the following in order to effectively decide the side conditions, we will fix one σ' and restrict the number of considered substitutions δ yielding a decidable instance of contextual rewriting.

First, $N_C \models \Gamma \to \Delta$ is equivalent to $N_C \cup \{\exists \boldsymbol{x}.\neg(\Gamma \to \Delta)\} \models \bot$. The existential quantifier can be eliminated by Skolemization yielding a Skolem substitution τ that maps each variable out of $\boldsymbol{x}$ to a new Skolem constant. Consequently, setting σ' to τ yields the instance $N_C \models (\Gamma \to \Delta)\tau$, where $(\Gamma \to \Delta)\tau$ is ground. Still there may exist infinitely many δ with $C_i\delta \prec C\tau$, $C_i \in N$ and $C\tau$ may still contain variables.

Therefore, we restrict δ to those grounding substitutions that map variables to terms only occurring in $C\tau$ or $D\sigma\tau$ where we additionally assume that τ is also grounding for C and $D\sigma$, i.e., it maps any variable occurring in C or $D\sigma$ to an arbitrary fresh Skolem constant. Let $N_{C\tau}^{D\sigma\tau}$ be the set of all ground instances of clauses from N smaller than $C\tau$ obtained by instantiation with ground terms from $D\sigma\tau, C\tau$. Then $N_{C\tau}^{D\sigma\tau}$ is finite and $N_{C\tau}^{D\sigma\tau} \subseteq N_{C\tau}$. Consequently, $N_{C\tau}^{D\sigma\tau} \models (\Gamma \to \Delta)\tau$ is a sufficient ground approximation of $N_C \models \Gamma \to \Delta$. Even

though this is a decidable approximation of the original problem the set $N_{C\tau}^{D\sigma\tau}$ is exponentially larger than N, in general. Therefore, we represent $N_{C\tau}^{D\sigma\tau}$ implicitly by approximating $N_{C\tau}^{D\sigma\tau} \models (\Gamma \to \Delta)\tau$ by the application of a reduction calculus $(\Gamma \to \Delta)\tau \vdash_{Red} \top$. The reduction calculus $\vdash_{Red}$ is composed of a set of reduction rules containing *tautology reduction*, *forward subsumption*, *obvious reduction* and a particular instance of contextual rewriting called *recursive contextual ground rewriting* defined below. Tautology reduction reduces syntactic and semantic tautologies to true whereas forward subsumption reduces subsumed clauses to true. Obvious reduction eliminates trivial literals [6].

The reduction calculus $\vdash_{Red}$ only needs to reduce ground clauses. Therefore, the following definition introduces an instance of contextual rewriting only working on ground clauses. Further, it adapts contextual rewriting such that it implicitly considers clauses from $N_{C\tau}^{D\sigma\tau}$.

Definition 2 (Recursive Contextual Ground Rewriting). *If N is a clause set, $D \in N$, C' ground, σ a substitution then the reduction*

$$\mathcal{R}\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C' = \Gamma_2, u[s\sigma] \approx v \to \Delta_2}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ \Gamma_2, u[t\sigma] \approx v \to \Delta_2\end{array}}$$

$$\mathcal{R}\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C' = \Gamma_2 \to \Delta_2, u[s\sigma] \approx v}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ \Gamma_2 \to \Delta_2, u[t\sigma] \approx v\end{array}}$$

where the following conditions are satisfied

1. *$s\sigma$ is a strictly maximal term in $D\sigma$*
2. *$u[s\sigma] \approx v \succ s\sigma \approx t\sigma$*
3. *$vars(s) \supset vars(D)$*
4. *$(\Gamma_2 \to A) \vdash_{Red} \top$ for all A in $\Gamma_1\sigma$*
5. *$(A \to \Delta_2) \vdash_{Red} \top$ for all A in $\Delta_1\sigma$*

is called recursive contextual ground rewriting.

Condition 2-1 and condition 2-2 ensure the ordering restrictions required by contextual rewriting. Condition 2-3 implies that $D\sigma$ is ground. A clause D meeting condition 2-1 and condition 2-3 is called *strongly universally reductive*. Condition 2-4 and condition 2-5 recursively apply the reduction calculus.

