SOCS

A COMPUTATIONAL LOGIC MODEL FOR THE DESCRIPTION, ANALYSIS AND VERIFICATION OF GLOBAL AND OPEN SOCIETIES OF HETEROGENEOUS COMPUTEES

IST-2001-32530

Proof of properties of the $\mathcal{S}$CIFF proof-procedure

Project number:	IST-2001-32530
Project acronym:	SOCS
Document type:	D (deliverable)
Document distribution:	I (internal to SOCS and PO)
CEC Document number:	IST32530/DIFERRARA/410/D/I/b1
File name:	6 410-b1[SCIFFprop].pdf
Editor:	Marco Gavanelli
Contributing partners:	DIFERRARA, UNIBO
Contributing workpackages:	WP5
Estimated person months:	N/A
Date of completion:	7 March 2005
Date of delivery to the EC:	18 March 2005
Number of pages:	54

ABSTRACT

We provide the proof of some of the properties of the $\mathcal{S}$CIFF proof-procedure.

Proof of properties of the $\mathcal{S}$CIFF proof-procedure

Marco Gavanelli♣, **Evelina Lamma**♣, **and Paola Mello**♠

♣ Dipartimento di Ingegneria, Università di Ferrara
♠ DEIS, Università di Bologna
Email: `[m gavanelli,e lamma]@ing·unife·it, p mello@deis·unibo·it`

ABSTRACT
We provide the proof of some of the properties of the $\mathcal{S}$CIFF proof-procedure.

Contents

1 Introduction

In this document, we collect the proofs of the main properties of the $\mathcal{S}$CIFF proof-procedure. First, we give a short recap of the society formal model. We report the proof of Soundness (Part I), that was proven in a document [3] annexed to D12. Then, we give the proof of Termination (Part II), extending the proof of termination of the IFF proof-procedure [10].

2 Society formal model: Recap

We provide here, for the sake of readability, the main parts of the declarative and operational semantics of the society. These parts are meant to make this report more self-contained, and are a synthesis of corresponding parts taken from Deliverable D8 [4].

The new work consists of the proofs of Theorems 3 and 4, which are given in Section 6.5.

2.1 The Syntax of the Society

The society knowledge consists of the following 4-tuple [8]:

$$\langle SOKB, SEKB, \mathcal{IC}_S, \mathcal{G}\rangle$$

where:

- $SOKB$ is the *Social Organization Knowledge Base*,
- $SEKB$ is the *Social Environment Knowledge Base*,
- $\mathcal{IC}_S$ is the set of *Social Integrity Constraints* (IC$_S$), and
- $\mathcal{G}$ is the set of *Goals* of the society.

Social Environment Knowledge Base. The $SEKB$ dynamically evolves and is composed of:

- *Happened events*: atoms indicated with functor **H**;
- *Expectations* on the future: events that should (but might not) happen in the future (atoms indicated with functor **E**), and events that should not (but might indeed) happen in the future (atoms indicated with functor **NE**).

The happened events are represented as ground atoms

$$\mathbf{H}(\mathit{Event}[, \mathit{Time}]).$$

The expectations can be

$$\mathbf{E}(\mathit{Event}[, \mathit{Time}])$$

$$\mathbf{NE}(\mathit{Event}[, \mathit{Time}])$$

and can contain variables, with the following scope rules and quantifications:

- variables in **E** atoms are always existentially quantified with scope the entire set of expectations

- the other variables, that occur only in **NE** atoms are universally quantified (the scope of universally quantified variables is not important, as $\forall X.p(X) \wedge q(X)$ is equivalent to $\forall X.p(X) \wedge \forall Y.q(Y)$).

Social Organization Knowledge Base. The SOKB is a logic program, consisting of clauses

$$\begin{array}{rcl} Clause & ::= & Atom \leftarrow Body \\ Body & ::= & ExtLiteral\ [\ \wedge ExtLiteral\ ]^{\star} \\ ExtLiteral & ::= & Literal \mid Expectation \mid Constraint \\ Expectation & ::= & [\neg]\mathbf{E}(Event\ [, T]) \mid [\neg]\mathbf{NE}(Event\ [, T]) \end{array} \tag{1}$$

In a clause, the variables are quantified as follows:

- Universally, if they occur only in literals with functor **NE** (and possibly constraints), with scope the body;
- Otherwise universally, with scope the entire *Clause*.

We call *definite* the predicates for which there exists a definition; i.e., a predicate that occurs in at least the head of a clause.

Goal. The goal $\mathcal{G}$ of the society has the same syntax as the *Body* of a clause in the SOKB. Notice that the variables occurring in G are considered *free* by the IFF proof procedure. In the devised proof procedure for the society infrastructure (named $\mathcal{S}$CIFF in the following), for ease of presentation and without loss of generality, they will be considered as existentially (or where appropriate as universally) quantified variables. Coherently with the SOKB, the variables occurring in $\mathcal{G}$ are quantified

- *existentially* if they occur in a definite literal, or literals with functor **E**;
- *universally* if they occur only in literals with functor **NE**.

Any variable in $\mathcal{G}$ must occur in at least a literal **E** or **NE**.

Social Integrity Constraints are in the form of implications. We report here, for better readability, the characterizing part of their syntax (the full syntax is given in document D5):

$$\begin{array}{rcl} ic & ::= & \chi \rightarrow \phi \\ \chi & ::= & (HEvent | Expectation)\ [\wedge BodyLiteral]^{\star} \\ BodyLiteral & ::= & HEvent | Expectation | Literal | Constraint \\ \phi & ::= & HeadDisjunct\ [\ \vee HeadDisjunct\ ]^{\star} | \bot \\ HeadDisjunct & ::= & Expectation\ [\ \wedge (Expectation | Constraint)]^{\star} \\ Expectation & ::= & [\neg]\mathbf{E}(Event\ [, T]) \mid [\neg]\mathbf{NE}(Event\ [, T]) \\ HEvent & ::= & [\neg]\mathbf{H}(Event\ [, T]) \end{array} \tag{2}$$

Given a ic_S $\chi \rightarrow \phi$, χ is called the *body* (or the *condition*) and ϕ is called the *head* (or the *conclusion*).

