Online appendix for the paper

Proving Correctness of Imperative Programs by Linearizing Constrained Horn Clauses

published in Theory and Practice of Logic Programming

EMANUELE DE ANGELIS, FABIO FIORAVANTI
DEC, University 'G. d'Annunzio', Pescara, Italy
(*e-mail:* {emanuele.deangelis,fabio.fioravanti}@unich.it)

ALBERTO PETTOROSSI
DICII, Università di Roma Tor Vergata, Roma, Italy
(*e-mail:* pettorossi@disp.uniroma2.it)

MAURIZIO PROIETTI
CNR-IASI, Roma, Italy
(*e-mail:* maurizio.proietti@iasi.cnr.it)

submitted 29 April 2015; revised 3 July 2015; accepted 14 July 2015

Appendix A

For the proof of Theorem 1 we need the following lemma.

Lemma 1. (i) The relation r_{prog} defined by *OpSem* is a functional relation, that is, $M(OpSem) \models \forall p_1,\ldots,p_s, y_1, y_2. r_{prog}(p_1,\ldots,p_s,y_1) \wedge r_{prog}(p_1,\ldots,p_s,y_2) \rightarrow y_1 = y_2$.
(ii) A program *prog* terminates for an environment δ_0 such that $\delta_0(z_1) = p_1,\ldots, \delta_0(z_s) = p_s$ and $pre(p_1,\ldots,p_s)$ holds, iff
$M(OpSem) \models pre(p_1,\ldots,p_s) \rightarrow \exists y.\ r_{prog}(p_1,\ldots,p_s,y)$.

Proof. Since the program *prog* is deterministic, the predicate r_{prog} defined by *OpSem* is a functional relation (which might not be total on *pre*, as *prog* might not terminate). Moreover, a program *prog*, with variables $z_1,\ldots,z_s$, terminates for an environment δ_0 such that: (i) $\delta_0(z_1) = p_1,\ \ldots,\ \delta_0(z_s) = p_s$, and (ii) δ_0 satisfies *pre*, iff $\exists y.\ r_{prog}(p_1,\ldots,p_s,y)$ holds in $M(OpSem)$. □

Proof of Theorem 1 (*Partial Correctness*).
Let $dom_r(X_1,\ldots,X_s)$ be a predicate that represents the *domain* of the functional relation r_{prog}. We assume that $dom_r(X_1,\ldots,X_s)$ is defined by a set *Dom* of clauses, using predicate symbols not in $OpSem \cup Spec$, such that

$$M(OpSem \cup Dom) \models \forall X_1,\ldots,X_s.((\exists Y.r_{prog}(X_1,\ldots,X_s,Y) \leftrightarrow dom_r(X_1,\ldots,X_s)) \quad (1)$$

Let us denote by $Spec^\sharp$ the set of clauses obtained from *Spec* by replacing each clause $f(X_1,\ldots,X_s,Y) \leftarrow B$ by the clause $f(X_1,\ldots,X_s,Y) \leftarrow dom_r(X_1,\ldots,X_s), B$. Then, for all integers $p_1,\ldots,p_s,y$,

$$M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, y) \text{ implies } M(Spec) \models f(p_1, \ldots, p_s, y) \quad (2)$$

Moreover, let us denote by $Spec'$ the set of clauses obtained from $Spec^\sharp$ by replacing all occurrences of f by r_{prog}. We show that $M(OpSem \cup Aux \cup Dom) \models Spec'$.
Let S be any clause in $Spec'$. If S belongs to Aux, then $M(OpSem \cup Aux) \models S$. Otherwise, S is of the form $r_{prog}(X_1, \ldots, X_s, Y) \leftarrow dom_r(X_1, \ldots, X_s), \widetilde{B}$ and, by construction, in F_{pcorr} there are two goals

G_1: $false \leftarrow Y > Z, r_{prog}(X_1, \ldots, X_s, Z), \widetilde{B}$, and
G_2: $false \leftarrow Y < Z, r_{prog}(X_1, \ldots, X_s, Z), \widetilde{B}$

such that $OpSem \cup Aux \cup \{G_1, G_2\}$ is satisfiable. Then,

$$M(OpSem \cup Aux) \models \neg\exists(Y \neq Z \wedge r_{prog}(X_1, \ldots, X_s, Z) \wedge \widetilde{B})$$