The reduction calculus $\vdash_{Red}$ is terminating since C' is reduced to a smaller ground clause. As a consequence, also the reduction relation $\vdash_{Red}$ is terminating. The approximated contextual rewriting rule eventually becomes the below rule.

Definition 3 (Approximated Contextual Rewriting). *Let N be a clause set, $C, D \in N$, σ be a substitution then the reductions*

$$\mathcal{R}\,\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C = \Gamma_2, u[s\sigma] \approx v \to \Delta_2}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ C'' = \Gamma_2, u[t\sigma] \approx v \to \Delta_2\end{array}}$$

$$\mathcal{R}\,\frac{D = \Gamma_1 \to \Delta_1, s \approx t \qquad C = \Gamma_2 \to \Delta_2, u[s\sigma] \approx v}{\begin{array}{c}\Gamma_1 \to \Delta_1, s \approx t \\ C'' = \Gamma_2 \to \Delta_2, u[t\sigma] \approx v\end{array}}$$

where the following conditions are satisfied

1. $s\sigma \succ t\sigma$
2. $C \succ D\sigma$
3. τ *maps all variables from* $C, D\sigma$ *to fresh Skolem constants*
4. $(\Gamma_2 \to A)\tau \vdash_{Red} \top$ *for all* A *in* $\Gamma_1\sigma$
5. $(A \to \Delta_2)\tau \vdash_{Red} \top$ *for all* A *in* $\Delta_1\sigma$

are called approximated contextual rewriting.

Note that unit rewriting and non-unit rewriting [6] are also instances of the approximated contextual rewriting rule. Note further that the conditions for the approximated contextual rewriting rule are weaker compared to the recursive contextual ground rewriting rule: the right premise needs not to be ground and the equation $s \approx t$ needs not to be maximal in the first premise. Approximated contextual rewriting uses recursive contextual ground rewriting to effectively decide the side conditions.

3 Implementation

The implementation of SPASS [6] focuses on a sophisticated reduction machinery. This machinery completely interreduces all clauses as it performs *forward reduction* and *backward reduction* whenever a clause is newly generated or modified. Forward reduction reduces the newly generated clause using the previously generated clauses and backward rewriting reduces the previously generated clauses with the new one.

The integration of contextual rewriting into this machinery consists of two steps. First, the search for appropriate contextual rewrite application candidates is analogous to the case of unit rewriting and non-unit rewriting. In addition, the side conditions of contextual rewriting have to be checked. Finding appropriate contextual rewrite application candidates can be solved by standard term indexing [8]. This functionality has already been implemented into SPASS via substitution trees [6, 9].

Second, validating the side conditions of contextual rewriting requires an effective implementation of the reduction calculus $\vdash_{Red}$. First of all, it is too costly to explicitly compute the Skolem substitution τ for each clause $\Gamma \to$

```
1  RecursiveValidityCheck(CLAUSE C, CLAUSE SET N);
2  Rewritten=TRUE;
3  while Rewritten do
4  |  Rewritten=FALSE;
5  |  if IsEmpty(C) then return FALSE;
6  |  if IsTautology(C) then  return TRUE ;
7  |  if ForwardSubsumption(C, N) then return TRUE;
8  |  if ObviousReduction(C) then Rewritten=TRUE;
9  |  if RecursiveContextualGroundRewriting(C, N) then
   |     Rewritten=TRUE ;
10 end
11 return FALSE
```

Algorithm 1: RecursiveValidityCheck

Δ which the reduction calculus considers for contextual rewriting. Applying τ explicitly requires to allocate memory for the new constants and the new clause and it requires additional computations to build the clause. Because of the recursive structure of the reduction calculus this is not feasible. Therefore, the idea is to simply treat variables as constants for this case. We adapt the ordering modules (KBO, RPOS) such that they can treat variables as constants. Since variables are not contained in the precedence we have to define an ordering on them. In Spass variables are represented by integers which implicitly gives an ordering on variables. Whenever we consider variables to be constants we assume them to have a lower precedence than any other symbol of the signature. Comparisons between variables are performed on the bases of their integer value. As a result, the implementation provides a method for applying τ to a clause $\Gamma \rightarrow \Delta$ without any computation or memory allocation.

The composition of the reduction calculus $\vdash_{Red}$ to an actual decision procedure is depicted in Algorithm 1. The algorithm uses tautology deletion, forward subsumption and obvious reductions from the reduction procedure of Spass. The procedures have to work with respect to the modified, above explained, orderings. Besides of this the implementation of these reductions remains unchanged.