The rules of scope and quantification are as follows:

1. Any variable in an ic_S must occur in at least an *Event* or in an *Expectation*.

2. The variables that occur both in the body and in the head are quantified universally with scope the entire ic_S.
3. The variables that occur only in the head must occur in at least one *Expectation*, and
   (a) if they occur in literals **E** or ¬**E** are quantified existentially and have as scope the disjunct they belong to;
   (b) otherwise they are quantified universally.
4. The variables that occur only in the body are quantified inside the body as follows:
   (a) if they occur only in conjunctions of ¬**H**, **NE**, ¬**NE** or *Constraint*s are quantified universally;
   (b) otherwise are quantified existentially.
5. Of course, the order of the quantifiers is, in general, significant. In our syntax, the quantifier $\forall$ cannot be followed by $\exists$.

2.2 Allowedness Conditions

We report here, for the sake of completeness, the allowedness conditions introduced in deliverable D8 [4].

We extend the IFF proof procedure allowedness condition, as follows.

Definition 1. *A clause* $Head \leftarrow Body$ *is* allowed *if every variable that occurs in a negative literal of a definite predicate*

- *occurs in at least a positive literal or in the head (as in the IFF)*
- *or it occurs in atoms* **E** *or* ¬**E**.

A Goal is Allowed *if every variable that occurs in a negative literal of a definite predicate*

- *occurs in at least a positive literal (as in the IFF)*
- *or it occurs in atoms* **E** *or* ¬**E**

The aim of these definitions is to ensure that the quantification of variables in negative literals in the resolvent cannot be universal.

The IFF also imposes a condition on the integrity constraints:

> An integrity constraint is allowed if (and only if) every variable in the conclusion occurs in the condition.

We do not need this condition, because we can always convert a non allowed integrity constraint by adding a new predicate. E.g., the integrity constraint:

$$\mathbf{E}(p(X)) \rightarrow \mathbf{E}(q(Y))$$

can be converted into

$$\mathbf{E}(p(X)) \rightarrow r.$$
$$r \leftarrow \mathbf{E}(q(Y)).$$

Moreover, a variable cannot occur in a Social Integrity Constraint only in negative, definite literals, but it must always appear in literals with predicates **H**, **E**, **NE**.

Definition 2. *A Social Integrity Constraint is* Quantifier Allowed *if any variable occurring in the head in literals of type* **E**, $\neg$**E***, or (in the body) in a negative, defined literal*

- *either does not occur in the body*
- *or it occurs in the body in a literal of type* **H**, **E**, $\neg$**E**.

The society knowledge is quantifier allowed if all the Social Integrity Constraints are quantifier allowed.

Definition 3. *A Social Integrity Constraint is* Constraint Allowed *if*

- *all the variables that are universally quantified with scope the body do not occur in quantifier restrictions;*
- *the other variables (that occur only in the head, or both in the head and in the body) can occur in quantifier restrictions. For each quantifier restriction c occurring in the Social Integrity Constraint,*
  - *either c only involves variables that also occur in* **E**, $\neg$**E**, **H** *atoms*
  - *or it involves* one *variable that also occurs in at least one* **NE** *atom and possibly other variables each of which occurs in* **H** *atoms.*

Definition 4. *A* Clause is *Constraint Allowed if the variables that are universally quantified with scope the body do not occur in quantifier restrictions, and each variable that occurs in a quantifier restriction also occurs in at least one atom* **E** *in the body.*

Definition 5. *A* Society Knowledge is Constraint Allowed *if all its social integrity constraints and all its clauses in the SOKB are Constraint Allowed.*

3 ALP Interpretation of the Society model and declarative semantics

In the following, we recall the society model as Abductive Logic Program, presented in previous deliverables [8, 4]. In particular, we introduce the notion of *instance* of a society as an Abductive Logic Program in order to capture the dynamic aspects of a society.

The (static) model of a society, $\mathcal{S}$, is represented as the following triple:

$$\langle SOKB, \mathcal{E}, \mathcal{IC}_S \rangle$$

where:

- $SOKB$ is the *Social Organization Knowledge Base*,
- $\mathcal{IC}_S$ is the set of *Social Integrity Constraints*, and
- $\mathcal{E}$ is the set of abductive predicates, where the abducible predicates correspond to **E** and **NE** (and their explicit negation $\neg$).

Definition 6. *An* instance $\mathcal{S}_{\mathbf{HAP}}$ *of a society $\mathcal{S}$ is represented as an ALP, i.e., a triple $\langle P, \mathcal{E}, \mathcal{IC}_S \rangle$ where:*

- *P is the $SOKB$ together with the history of happened events* $\mathbf{HAP}$*;*
- *$\mathcal{E}$ is the set of* abducible predicates *of $\mathcal{S}$;*
- *$\mathcal{IC}_S$ are the social integrity constraints of $\mathcal{S}$.*

In this way, our social framework (and its dynamic counterpart, as instance of a society) has been smoothly given an abductive interpretation.

If the society is goal driven, then there exists a goal G at the society level (which is simply *true* if the society is not goal driven).

