Set-valued field language to 1st order logic conversion rules

Charles Bouillaguet
charles.bouillaguet@dptinfo.ens-cachan.fr

May 10, 2006

Language of incomming formulae :

$$
\begin{array}{rcl}
O & ::= & V_O | \operatorname{null} | O.f_O \\
S & ::= & V_S | S \cap S | S \cup S | S \setminus S | \{O, \ldots, O\} | O.f_S \\
f_O & ::= & V_{f_O} | f_O \left[O \longrightarrow O \right] \\
f_S & ::= & V_{f_S} | f_S \left[O \longrightarrow S \right] \\
A & ::= & O = O | S = S | O \in S | \operatorname{Card}(S) \leq k | f_O = f_O | f_S = f_S \\
F & ::= & A | F \wedge F | F \vee F | \neg F | \top | \bot
\end{array}
$$

Figure 1: Syntax of expresions and formulas

1 Unnesting for dummies

Cardinalities constraints Just expresses the fact that the set is equal to a finite set of cardinality k. All the introduced variables are fresh.

Card-Constraint

$$\frac{\operatorname{Card}(S) \leq k}{\exists (x_1, \ldots, x_k), S \subseteq \{x_1, \ldots, x_k\}} \quad k \in \mathbb{N}$$

Set inclusion and equality We unfold the definition. After that, we only have set membership.

Set-Inclusion

$$\frac{S_1 \subseteq S_2}{\forall x, x \in S_1 \Rightarrow x \in S_2}$$

Set-Equality

$$\frac{S_1 = S_2}{\forall x, x \in S_1 \iff x \in S_2}$$

Complex set expressions We unfold the definition. After that, the only set expressions are set variables or set-valued fields.

Intersection

$$\frac{x \in S_1 \cap S_2}{x \in S_1 \wedge x \in S_2}$$

Union

$$\frac{x \in S_1 \cup S_2}{x \in S_1 \vee x \in S_2}$$

Difference

$$\frac{x \in S_1 \setminus S_2}{x \in S_1 \wedge x \notin S_2}$$

FiniteSet

$$\frac{x \in \{O_1, \ldots, O_k\}}{x = O_1 \vee \cdots \vee x = O_k}$$

Field Equalities we can unfold the definitions, and flaten everything to be in a finite number of case. Note that the set expressions that appears here may be complex, as they have not been processed by the rules above. Note that we can't write existential quantifiers on fields variables, so there must NOT be universal quantifier above in the formula. Just introducing a fresh constant symbol as we do is like doing some kind of immediate skolemization, assuming there are not universally quantified variables above.

Object-Field-Variables-Equality

$$\frac{V^1_{f_O} = V^2_{f_O}}{\forall x, x.V^1_{f_O} = x.V^2_{f_O}}$$

Object-Field-Write-Equality-Rectify

$$\frac{f^2_O\,[O_1 \longrightarrow O_2] = V^1_{f_O}}{V^1_{f_O} = f^2_O\,[O_1 \longrightarrow O_2]}$$

Object-Field-Write-Flattening

$$\frac{f^1_O\,[O_1 \longrightarrow O_2] = f^2_O\,[O_3 \longrightarrow O_4]}{f^1_O\,[O_1 \longrightarrow O_2] = V^{fresh}_{f_O} \wedge V^{fresh}_{f_O} = f^2_O\,[O_3 \longrightarrow O_4]}$$

Set-Field-Variables-Equality

$$\frac{V^1_{f_S} = V^2_{f_S}}{\forall x, y, x \in y.V^1_{f_S} \iff x \in y.V^2_{f_S}}$$

Set-Field-Write-Equality-Rectify

$$\frac{f^2_S\,[O \longrightarrow S] = V^1_{f_S}}{V^1_{f_S} = f^2_S\,[O \longrightarrow S]}$$

Set-Field-Write-Flattening

$$\frac{f^1_S\,[O_1 \longrightarrow S_1] = f^2_S\,[O_2 \longrightarrow S_2]}{V^{fresh}_{f_S} = f^1_S\,[O_1 \longrightarrow S_1] \wedge V^{fresh}_{f_S} = f^2_S\,[O_2 \longrightarrow S_2]}$$

Object-Field-Write-Equality-Unfolding

$$\frac{V^1_{f_O} = f^2_O\,[O_1 \longrightarrow O_2]}{\forall (x, y : Obj), x.V^1_{f_O} = y \iff (x = O_1 \wedge y = O_2) \vee \left(x \neq O_1 \wedge y = x.f^2_O\right)}$$

Set-Field-Write-Equality-Unfolding

$$\frac{V^1_{f_S} = f^2_S\,[O \longrightarrow S]}{\forall (x, y : Obj), y \in x.V^1_{f_S} \iff (x = O \wedge y \in S) \vee \left(x \neq O \wedge y \in x.f^2_S\right)}$$

Field Writes We instanciate the previous UNFOLDING rules.

Object-Field-Write-Read

$$\frac{V^1_O = V^2_O.f_O\,[O_1 \longrightarrow O_2]}{\left(V^2_O = O_1 \wedge V^1_O = O_2\right) \vee \left(V^2_O \neq O_1 \wedge V^1_O = V^2_O.f_O\right)}$$

Set-Field-Write-Read

$$\frac{V^1_O \in V^2_O.f_S\,[O \longrightarrow S]}{\left(V^2_O = O \wedge V^1_O \in S\right) \vee \left(V^2_O \neq O \wedge V^1_O \in V^2_O.f_S\right)}$$

Flattening We must find any remaining field write : for this purpose we flatten everything. This may trigger the two previous rules, and the set expressions rules.