Since $M(OpSem \cup Dom) \models r_{prog}(X_1, \ldots, X_s, Z) \rightarrow dom_r(P_1, \ldots, P_s)$, we also have that

$$M(OpSem \cup Aux \cup Dom) \models \neg\exists(Y \neq Z \wedge dom_r(X_1, \ldots, X_s) \wedge r_{prog}(X_1, \ldots, X_s, Z) \wedge \widetilde{B})$$

From the functionality of r_{prog} it follows that

$$M(OpSem \cup Aux \cup Dom) \models \neg r_{prog}(X_1, \ldots, X_s, Y) \leftrightarrow (\neg\exists Z \cdot r_{prog}(X_1, \ldots, X_s, Y) \vee (r_{prog}(X_1, \ldots, X_s, Z) \wedge Y \neq Z))$$

and hence, by using (1),

$$M(OpSem \cup Aux \cup Dom) \models \neg\exists(dom_r(X_1, \ldots, X_s) \wedge \neg r_{prog}(X_1, \ldots, X_s, Y) \wedge \widetilde{B})$$

Thus, we have that

$$M(OpSem \cup Aux \cup Dom) \models \forall(dom_r(X_1, \ldots, X_s) \wedge \widetilde{B} \rightarrow r_{prog}(X_1, \ldots, X_s, Y))$$

that is, clause S is true in $M(OpSem \cup Aux \cup Dom)$. We can conclude that $M(OpSem \cup Aux \cup Dom)$ is a model of $Spec' \cup Dom$, and since $M(Spec' \cup Dom)$ is the *least* model of $Spec' \cup Dom$, we have that

$$M(Spec' \cup Dom) \subseteq M(OpSem \cup Aux \cup Dom) \quad (3)$$

Next we show that, for all integers $p_1, \ldots, p_s, y$,

$$M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, y) \quad \text{iff} \quad M(OpSem) \models r_{prog}(p_1, \ldots, p_s, y) \quad (4)$$

Only If Part of (4). Suppose that $M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, y)$. Then, by construction,

$$M(Spec' \cup Dom) \models r_{prog}(p_1, \ldots, p_s, y)$$

and hence, by (3),

$$M(OpSem \cup Aux \cup Dom) \models r_{prog}(p_1, \ldots, p_s, y)$$

Since r_{prog} does not depend on predicates in $Aux \cup Dom$,

$$M(OpSem) \models r_{prog}(p_1, \ldots, p_s, y)$$

If Part of (4). Suppose that $M(OpSem) \models r_{prog}(p_1, \ldots, p_s, y)$.
Then, by definition of r_{prog},

$$M(Dom) \models dom_r(p_1, \ldots, p_s) \quad (5)$$

and

$$M(Spec) \models pre(p_1, \ldots, p_s) \quad (6)$$

Thus, by (6) and Condition (3.1) of Definition 1, there exists z such that

$M(Spec) \models f(p_1, \ldots, p_s, z)$ (7)

By (5) and (7),

$M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, z)$ (8)

By the *Only If Part* of (4),

$M(OpSem) \models r_{prog}(p_1, \ldots, p_s, z)$

and by the functionality of r_{prog}, $z = y$. Hence, by (8),

$M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, y)$

Let us now prove partial correctness. If $M(Spec) \models pre(p_1, \ldots, p_s)$ and *prog* terminates, that is, $M(Dom) \models dom_r(p_1, \ldots, p_s)$, then for some integer y, $M(OpSem) \models r_{prog}(p_1, \ldots, p_s, y)$. Thus, by (4), $M(Spec^\sharp \cup Dom) \models f(p_1, \ldots, p_s, y)$ and hence, by (2), $M(Spec) \models f(p_1, \ldots, p_s, y)$. Suppose that the postcondition ψ is $f(p_1, \ldots, p_s, z_k)$. Then, by Condition (3.2) of Definition 1, $y = z_k$.

Thus, $\{\varphi\}$ *prog* $\{\psi\}$. □

Removal of the Interpreter

Here we report the variant of the transformation presented in (De Angelis et al. 2014a) that we use in this paper to perform the removal of the interpreter. In this transformation we use the function $Unf(C, A, Cls)$ defined as the set of clauses derived by unfolding a clause C with respect to an atom A using the set Cls of clauses (see the unfolding rule in Section 4.2).

The predicate *reach* is defined as follows:

$reach(X, X) \leftarrow$

$reach(X, Z) \leftarrow tr(X, Y),\ reach(Y, Z)$

where, as mentioned in Section 2, *tr* is a (nonrecursive) predicate representing one transition step according to the operational semantics of the imperative language.