Definition 7. *Given two instances, $\mathcal{S}_{\mathbf{HAP}}$ and $\mathcal{S}_{\mathbf{HAP}'}$, of a society $\mathcal{S}$, $\mathcal{S}_{\mathbf{HAP}'}$ is a* proper extension *of $\mathcal{S}_{\mathbf{HAP}}$ if and only if $\mathbf{HAP} \subset \mathbf{HAP}'$.*

Definition 8. *Given an instance, $\mathcal{S}_{\mathbf{HAP}}$, of a society $\mathcal{S}$, the instance is* closed *iff it has no proper extensions. We denote a closed instance as $\mathcal{S}_{\overline{\mathbf{HAP}}}$.*

In the following, we indicate a closed history by means of an overline: $\overline{\mathbf{HAP}}$. Notice that in a closed instance, we assume that no further event might occur (i.e., the instance has no further extensions and the history is closed under CWA).

3.1 Declarative semantics

In the following we give semantics to a society instance by identifying sets of expectations which, together with the society's knowledge base and the happened events, imply an instance of the goal - if any - and *satisfy* the integrity constraints.

For notion of integrity constraint satisfaction we rely, in the following, upon a notion of entailment in a three-valued logic, since more general and capable of dealing with both open and closed society instances. Therefore, in the following, the symbol $\models$ has to be interpreted as the notion of entailment in a three-valued setting.

Furthermore, in this section, we consider negative literals of the kind $\neg\mathbf{H}()$ as new positive literals that have no definition in each open society instance. For closed society instances, we use Clark's completion of the history, *Comp*($\mathbf{HAP}$), and negation is interpreted in the Closed World Assumption (CWA).

Throughout this section, for the sake of simplicity, we always consider a ground version of society's knowledge base and integrity constraints, and do not consider CLP-like constraints.

We first recall the concept of *$\mathcal{IC}_S$-consistent set of social expectations.* Intuitively, given a society instance, a $\mathcal{IC}_S$-consistent set of social expectations consists of a set of expectations about social events that are compatible with P (i.e., the $SOKB$ and the set $\mathbf{HAP}$), and with $\mathcal{IC}_S$.

Definition 9. **($\mathcal{IC}_S$-consistency)** *Given a (closed/open) society instance $\mathcal{S}_{\mathbf{HAP}}$, an $\mathcal{IC}_S$-*consistent *set of social expectations Δ is a set of expectations such that:*

$$SOKB \cup \mathbf{HAP} \cup \Delta \models \mathcal{IC}_S \tag{3}$$

In definition 9 (and in the following definitions 12, 13, 14 and 15), for open instances we refer to a three-valued completion where only the history of events has not been completed. Therefore, for open instances,

$$SOKB \cup \mathbf{HAP} \cup \Delta \models \mathcal{IC}_S$$

is a shorthand for:

$$Comp(SOKB \cup \Delta) \cup \mathbf{HAP} \cup CET \models \mathcal{IC}_S$$

where *Comp*() is three-valued completion [7] and *CET* Clark's equational theory.

For closed instances, instead,

$$SOKB \cup \overline{\mathbf{HAP}} \cup \Delta \models \mathcal{IC}_S$$

is a shorthand for:

$$Comp(SOKB \cup \Delta \cup \overline{\mathbf{HAP}}) \cup CET \models \mathcal{IC}_S$$

since also the history of events (closed) needs to be completed.

Among $\mathcal{IC}_S$-consistent sets of expectations, we are interested in those which are also consistent with respect to E-consistency and ¬-consistency.

Definition 10. (E-consistency) *A set of social expectations* Δ *is* E-consistent *if and only if for each (ground) term p:*

$$\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq \Delta$$

Definition 11. (¬-consistency) *A set of social expectations* Δ *is* ¬-consistent *if and only if for each (ground) term p:*

$$\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq \Delta$$

and

$$\{\mathbf{NE}(p), \neg\mathbf{NE}(p)\} \not\subseteq \Delta$$

Given a closed (respectively, open) society instance, a set of expectations is called *closed* (resp. *open*) *admissible* if it satisfies Definitions 9, 10 and 11, i.e. if it is $\mathcal{IC}_S$-, E- and ¬-consistent.

Definition 12. (Fulfillment) *Given a (closed/open) society instance* $\mathcal{S}_{\mathbf{HAP}}$*, a set of social expectations* Δ *is* fulfilled *if and only if for each (ground) term p:*

$$\mathbf{HAP} \cup \Delta \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models \bot \qquad (4)$$

Notice that Definition 12 above requires, for a closed instance of a society, that each positive expectation in Δ has a corresponding happened event in **HAP**, and each negative expectation in Δ has no corresponding happened event. This requirement is weaker for open instances, where a set Δ is not fulfilled only when a negative expectation occurs in the set, but the corresponding event happened (i.e., the implication $\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)$ is false).

Symmetrically, we define a violation:

Definition 13. (Violation) *Given a (closed/open) society instance* $\mathcal{S}_{\mathbf{HAP}}$*, a set of social expectations* **EXP** *is* violated *if and only if there exists a (ground) term p such that:*

$$\mathbf{HAP} \cup \Delta \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \models \bot \qquad (5)$$

Finally, we give, in the following, the notion of goal achievability and achievement.

Definition 14. Goal achievability *Given an open instance of a society, $\mathcal{S}_{\mathbf{HAP}}$, and a ground goal G, we say that G is* achievable *(and we write $\mathcal{S}_{\mathbf{HAP}} \models\!\approx_{\Delta} G$) iff there exists an (open) admissible and fulfilled set of social expectations Δ, such that:*

$$SOKB \cup \mathbf{HAP} \cup \Delta \vDash G \tag{6}$$

(which, as explained earlier, is a shorthand for $Comp(SOKB \cup \Delta) \cup \mathbf{HAP} \cup CET \models G$).

Definition 15. Goal achievement *Given a closed instance of a society, $\mathcal{S}_{\overline{\mathbf{HAP}}}$, and a ground goal G, we say that G is* achieved *(and we write $\mathcal{S}_{\overline{\mathbf{HAP}}} \vDash_{\Delta} G$) iff there exists a (closed) admissible and fulfilled set of social expectations Δ, such that:*

$$SOKB \cup \overline{\mathbf{HAP}} \cup \Delta \vDash G \tag{7}$$

(i.e., $Comp(SOKB \cup \overline{\mathbf{HAP}} \cup \Delta) \cup CET \models G$).