Algorithm 1 expects as input a clause C and a clause set N and reduces C with respect to N in the main loop. IsEmpty(C) checks whether the given clause is the empty clause, IsTautology(C) checks whether C is either a syntactic or a semantic tautology. ForwardSubsumption(C, N) checks whether C is already subsumed by clauses from N and ObviousReduction(C) removes duplicated literals and trivial equations from a clause. Further details can be found in the Spass Handbook [10].

RecursiveContextualGroundRewriting(C, N), depicted in Algorithm 2, subsumes unit and non-unit rewriting and implements recursive contextual ground rewriting. The variables occurring in C are interpreted as constants in the above explained sense. The call to $general_{SDT}(N, u')$ returns the set of generalizations G from N of u' and the respective matcher σ. Then the procedure computes for each of the generalizations the literals and the clauses where they occur resulting

```
1  RecursiveContextualGroundRewriting(CLAUSE C[u[u'] ≈ v],
                                      CLAUSE SET N);
2  G = general_SDT(N, u');
3  foreach (s, σ) ∈ G do
4  |  Lits = LiteralsContainingTerm(s);
5  |  foreach (s ≈ t) ∈ Lits do
6  |  |  D = LiteralOwningClause(s ≈ t);
7  |  |  if (vars(s) ⊃ vars(D) ∧
8  |  |        u[sσ] ≈ v ≻ sσ ≈ tσ
9  |  |        sσ strictly maximal term in Dσ ∧
10 |  |        ∀A ∈ Ante(Dσ) RecursiveValidityCheck(Γ → A) ∧
11 |  |        ∀A ∈ Succ(Dσ) RecursiveValidityCheck(A → Δ) ) then
12 |  |  |  return C[u[tσ] ≈ v];
13 |  |  end
14 |  end
15 end
```

Algorithm 2: RecursiveContextualGroundRewriting

in the contextual rewrite candidates. The candidate clauses are then checked for the non-recursive side conditions of contextual rewriting. If these hold, RecursiveContextualGroundRewriting builds the subproblems and recursively calls RecursiveValidityCheck.

The implementation of approximated contextual rewriting is analogous to the implementation of recursive contextual ground rewriting. The difference is that the input clause C is not interpreted as ground and the local side conditions (line 7 - line 9) are changed with respect to the definition of approximated contextual rewriting.

4 Results

4.1 Results on the TPTP

The *TPTP 3.2.0* [7] is a library consisting of 8984 problems for automated theorem proving systems. We compared the Spass version 3.1 that is version 3.0 extended by some bug fixes, containing our implementation of contextual rewriting to Spass version 3.1 without contextual rewriting.

Table 1 depicts the results. We compare two runs of Spass with a reference run. All runs were performed with Spass options set to *-RFRew=4 -RBRew=2 -RTaut=2* on Opteron nodes running at speed of 2.4 GHz equipped with 4 GB RAM for each node. For the reference run we let Spass run on the TPTP with contextual rewriting turned off and a time limit of 300 seconds. The first run of Spass with contextual rewriting activated and a 300 seconds time bound found 77 additional proofs and lost 151 proofs compared to the run with contextual rewriting disabled. There were no significant differences among the versions on satisfiable problems. The second run of Spass with contextual

Time	Won	Lost	Deceleration Coefficient	Solved Open Problems
300	77	151	1.46	2
600	122	133	1.62	5

Table 1. Forward contextual rewriting

rewriting and a 600 seconds time bound found 122 additional proofs and lost 133 proofs compared to the run with contextual rewriting disabled. Additionally, we computed the average running time of the reference run and each of the other two runs. We considered only those cases where both the reference run and the respective test run terminated and the version with contextual rewriting took at least 10 seconds. The running time slows down on the average at a coefficient of 1.46 for the 300 seconds run and at a coefficient of 1.62 for the 600 seconds run.

The problems where SPASS with contextual rewriting found a proof and the reference run did not were mostly problems with a high TPTP difficulty rating. In the run with 300 seconds our instances even terminated on two problems with rating 1.00 and with 600 seconds on five problems with rating 1.00 meaning that no system has been able to solve these problems so far. The problems are SWC308+1, SWC335+1, in the 300 seconds time bound and additionally in the 600 seconds time bound SWC329+1, SWC342+1, SWC345+1. All proofs were checked by the SPASS proof checker.