In order to perform the UNFOLDING step, we assume that the atoms occurring in bodies of clauses are annotated as either *unfoldable* or *not unfoldable*. This annotation ensures that any sequence of clauses constructed by unfolding w.r.t. unfoldable atoms is finite. In particular, the atoms with predicate *initCf*, *finalCf*, and *tr* are unfoldable. The atoms of the form $reach(cf_1, cf_2)$ are unfoldable if cf_1 is not associated with a while or goto command. Other annotations based on a different analysis of program *OpSem* can be used.

Input: Program *OpSem*.
Output: Program $OpSem_{RI}$ such that, for all integers $p_1,\ldots,p_s,z_k$,
$r_{prog}(p_1,\ldots,p_s,z_k) \in M(OpSem)$ iff $r_{prog}(p_1,\ldots,p_s,z_k) \in M(OpSem_{RI})$.

INITIALIZATION:
$OpSem_{RI} := \emptyset$; $Defs := \emptyset$;
$InCls := \{r_{prog}(P_1,\ldots,P_s,Z_k) \leftarrow initCf(C_0,P_1,\ldots,P_s), reach(C_0,C_h), finalCf(C_h,Z_k)\}$;
while in *InCls* there is a clause C which is not a constrained fact *do*

UNFOLDING:
$SpC := Unf(C, A, OpSem)$, where A is the leftmost atom in the body of C;
while in SpC there is a clause D whose body contains an occurrence of an unfoldable atom A *do*
$SpC := (SpC - \{D\}) \cup Unf(D, A, OpSem)$
end-while;

DEFINITION & FOLDING:
while in SpC there is a clause E of the form: $H \leftarrow e, reach(cf_1, cf_2)$
do
if in *Defs* there is no clause of the form: $newp(V) \leftarrow reach(cf_1, cf_2)$
where V is the set of variables occurring in $reach(cf_1, cf_2)$
then add the clause N: $newp(V) \leftarrow reach(cf_1, cf_2)$ to *Defs* and *InCls*;
$SpC := (SpC - \{E\}) \cup \{H \leftarrow e, newp(V)\}$
end-while;
$InCls := InCls - \{C\}$; $OpSem_{RI} := OpSem_{RI} \cup SpC$;
end-while;

RI: Removal of the Interpreter.

Let us now prove Theorem 3 stating the relevant properties of the RI transformation.

The RI transformation terminates. The termination of the UNFOLDING step is guaranteed by the *unfoldable* annotations. Indeed, (i) the repeated unfolding of the unfoldable atoms with predicates *initCf*, *finalCf*, and *tr*, always terminates because those atoms have no recursive clauses, (ii) by the definition of the semantics of the imperative program, the repeated unfolding of an atom of the form $reach(cf_1, cf_2)$ eventually derives a new $reach(cf_3, cf_4)$ atom where cf_3 is either a final configuration or a configuration associated with a while or goto command, and in both cases unfolding terminates. The termination of the DEFINITION & FOLDING step follows from the fact that SpC is a finite set of clauses.

The outer while loop terminates because a finite set of new predicate definitions of the form $newp(V) \leftarrow reach(cf_1, cf_2)$ can be introduced. Indeed, each configuration cf is represented as a term `cf(LC,E)`, where `LC` is a labeled command and `E` is an environment (see Example 1). An environment is represented as a list of (v, X) pairs where v is a variable identifier and X is its value, that is, a logical variable whose

value may be subject to a given constraint. Considering that: (i) the labeled commands and the variable identifiers occurring in an imperative program are finitely many, and (ii) predicate definitions of the form $newp(V) \leftarrow reach(cf_1, cf_2)$ abstract away from the constraints that hold on the logical variables occurring in cf_1 and cf_2, we can conclude that there are only finitely many such clauses (modulo variable renaming).

Point 1: *$OpSem_{RI}$ is a set of linear clauses over the integers.* By construction, every clause in $OpSem_{RI}$ is of the form $H \leftarrow c, B$, where (i) H is either $r_{prog}(P_1, \ldots, P_s, Z_k)$ or $newp(V)$, for some new predicate $newp$ and tuple of variables V, and (ii) B is either absent or of the form $newp(V)$, for some new predicate $newp$ and tuple of variables V. Thus, every clause is a linear clause over the integers.