4 The society proof procedure

The proof procedure of the society, called $\mathcal{S}$CIFF (Society C-IFF) has the following features:

- it accepts new events as they happen
- it produces sets of expectations
- it detects fulfillment of expectations
- it detects violations as soon as possible.

In this section we recall the basic concepts of the $\mathcal{S}$CIFF proof procedure. The complete description, with all the transitions, is given in Deliverable D8 [4].

4.1 Data Structures

The $\mathcal{S}$CIFF proof procedure is based on a rewriting system transforming one node to another (or to others). A node can be either the special node *false*, or defined by the following tuple

$$T \equiv \langle R, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL} \rangle$$

where

- R is a conjunction (initially set to the goal G), the conjuncts can be atoms or disjunctions (of conjunctions of atoms)
- CS is the constraint store
- $PSIC$ is the set of partially solved integrity constraints
- **EXP** is the set of (pending) expectations
- **HAP** is the history of happened events
- **FULF** is a set of fulfilled expectations

- **VIOL** is a set of violated expectations

If one of the elements of the tuple is *false*, then the whole tuple is the special node *false*, which cannot have successors.

We have seen that a society instance can be open or closed, depending on whether more events can happen or not, i.e., whether **HAP** is an open set or a closed set. We assume that we can rely on a predicate *closed*/1, which holds true if its argument represents a closed set.

4.1.1 Initial Node and Success

A derivation D is a sequence of nodes

$$T_0 \to T_1 \to \cdots \to T_{n-1} \to T_n.$$

Given a goal G and a set of integrity constraints $\mathcal{IC}_S$, we build the first node in the following way:

$$T_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

i.e., the conjunction R is initially the query ($R_0 = \{G\}$) and the partially solved integrity constraints *PSIC* is the set of integrity constraints ($PSIC_0 = \mathcal{IC}_S$).

The other nodes $T_j, j > 0$, are obtained by applying the transitions that we will define in the next section, until no further transition can be applied (we call this last condition *quiescence*).

Every arc in a derivation is labelled with the name of a transition.

Let us now give the definition of successful derivation, both in the case of an open society instance (where new events may be added to the history) and of a closed society instance.

Definition 16. *Starting with an open society instance* $\mathcal{S}_{\mathbf{HAP}^i}$ *there exists an* open successful derivation *for a goal G iff the proof tree with root node* $\langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \mathbf{HAP}^i, \emptyset, \emptyset \rangle$ *has at least one leaf node*

$$\langle \emptyset, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}^f, \mathbf{FULF}, \emptyset \rangle$$

where $\mathbf{HAP}^f \supseteq \mathbf{HAP}^i$ *and CS is consistent (i.e., there exists a ground variable assignment such that all the constraints are satisfied). In that case, we write:*

$$\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim^{\mathbf{HAP}^f}_{\mathbf{EXP} \cup \mathbf{FULF}} G$$

Definition 17. *Starting with a society instance* $\mathcal{S}_{\mathbf{HAP}^i}$ *there exists a* closed successful derivation *for a goal G iff the proof tree with root node* $\langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \mathbf{HAP}^i, \emptyset, \emptyset \rangle$ *has at least one leaf node*

$$\langle \emptyset, CS, PSIC, \mathbf{EXP}, \overline{\mathbf{HAP}^f}, \mathbf{FULF}, \emptyset \rangle$$

where $\overline{\mathbf{HAP}^f} \supseteq \mathbf{HAP}^i$*, CS is consistent, and* **EXP** *contains only negative literals* $\neg\mathbf{E}$ *and* $\neg\mathbf{NE}$*. In such a case, we write:*

$$\mathcal{S}_{\mathbf{HAP}^i} \vdash^{\overline{\mathbf{HAP}^f}}_{\mathbf{EXP} \cup \mathbf{FULF}} G.$$

From a non-failure leaf node N, answers can be extracted in a very similar way to the IFF proof procedure. Answers of the $\mathcal{S}$CIFF proof procedure are called *expectation answers.* To compute an expectation answer, first, a substitution σ' is computed such that

- σ' replaces all variables in N that are not universally quantified by a ground term

- σ' satisfies all the constraints in the store CS_N.

If the constraint solver is (theory) complete [6] (i.e., for each set of constraints c, the solver always returns *true* or *false*, and never *unknown*), then there will always exist a substitution σ' for each non-failure leaf node N. Otherwise, if the solver is incomplete, σ' may not exist. The non-existence of σ' is discovered during the answer extraction phase. In such a case, the node N will be marked as a failure node, and another leaf node can be selected (if it exists).

Definition 18. *Let $\sigma = \sigma'|_{vars(G)}$ be the restriction of σ' to the variables occurring in the initial goal G. Let $\Delta = (\mathbf{FULF}_N \cup \mathbf{EXP}_N)\sigma'$. The pair (Δ, σ) is the* expectation answer *obtained from the node N.*

Part I
Soundness

5 Introduction

The soundness of the $\mathcal{S}$CIFF proof-procedure was stated and proven in some limited cases in Deliverable D8 [4]. The proof was then extended to more general cases in a document [3] annexed to Deliverable D12.

6 Soundness Properties of $\mathcal{S}$CIFF Proof Procedure

We state the desirable properties of soundness and completeness for the $\mathcal{S}$CIFF proof procedure (Section 4) with respect to the declarative semantics (Section 3.1) by considering, in the following, a given society instance $\mathcal{S}_{\mathbf{HAP}}$ and a goal G for it. Depending on the openness or closure of the society instance, we state in the following the desirable properties of correctness for the proof procedure. For the sake of simplicity we do not consider CLP constraints in the program, except for equality and disequality, that are dealt with the rules given in Deliverable D8 [4].