Object-Valued-Field-Flattening

$$\frac{O_1.f_O^1 = O_2.f_O^2}{\exists x, x = O_1.f_O^1 \wedge x = O_2.f_O^2}$$

Objet-Flattening

$$\frac{V_O = O_1.f_O.V_{f_O}}{\exists x, x = O_1.f_O \wedge V_O = x.V_{f_O}}$$

Membership-Flattening-Right

$$\frac{V_O \in O.f_O.f_S}{\exists x, x = O.f_O \wedge V_O \in x.f_S}$$

Membership-Flattening-Left

$$\frac{V_O.f_O \in S}{\exists x, x = V_O.f_O \wedge x \in S}$$

S not a set expression

1st order logic Now we're finally ready to finish the conversion to first-order logic.

Field-Deferencing

$$\frac{V_O^1 = V_O^2.V_{f_O}}{V_{f_O}\left(V_O^2, V_O^1\right)}$$

Variable-Membership

$$\frac{V_O \in V_S}{V_S\left(V_O\right)}$$

Field-Membership

$$\frac{V_O^1 \in V_O^2.V_S}{V_S\left(V_O^2, V_O^1\right)}$$

Equality-Normalization

$$\frac{V_O^1.V_{f_O} = V_O^2}{V_O^2 = V_O^1.V_{f_O}}$$

I chose to instanciate the Field Writes rules instead of flatening the field writes because it generates less hypothesis.

Figure 2 shows a summary of what's happening.

Termination is clear.

Confluence is trivial because for a given atom, there is no situation where two different rules could be applied (at least, if we enforce the S in the OBJECT-VALUED-FIELD-FLATENING rule not to be a complex set expression).

Correcteness (i.e. that once the rewriting has terminated the result is a FO formula) is less evident.

1. When no rule from the Field Equality section can be applied, then there is no field equality in the formula. Although the UNFOLDING rules only apply to equality which left-side term is a field variable, the RECTIFY and FLATTENING rules are enough to put all the field equalities in the desired form.

2. the trickiest point is to see that all the field writes disappear in the process. This is achieved by the recursive interaction between the "Field

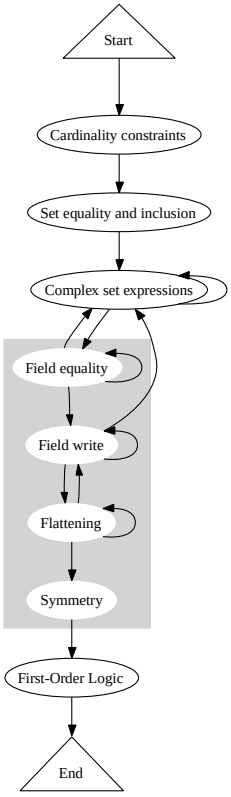

Figure 2: Possibles triggering of rules

write" and the "Flattening" sets of rules. A field write cannot be hidden in the left-hand side of an equality or membership atom because of the OBJECT-VALUED-FIELD-FLATTENING and MEMBERSHIP-FLATTENING-LEFT rules. It cannot be the first deferenced field because of the WRITE-READ rules. However, if we have multiple cascaded field deferences, they will be flattened by the OBJECT-FLATTENING and MEMBERSHIP-FLATTENING-RIGHT rules.This way, all the fields occuring in the formula will occur as the first and only field deference of an object variable, and the READ-WRITE rules then apply.

The real implementation is summarized by figure 3

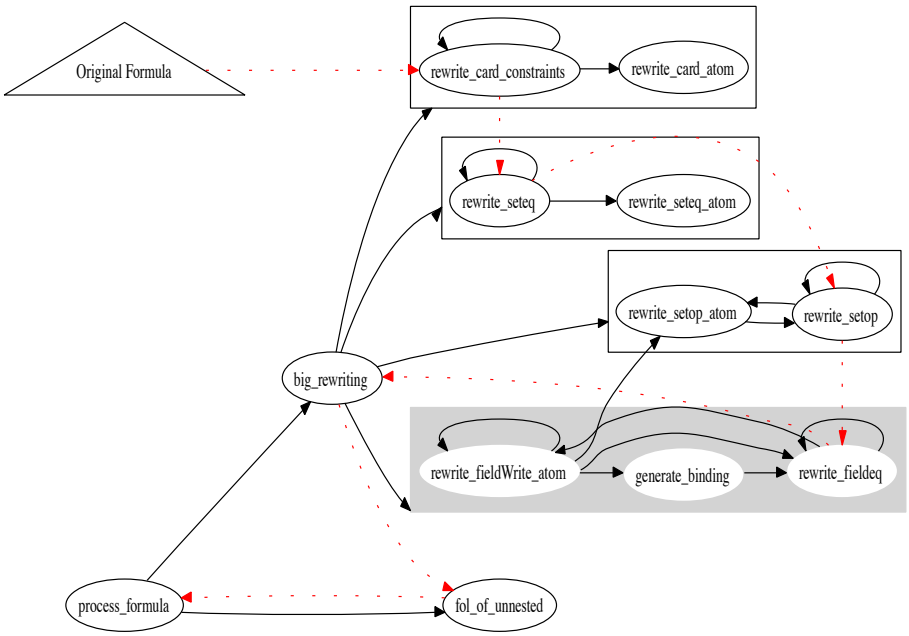

Figure 3: Call graph of the real implementation (black arrows), and result flow graph (red arrows)