Point 2: *$OpSem \cup Aux \cup F_{pcorr}$ is satisfiable iff $OpSem_{RI} \cup Aux \cup F_{pcorr}$ is satisfiable.* From the correctness of the unfolding, definition, and folding rules with respect to the least model semantics of CLP programs (Etalle and Gabbrielli 1996), it follows that, for all integers $p_1, \ldots, p_s, z_k$,

$r_{prog}(p_1, \ldots, p_s, z_k) \in M(OpSem)$ iff $r_{prog}(p_1, \ldots, p_s, z_k) \in M(OpSem_{RI})$ (†1)

$OpSem \cup Aux \cup F_{pcorr}$ is satisfiable iff for every ground instance G of a goal in F_{pcorr}, $M(OpSem \cup Aux) \models G$. Since the only predicate of $OpSem$ on which G may depend is r_{prog}, by (†1), we have that $M(OpSem \cup Aux) \models G$ iff $M(OpSem_{RI} \cup Aux) \models G$. Finally, $M(OpSem_{RI} \cup Aux) \models G$ for every ground instance G of a goal in F_{pcorr}, iff $OpSem_{RI} \cup Aux \cup F_{pcorr}$ is satisfiable.

Point 3: *$OpSem \cup Aux \cup F_{pcorr}$ is LA-solvable iff $OpSem_{RI} \cup Aux \cup F_{pcorr}$ is LA-solvable.*

Suppose that $OpSem \cup Aux \cup F_{pcorr}$ is LA-solvable, and let Σ be an LA-solution of $OpSem \cup Aux \cup F_{pcorr}$. Now we construct an LA-solution Σ_{RI} of $OpSem_{RI} \cup Aux \cup F_{pcorr}$. To this purpose it is enough to define a symbolic interpretation for the new predicates introduced by RI.

For any predicate $newp$ introduced by RI via a clause of the form:

$newp(V) \leftarrow reach(cf_1, cf_2)$

we define a symbolic interpretation as follows:

$\Sigma_{RI}(newp(V)) = \Sigma(reach(cf_1, cf_2))$

Moreover, Σ_{RI} is identical to Σ for the atoms with predicate occurring in $OpSem$.

Now we have to prove that Σ_{RI} is indeed an LA-solution of $OpSem_{RI} \cup Aux \cup F_{pcorr}$. This proof is similar to the proof of Theorem 5 (actually, simpler, because RI introduces new predicates defined by single atoms, while LIN introduces new predicates defined by conjunctions of atoms), and is omitted.

Vice versa, if Σ_{RI} is an LA-solution of $OpSem_{RI} \cup Aux \cup F_{pcorr}$, we construct an LA-solution Σ of $OpSem \cup Aux \cup F_{pcorr}$ by defining

$\Sigma(reach(cf_1, cf_2)) = \Sigma_{RI}(newp(V))$. □

Proof of Theorem 4

Let $LCls$ be a set of linear clauses and Gls be a set of nonlinear goals. We split the proof of Theorem 4 in three parts:

Termination: The linearization transformation LIN terminates for the input set of clauses $LCls \cup Gls$;
Linearity: The output *TransfCls* of LIN is a set of linear clauses;
Equisatisfiability: $LCls \cup Gls$ is satisfiable iff *TransfCls* is satisfiable.

(*Termination*) Each UNFOLDING and DEFINITION & FOLDING step terminates. Thus, in order to prove the termination of LIN it is enough to show that the while loop is executed a finite number of times, that is, a finite number of clauses are added to *NLCls*. We will establish this finiteness property by showing that there exists an integer M such that every clause added to *NLCls* is of the form:

$$newp(X_1, \ldots, X_t) \leftarrow A_1, \ldots, A_k \qquad (\dagger 2)$$

where: (i) $k \leq M$, (ii) for $i = 1, \ldots, k$, A_i is of the form $p(X_1, \ldots, X_m)$, and (iii) $\{X_1, \ldots, X_t\} \subseteq vars(A_1, \ldots, A_k)$.

Indeed, let M be the maximal number of atoms occurring in the body of a goal in *Gls*, to which *NLCls* is initialized. Now let us consider a clause C in *NLCls* and assume that in the body of C there are at most M atoms. The clauses in the set *LCls* used for unfolding C are linear, and hence in the body of each clause belonging to the set $U(C)$ obtained after the UNFOLDING step, there are at most M atoms. Thus, each clause in $U(C)$ is of the form $H \leftarrow c, A_1, \ldots, A_k$, with $k \leq M$. Since the body of every new clause introduced by the subsequent DEFINITION & FOLDING step is obtained by dropping the constraint from the body of a clause in $U(C)$, we have that every clause added to *NLCls* is of the form (†2), with $k \leq M$. Thus, LIN terminates.