The following theorem relates the operational notion of open successful derivation with the corresponding declarative notion of goal achievability.

Theorem 1. Open Soundness. *Given an open society instance $\mathcal{S}_{\mathbf{HAP}^i}$, if*

$$\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim^{\mathbf{HAP}^f}_{\mathbf{EXP}\cup\mathbf{FULF}} G$$

with expectation answer (Definition 18) $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$ then

$$\mathcal{S}_{\mathbf{HAP}^f} \models\!\!\approx_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

The theorem above states that if there exists an open successful derivation for a goal G starting from an initial history $\mathbf{HAP}^i$ and leading to the (open) society instance $\mathcal{S}_{\mathbf{HAP}^f}$ with abduced expectation set $\mathbf{EXP} \cup \mathbf{FULF}$, and with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$, then $G\sigma$ is achievable in $\mathcal{S}_{\mathbf{HAP}^f}$ (with the expectation set $(\mathbf{EXP} \cup \mathbf{FULF})\sigma$).

In the closed case, the soundness property is stated as follows, relating the operational notion of closed successful derivation with the corresponding declarative notion of goal achievement.

Theorem 2. Closed Soundness. *Given a closed society instance* $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$, *if*

$$\mathcal{S}_{\mathbf{HAP}^i} \vdash^{\overline{\mathbf{HAP}^f}}_{\mathbf{EXP} \cup \mathbf{FULF}} G$$

with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$ *then*

$$\mathcal{S}_{\overline{\mathbf{HAP}^f}} \models_{(\mathbf{EXP} \cup \mathbf{FULF})\sigma} G\sigma$$

Soundness in the closed case states that if there exists a closed successful derivation for a goal G starting from an initial history $\mathbf{HAP}^i$ and leading to the (closed) society instance $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$ with abduced expectation set $\mathbf{EXP} \cup \mathbf{FULF}$, and with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$, then $G\sigma$ is achieved in $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$ (with the expectation set $(\mathbf{EXP} \cup \mathbf{FULF})\sigma$).

In Section 6.1, we introduce some lemmas useful to prove the property of open and closed soundness. These Lemmas allow us to establish a corresponding $\mathcal{S}$CIFF computation where all the incoming events are considered at the beginning of the computation, instead of interleaving Happening transitions with the other ones.

We first prove soundness property for the open case (Proposition 3) in Section 6.3, and for the closed case (Proposition 4) in Section 6.4, both in the case in which the final node does not contain universally quantified abducibles. Then we extend both proofs to the general case, i.e., the case in which the final node can contain universally quantified abducibles.

6.1 Lemmas

In this section, we prove some lemmas that will be useful in the following proofs. These Lemmas allow us to establish a corresponding $\mathcal{S}$CIFF computation where all the incoming events are considered at the beginning of the computation, instead of interleaving Happening transitions with the other ones. These results are represented by Lemma 7 for the open case and Lemma 8 for the closed case.

A further useful results proved in this section is Lemmas 4, that will prove that if in a derivation we have a node containing an abduced atom with universally quantified variables, then there will be a universally quantified variable in every non-failure successor node. Thanks to this lemma, we will be able to use results from the IFF proof procedure (in which universally quantified abducibles cannot occur *in any node* of a derivation) in $\mathcal{S}$CIFF derivations that do not *terminate* in a node with universally quantified abducibles.

We first give some intermediate results; first of all, we relate the treatment of disequality in $\mathcal{S}$CIFF and in IFF proof procedures.

Lemma 1. *The $\mathcal{S}$CIFF proof procedure deals with disequalities in the Constraint Store [4]. The IFF proof procedure transforms a disequality $A \neq B$ into an implication $A = B \rightarrow false$.*

For each of the rules for disequality in $\mathcal{S}$CIFF that does not involve quantifier restrictions, there is one or more rules in IFF that lead to the same node.

Proof. Let us consider a disequality $A \neq B$ in both proof procedures. Let us assume that one of the rules for disequality is used in $\mathcal{S}$CIFF; we prove that there are one or more IFF rules applicable that lead to the same result.

1. Replace $f(t_1, \dots, t_j) \neq f(s_1, \dots, s_j)$ with $t_1 \neq s_1 \vee \dots \vee t_j \neq s_j$.

   In the IFF, we could

   - rewrite $f(t_1, \dots, t_j) \neq f(s_1, \dots, s_j)$ as $f(t_1, \dots, t_j) = f(s_1, \dots, s_j) \rightarrow false$.
   - Apply the Rules for Equality obtaining $t_1 = s_1 \wedge \dots \wedge t_j = s_j \rightarrow false$.
   - Apply j times Case Analysis; in the first application we get $t_1 = s_1 \wedge (t_2 = s_2 \wedge \dots \wedge t_j = s_j \rightarrow false) \vee t_1 \neq s_1$. By iteratively applying Case Analysis to the implications we will get $t_1 \neq s_1 \vee \dots \vee t_j \neq s_j \vee [t_1 = s_1 \wedge \dots \wedge t_j = s_j \wedge (true \rightarrow false)]$
   - by applying Logical Equivalences, we have $t_1 \neq s_1 \vee \dots \vee t_j \neq s_j$.

2. Replace $f(t_1, \dots, t_j) \neq g(s_1, \dots, s_l)$ with $true$ whenever f and g are distinct or $j \neq l$.

   In the IFF, we can

   - Rewrite $f(t_1, \dots, t_j) \neq g(s_1, \dots, s_l)$ as $f(t_1, \dots, t_j) = g(s_1, \dots, s_l) \rightarrow false$.
   - Apply Rewriting Rules for Equality and get $true \rightarrow false$.
   - Apply logical equivalence and get $true$.