It was both a surprise to us that the contextual rewriting version did not improve on satisfiable problems and that it lost so many unsatisfiable problems. For the satisfiable problems this is simply due to the fact that the TPTP does not contain suitable problems. For the unsatisfiable ones we inspected some in detail and figured out that the actual proof found by the standard version got lost through a contextual rewriting application. This is not a new phenomenon, however, it is surprising that it occurs so often on the TPTP. We will further dig into this hoping to find further insight.

4.2 Application to the Example from the Introduction

In this part we depict the application of approximated recursive contextual rewriting on the introductory example in detail. Therewith, we show that superposition together with our instance of contextual rewriting terminates on this example. SPASS with contextual rewriting is able to saturate the clause set from section 1 whereas SPASS without contextual rewriting is not. Recall that we can reduce clause 5 with clause 3 using contextual rewriting if the side conditions are fulfilled. The ground clauses

$8 : i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \rightarrow i(x_1) \approx b$

$9 : i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, b \approx a \rightarrow q(c(x_2, r(x_3, y_1))) \approx b$

must be entailed by clauses from N_C. Clause 8 is a tautology whereas clause 9 can be rewritten with clause 3 to

10: $i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, a \approx b \rightarrow a \approx b$

using contextual rewriting if in addition the clauses

11: $i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, a \approx b \rightarrow i(x_2) \approx b$

12: $i(x_1) \approx b, i(x_3) \approx b, i(x_2) \approx b, q(y_1) \approx b, a \approx b \rightarrow q(r(x_3, y_1)))) \approx b$

are also entailed by clauses from N_C. Clause 11 is a syntactic tautology and clause 12 is subsumed by clause 2.

5 Summary

In summary, contextual rewriting is costly but it helps solving difficult problems. Our current implementation offers reasonable room for improvement. For example, in SPASS, contextual rewriting is implemented independently from unit and non-unit rewriting such that in case of non-applicability of the rule, the same checks and indexing queries are repeated.

For subsumption nice filters are known to detect non-applicability. It would be worthwhile to search for such filters for contextual rewriting. Eventually, we need to better understand why we lost surprisingly many problems from the TPTP. One possible answer could be that the standard SPASS heuristics for inference selection don't work well anymore with contextual rewriting. The SPASS version described in this paper and used for the experiments can be obtained from the SPASS homepage (`http://spass-prover.org/`) following the prototype link.

References

1. Bachmair, L., Ganzinger, H., Waldmann, U.: Superposition with simplification as a decision procedure for the monadic class with equality. In: Computational Logic and Proof Theory. Volume 713 of LNCS. (1993) 83–96
2. Hustadt, U., Schmidt, R.A., Georgieva, L.: A survey of decidable first-order fragments and description logics. Journal of Relational Methods in Computer Science **1** (2004) 251–276
3. Jacquemard, F., Meyer, C., Weidenbach, C.: Unification in extensions of shallow equational theories. In: Proceedings of RTA-98. Volume 1379 of LNCS., Springer (1998) 76–90
4. Ganzinger, H., de Nivelle, H.: A superposition decision procedure for the guarded fragment with equality. In: LICS. (1999) 295–304
5. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. Journal of Logic and Computation **4**(3) (1994) 217–247
6. Weidenbach, C.: Combining superposition, sorts and splitting. In Robinson, A., Voronkov, A., eds.: Handbook of Automated Reasoning. Volume 2. Elsevier (2001) 1965–2012
7. Sutcliffe, G., Suttner, C.: The TPTP Problem Library: CNF Release v1.2.1. Journal of Automated Reasoning **21**(2) (1998) 177–203
8. Ramakrishnan, I.V., Sekar, R.C., Voronkov, A.: Term indexing. In Robinson, J.A., Voronkov, A., eds.: Handbook of Automated Reasoning. Elsevier and MIT Press (2001) 1853–1964
9. Graf, P.: Term Indexing. Volume 1053 of LNAI. Springer-Verlag (1995)
10. Weidenbach, C., Schmidt, R., Keen, E.: Spass handbook version 3.0. Contained in the documentation of SPASS Version 3.0 (2007)