(*Linearity*) *TransfCls* is initialized to the set *LCls* of linear clauses. Moreover, each clause added to *TransfCls* is of the form $H \leftarrow c$, $newp(X_1, \ldots, X_t)$, and hence is linear.

(*Equisatisfiability*) In order to prove that LIN ensures equisatisfiability, let us adapt to our context the basic notions about the unfold/fold transformation rules for CLP programs presented in (Etalle and Gabbrielli 1996).

Besides the unfolding rule of Section 4.2, we also introduce the following *definition* and *folding* rules.

Definition Rule. By definition we introduce a clause of the form $newp(X) \leftarrow G$, where *newp* is a new predicate symbol and X is a tuple of variables occurring in G.

Folding Rule. Given a clause E: $H \leftarrow c, G$ and a clause D: $newp(X) \leftarrow G$ introduced by the definition rule. Suppose that, $X = vars(G) \cap vars(H,c)$. Then by folding E using D we derive $H \leftarrow c, newp(X)$.

From a set *Cls* of clauses we can derive a new set *TransfCls* of clauses either by adding a new clause to *Cls* using the definition rule or by: (i) selecting a clause C in *Cls*, (ii) deriving a new set *TransfC* of clauses using one or more transformation rules among unfolding and folding, and (iii) replacing C by *TransfC* in *Cls*. We can apply a new sequence of transformation rules starting from *TransfCls* and iterate this process at will.

The following theorem is an immediate consequence of the correctness results for the unfold/fold transformation rules of CLP programs (Etalle and Gabbrielli 1996).

Theorem 6 (*Correctness of the Transformation Rules*)
Let the set *TransfCls* be derived from *Cls* by a sequence of applications of the unfolding, definition and folding transformation rules. Suppose that every clause introduced by the definition rule is unfolded at least once in this sequence. Then, *Cls* is satisfiable iff *TransfCls* is satisfiable.

Now, equisatisfiability easily follows from Theorem 6. Indeed, the UNFOLDING and DEFINITION & FOLDING steps of LIN are applications of the unfolding, definition, and folding rules (strictly speaking, the rewriting performed after unfolding is not included among the transformation rules, but obviously preserves all LA-models). Moreover, every clause introduced during the DEFINITION & FOLDING step is added to *NCls* and unfolded in a subsequent step of the transformation. Thus, the hypotheses of Theorem 6 are fulfilled, and hence we have that $LCls \cup Gls$ is satisfiable iff *TransfCls* is satisfiable. □

Linearized clauses for Fibonacci.
The set of *linear* constrained Horn clauses obtained after applying LIN is made out of clauses `E1`, `E2`, `E3`, and `C3`, together with the following clauses:

```
new1(N1,U,V,U,N2,U,N3,U):- N1=<0, N2=<0, N3=<0.
new1(N1,U,V,U,N2,U,N3,F3):- N1=<0, N2=<0, N4=N3-1, W=U+V, N3>=1,new2(N4,W,U,F3).
new1(N1,U,V,U,N2,F2,N3,U):- N1=<0, N4=N2-1, W=U+V, N2>=1, N3=<0,new2(N4,W,U,F2).
new1(N1,U,V,U,N2,F2,N3,F3):- N1=<0, N4=N2-1, N2>=1, N5=N3-1, N3>=1,
          new3(N4,W,U,F2,N5,F3).
new1(N1,U,V,F1,N2,U,N3,U):- N4=N1-1, W=U+V, N1>=1, N2=<0, N3=<0,new2(N4,W,U,F1).
new1(N1,U,V,F1,N2,U,N3,F3):- N4=N1-1, N1>=1, N2=<0, N5=N3-1, W=U+V, N3>=1,
          new3(N4,W,U,F1,N5,F3).
new1(N1,U,V,F1,N2,F2,N3,U):- N4=N1-1, N1>=1, N5=N2-1, W=U+V, N2>=1, N3=<0,
          new3(N4,W,U,F1,N5,F2).
new1(N1,U,V,F1,N2,F2,N3,F3):- N4=N1-1, N1>=1, N5=N2-1, N2>=1, N6=N3-1, W=U+V,
          N3>=1, new1(N4,W,U,F1,N5,F2,N6,F3).
new2(N,U,V,U):- N=<0.
new2(N,U,V,F):- N2=N-1, W=U+V, N>=1, new2(N2,W,U,F).
new3(N1,U,V,U,N2,U):- N1=<0, N2=<0.
new3(N1,U,V,U,N2,F2):- N1=<0, N3=N2-1, W=U+V, N2>=1, new2(N3,W,U,F2).
new3(N1,U,V,F1,N2,F2):- N3=N1-1, N1>=1, N4=N2-1, W=U+V, N2>=1,
          new3(N3,W,U,F1,N4,F2).
new3(N1,U,V,F1,N2,U):- N3=N1-1, W=U+V, N1>=1, N2=<0, new2(N3,W,U,F1).
```