The same reasoning can be applied for the other rules 3, 4, 5 and 6a of $\mathcal{S}$CIFF proof procedure [4]. Rules 6b and 6c involve quantifier restrictions. □

Lemma 2. *Applying the rules for disequality [4] cannot change the quantification of a universally quantified variable. Moreover, after applying rules for disequality, the disequality constraints are not imposed on universally quantified variables.*

Proof. Trivial, considering the rules of disequality: either they fail, or they succeed without creating new constraints, or they impose constraints only on existentially quantified variables. □

Lemma 3. *Let us suppose that the constraint solver only contains the rules for equality and disequality [4] (otherwise, the behavior of the proof depends also on the type of constraint solver). Let us suppose that the rules for equality and disequality are applied before the other transitions.*

If an atom A is abduced containing a universally quantified variable $\hat{X}$, and $\hat{X}$ only occurs in abduced atoms, if Propagation is not applied, then the atom A will remain in the set of abduced atoms and variable $\hat{X}$ will remain universally quantified in any success nodes.

Proof. Since variable $\hat{X}$ only occurs in abduced atoms, its state can be changed only by transitions that affect abduced atoms. Let us consider the single transitions:

Unfolding does not affect an abduced atom; it may unify the variables appearing in a goal or in the body of an implication with the head of a clause. However, since variable $\hat{X}$ does not occur in non abducible atoms, it will not be affected.

Abduction does not affect atoms already abduced.

Splitting does not affect abduced atoms.

Case Analysis affects the variables appearing in an implication. Since variable $\hat{X}$ only occurs in abduced atoms, it will not be affected by case analysis.

Factoring is not applicable to universally quantified atoms.

Equivalence Rewriting rules apply only to equalities, thus they will not affect variables that do not occur in an equality.

Logical Equivalence The only rule that can change a (positive) atom is $A \vee true \leftrightarrow true$. This rule can only be applied to a disjunction, but, since atom A has been abduced, it cannot be argument of a disjunction (in fact, disjunctions cannot occur in **EXP**, **FULF**, and **VIOL**, but only in the Constraint Store or in R).

Happening does not change abduced atoms.

non-Happening does not change abduced atoms.

Closure only changes the history and does not change abduced atoms.

Violation NE generates two nodes. One is a violation node. The other imposes a disequality constraint. We know that a disequality constraint cannot bind a universally quantified variable (Lemma 2). Since we chose a preferred order of application of transitions (namely, we apply the rules for equality and disequality before the other transitions), we can ensure that Violation **NE** will not bind any universally quantified variable in non failure nodes.

Fulfillment E, Violation E deal with **E** atoms, that cannot contain universally quantified variables.

Fulfillment NE does not change atoms, only moves them from **EXP** to **FULF**.

Constraint Solving We do not deal with constraints (except for disequality, for which we know that it does not bind universally quantified variables, and equality, already considered in Equivalence Rewriting rules).

□

Lemma 4. *If, in a node N, an atom A is abduced containing a universally quantified variable $\hat{X}$, then in any node which is a descendant of N there will be such atom A in the set of abduced atoms with universally quantified variable $\hat{X}$ (unless the node is* false*).*

Proof. We prove the lemma in the following steps:

1. whenever an atom is abduced with a new, universally quantified variable $\hat{X}$, then $\hat{X}$ cannot occur elsewhere except for abducible atoms.
2. Lemma 3 is applicable, thus the thesis holds if *Propagation* is not applied.
3. Applying *Propagation* creates new universally quantified variables, while the abduced atoms are not touched.

We now elaborate on steps 1 and 3.

1. If the proof procedure abduces an atom with universally quantified variables, then the universally quantified variables can occur only in abducibles.

In fact, the *Abduction* transition can be applied only to atoms in the set R. R may contain a universally quantified atom because it was in the initial goal of the society. In this case, the variable cannot occur in other atoms (except abducibles and constraints, see Section 2.1). We will suppose that the goal does not contain equality constraints; this is not a limitation. A universally quantified atom may be inserted in R by the following transitions:

Unfolding. In this case the universally quantified atom was in the body of a clause; the syntax [8] imposes that the universally quantified variable does not appear elsewhere (except for constraints and other abducibles). Again, we suppose that the body does not contain equality constraints (this is not a limitation).

Logical Equivalence: $(true \rightarrow A) \leftrightarrow A$. In this case, the body of an implication has become true. The implication was written with the syntax given in [8]. In particular, since the universally quantified variable $\hat{X}$ is new, then it occurred only in **NE** atoms and constraints. We suppose that the conclusion of the implication does not contain equality constraints (this is not restrictive); for disequality constraints we rely on Lemma 2.

3. We still have to show that adding the *Propagation* transition does not undermine the thesis. The *Propagation* transition performs a copy of an atom and of an IC, then it operates only on the copy of the two. The universally quantified variables are renamed by the copy, thus any subsequent operation on the copied universally quantified variables will not affect the universally quantified variables occurring in the abduced atom. □

The IFF proof procedure deals with a static theory. $\mathcal{S}$CIFF deals with a dynamic theory to which new happened events may be added during a derivation. So to show a mapping between $\mathcal{S}$CIFF and IFF derivations, we need to address the following question:

> Does the success nodes in $\mathcal{S}$CIFF depend on the events arrival rate? An open successful derivation may disappear, if a new event E happens. Would we have the same success nodes if we had known event E in advance?

Lemmas 7 and 8 will try to answer these questions. We first need some intermediate results; we are going to prove that if a transition Tr is applicable to some elements[1] of a node N_k, and leads to a node N_{k+1}, then it can be applied to the same elements of an identical node but with a larger history, and will lead to a node identical to N_{k+1} but with a larger history, as informally suggested by the following scheme:

$$\begin{array}{ccc} N_k & \xrightarrow{Tr} & N_{k+1} \\ & \Downarrow & \\ N_k \cup \{\mathbf{H}(E)\} & \xrightarrow{Tr} & N_{k+1} \cup \{\mathbf{H}(E)\} \end{array}$$

Lemma 5. *If a transition (except for Happening, Non-happening and Closure) is applicable to some elements of a non-closed node (i.e., a node with an open history)*

$$N_k \equiv \langle R_k, CS_k, PSIC_k, \mathbf{EXP}_k, \mathbf{HAP}_k, \mathbf{FULF}_k, \mathbf{VIOL}_k \rangle$$

and it produces a new node

$$N_{k+1} \equiv \langle R_{k+1}, CS_{k+1}, PSIC_{k+1}, \mathbf{EXP}_{k+1}, \mathbf{HAP}_{k+1}, \mathbf{FULF}_{k+1}, \mathbf{VIOL}_{k+1} \rangle$$

then the same transition is also applicable to the same elements in the (non-closed) node

$$N'_k \equiv \langle R_k, CS_k, PSIC_k, \mathbf{EXP}_k, \mathbf{HAP}_k \cup \{\mathbf{H}(E)\}, \mathbf{FULF}_k, \mathbf{VIOL}_k \rangle$$

[1]Recall that transitions are applicable to elements of the nodes; for example, the transition *Violation* **NE** is applied to a happened event and an abduced **NE** atom in a node.

where E is an event, and it produces a new node

$$N'_{k+1} \equiv \langle R_{k+1}, CS_{k+1}, PSIC_{k+1}, \mathbf{EXP}_{k+1}, \mathbf{HAP}_{k+1} \cup \{\mathbf{H}(E)\}, \mathbf{FULF}_{k+1}, \mathbf{VIOL}_{k+1}\rangle.$$

Proof. Let us consider the single transitions.

Unfolding is applicable when a literal in R or in the body of an IC matches with the head of one or more rules in the SOKB. It is not affected by the history.

Abduction is applicable when an abducible atom is in R. Its applicability does not directly depend on the history, nor its results.

Propagation is applicable when an atom in the body of an IC matches an atom A (that can either be in the history or in the abduced atoms). If the history is enlarged, the atom A is still its member, thus Propagation can be applied in the same way.

Splitting does not depend on the history.

Case Analysis does not depend on the history.

Factoring is not affected.

Equivalence Rewriting rules the history cannot contain equalities.

Logical Equivalence The applicability of these rules depend only on the presence, in the tuple, of implications, conjunctions, and disjunctions; not on the elements in the history.

Violation NE considers an atom $\mathbf{H}(A) \in \mathbf{HAP}_k$ and a $\mathbf{NE}(B) \in \mathbf{EXP}_k$. Thus, if *violation* **NE** is applicable in N_k, then $\exists \mathbf{H}(A) \in \mathbf{HAP}_k$ that has been used in the transition. But $\mathbf{H}(A) \in \mathbf{HAP}_k \cup \{\mathbf{H}(E)\}$. So the same *violation* **NE** transition is applicable in N'_k and the result will be the same.

Fulfillment E is true for the same reasons as transition Violation **NE**.

Violation E is true for the same reasons as transition Violation **NE** if we make the hypothesis of full temporal knowledge [4], and is not applicable otherwise.

Fulfillment NE is not applicable if the history is open.

Consistency does not depend on the history.

Constraint Solving does not depend on the history.

□

In the previous lemma we excluded transitions *Happening, Closure*, and *Non-happening.* We now extend the same result given in the previous lemma to the *Happening* transition.

Lemma 6. *Consider two nodes N_k and N'_k, which are identical except for the history:* $\mathbf{HAP}_k \cup \{\mathbf{H}(E^1)\} = \mathbf{HAP}'_k$*. If transition Happening of an event E is applicable to the node N_k, leading to a history* $\mathbf{HAP}_{k+1}$*, then*

- *either Happening of E is not applicable to N'_k because E was already in its history (i.e., $E = E^1$)*

- *or Happening of E is applicable to N'_k but it fails (and in this case the history $\mathbf{HAP}'_k$ is closed)*
- *or the transition Happening of the event E is applicable to the node N'_k, and in the obtained node $\mathbf{HAP}'_{k+1} = \mathbf{HAP}'_k \cup \{\mathbf{H}(E^1)\}$*

Proof. Trivial, from the definition of transition Happening. □

Now we know that, given a derivation containing a sequence of transitions

$$\ldots \longrightarrow N_k \xrightarrow{Tr} N_{k+1} \xrightarrow{Happening} N_{k+2} \longrightarrow \ldots$$

where Tr is one of the transitions of the proof procedure, except for non-Happening and Closure, we can safely exchange the two transitions:

$$\ldots \longrightarrow N_k \xrightarrow{Happening} N'_{k+1} \xrightarrow{Tr} N_{k+2} \longrightarrow \ldots$$

and obtain an analogous derivation (in fact, from node N_{k+2} the two derivations are the same).

Lemma 7. *Consider an open successful derivation D*

$$N_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_{n-1} \rightarrow N_n$$

with $\mathbf{HAP}_n$ the history in the node N_n and

$$N_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \emptyset, \emptyset, \emptyset \rangle.$$

In this case, there exists an open successful derivation D'

$$N'_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_{m-1} \rightarrow N_m$$

with

$$N'_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \mathbf{HAP}_n, \emptyset, \emptyset \rangle$$

in which the final node $N_m \equiv N_n$.

Proof. Let N_h be the first node in D to which transition *happening* was applied. Suppose that *happening* inserts the event $\mathbf{H}(E_\alpha)$ in the history. Since D is an open successful derivation, there is no transition in D of type *closure*; moreover in all the nodes in D the history is open. Thus, we can apply Lemma 5 to the transition $Tr(N_{h-1})$ (i.e., the transition that was applied to the node N_{h-1}) and get an equivalent derivation D' in which Tr and *happening* are exchanged. Again, we can apply the same method to the node N_{h-2} and so on, until the transition *happening* becomes the first; call D_α the derivation obtained in this way. Of course, D_α terminates in the same node as a derivation D'_α that starts from a history $\mathbf{H}_1 = \{\mathbf{H}(E_\alpha)\}$.