Proof of Theorem 5 (*Monotonicity with respect to LA-Solvability*).
Suppose that the set $LCls \cup Gls$ of constrained Horn clauses is LA-solvable, and let *TransfCls* be obtained by applying LIN to $LCls \cup Gls$. Let Σ be an LA-solution of $LCls \cup Gls$. We now construct an LA-solution of *TransfCls*. For any predicate *newp* introduced by LIN via a clause of the form:

$newp(X_1, \ldots, X_t) \leftarrow A_1, \ldots, A_k$

we define a symbolic interpretation Σ' as follows:

$\Sigma'(newp(X_1, \ldots, X_t)) = \Sigma(A_1) \wedge \ldots \wedge \Sigma(A_k)$

Now, we are left with the task of proving that Σ' is indeed an LA-solution of *TransfCls*. The clauses in *TransfCls* are either of the form

$\quad$ *false* $\leftarrow c, newq(X_1, \ldots, X_u)$

or of the form

$\quad newp(X_1, \ldots, X_t) \leftarrow c, newq(X_1, \ldots, X_u)$

where *newp* and *newq* are predicates introduced by LIN. We will only consider the more difficult case where the conclusion is not *false*.

The clause $newp(X_1, \ldots, X_t) \leftarrow c, newq(X_1, \ldots, X_u)$ has been derived (see the linearization transformation LIN in Figure 2) in the following two steps.

(Step i) Unfolding $newp(X_1, \ldots, X_t) \leftarrow A_1, \ldots, A_k$ w.r.t. all atoms in its body using k clauses in $LCls$:

$\quad A_1 \leftarrow c_1, B_1 \quad \ldots \quad A_k \leftarrow c_k, B_k$

where some of the B_i's can be the *true* and $c \equiv c_1, \ldots, c_k$, thereby deriving

$\quad newp(X_1, \ldots, X_t) \leftarrow c_1, \ldots, c_k, B_1, \ldots, B_k$

(Without loss of generality we assume that the atoms in the body of the clauses are equal to, instead of *unifiable* with, the heads of the clauses in $LCls$.)

(Step ii) Folding $newp(X_1, \ldots, X_t) \leftarrow c_1, \ldots, c_k, B_1, \ldots, B_k$ using a clause of the form:

$\quad newq(X_1, \ldots, X_u) \leftarrow B_1, \ldots, B_k$

Thus, for $newq(X_1, \ldots, X_u))$ we have the following symbolic interpretation:

$\quad \Sigma'(newq(X_1, \ldots, X_u)) = \Sigma(B_1) \wedge \ldots \wedge \Sigma(B_k)$

To prove that Σ' is an LA-solution of *TransfCls*, we have to show that

$\quad LA \models \forall(c \wedge \Sigma'(newq(X_1, \ldots, X_u)) \rightarrow \Sigma'(newp(X_1, \ldots, X_t)))$

Assume that

$\quad LA \models c \wedge \Sigma'(newq(X_1, \ldots, X_u))$

Then, by definition of Σ',

$\quad LA \models c \wedge \Sigma(B_1) \wedge \ldots \wedge \Sigma(B_k)$

Since Σ is an LA-solution of $LCls$, we have that:

$\quad LA \models \forall(c_1 \wedge \Sigma(B_1) \rightarrow \Sigma(A_1)) \quad \ldots \quad LA \models \forall(c_k \wedge \Sigma(B_k) \rightarrow \Sigma(A_k))$

and hence

$\quad LA \models \Sigma(A_1) \wedge \ldots \wedge \Sigma(A_k)$

Thus, by definition of Σ',

$\quad LA \models \Sigma'(newp(X_1, \ldots, X_t))$. $\square$