By repeatedly applying the same method for all the *happening* transitions in D, we obtain an equivalent derivation that terminates in the same node. Since no transition was applicable in the final node in D, no transition is applicable in the final node of D'. □

We can prove a similar result for closed successful derivations:

Lemma 8. *Consider a closed successful derivation D*

$$N_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_{n-1} \rightarrow N_n$$

with $\overline{\mathbf{HAP}}_n$ the history in the node N_n and

$$N_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \emptyset, \emptyset, \emptyset \rangle.$$

In this case, there exists a closed successful derivation D'

$$N'_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_{m-1} \rightarrow N_m$$

with

$$N'_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \overline{\mathbf{HAP}}_n, \emptyset, \emptyset \rangle$$

in which the final node $N_m \equiv N_n$.

Proof. D is a closed successful derivation, thus there exists exactly one transition of type *closure* in D; call N_c the first node with closed history ($\overline{\mathbf{HAP}_c}$). Let us consider the derivation $D_o \subset D$ that starts from the same initial node N_0 up to the node N_{c-1}.

$$\underbrace{\underbrace{N_0 \longrightarrow N_1 \longrightarrow \ldots \longrightarrow N_{c-1}}_{D_o} \stackrel{closure}{\longrightarrow} N_c \longrightarrow N_{c+1} \longrightarrow \ldots \longrightarrow N_n}_{D}$$

Derivation D_o does not contain closed nodes, thus we can apply the same proof as in Lemma 7.

On the rest of the derivation, from N_c to N_n, the history is closed, thus transition *happening* would give a failure. Since D is a successful closed derivation, there is no *happening* transition from N_c to N_n. Thus the lemma holds for the whole derivation D. $\square$

6.2 IFF-like Rewritten Program

The proof of correctness (soundness, in particular) will be given by exploiting soundness results of the IFF proof procedure with respect to three-valued completion semantics. To this end, here we map $\mathcal{S}$CIFF programs into IFF-like (rewritten) programs, and then prove (in Sections 6.3 and 6.4) that open/closed $\mathcal{S}$CIFF successful derivations in which no literal is abduced with universally quantified variables have a counterpart in IFF derivations.

We first define the rewritten program, which is a translation in IFF syntax of the society's knowledge base. Since we know that no literal is abduced with universally quantified variables, we can replace universally quantified variables with constants.

The allowedness condition for integrity constraints of the IFF proof procedure requires that every variable in the conclusion occurs in the condition. This cannot be the case in our social integrity constraints. However, as discussed in Section 2.2, we can transform our $\mathcal{IC}_S$ into a new set of integrity constraints satisfying the allowedness condition. We give a simple example in the following. An integrity constraint of kind

$$H(p(X)) \rightarrow E(q(Z))$$

is not allowed since a new variable (Z) occurs in the conclusion. But, it can be transformed into the (IFF-like) integrity constraint:

$$H(p(X)) \rightarrow a$$

and the definition:

$$a \leftarrow E(q(Z))$$

which are both allowed.

Definition 19. *Given an instance of a society knowledge base $\langle SOKB \cup \mathbf{HAP}, \mathcal{E}, \mathcal{IC}_S \rangle$, we define the* IFF rewritten program $\langle SOKB^* \cup \mathbf{HAP}, \mathcal{E}, \mathcal{IC}_S^* \rangle$ *as follows:*

- *For each $ic_S \in \mathcal{IC}_S$ that does not satisfy the allowedness condition of the IFF proof procedure, we rewrite it as explained earlier.*
- *For each $ic_S \in \mathcal{IC}_S$ with a universally quantified variable X occurring in the head of a social integrity constraint but not in the body, X is replaced in the corresponding ic_S^* in $\mathcal{IC}_S^*$ with a constant symbol not occurring elsewhere.*
- *In the same way, for each clause in $SOKB$ with a variable X which is universally quantified in the* Body *of the clause, X is replaced in $SOKB^*$ with a new constant symbol.*
- *In the same way, for each atom in the goal G with a variable X which is universally quantified, X is replaced in G^* with a new constant symbol.*
- *All $\neg\mathbf{H}$ atoms are considered as a new predicate without definition (i.e., always false).* **H** *events in the history are considered as a predicate in the $SOKB^*$.*
- *We complete the $SOKB$ with the Clark's completion to obtain $SOKB^*$.*

Notice that, by construction, given a set of abduced atoms Δ (not containing universally quantified atoms), and an open history for the society, the set of atoms that are true in the rewritten program and in the original society instance are the same.

Lemma 9. *For every finite ground set $\Delta \subseteq \mathcal{E}$ (non containing universally quantified variables and) non containing the new constant symbols introduced in $SOKB^*$,*

$$SOKB^* \cup \mathbf{HAP} \cup \Delta \models a \Leftrightarrow SOKB \cup \mathbf{HAP} \cup \Delta \models a$$

and

$$SOKB^* \cup \mathbf{HAP} \cup \Delta \models \mathcal{IC}_S^* \Leftrightarrow SOKB \cup \mathbf{HAP} \cup \Delta \models \mathcal{IC}_S$$

where the symbol $\models$ stands for the three-valued completion semantics.

Proof. The syntax imposes that only abducible atoms can be universally quantified in the *Body* of a clause. Thus, the body of a clause

$$a \leftarrow [\forall_X p(X)]$$

(where p is a predicate symbol, in our case, only **NE** and $\neg$**NE**), is true if and only if there exists an atom $\forall_Y p(Y) \in \Delta$, or if for every possible ground atom A with functor p, $A \in \Delta$, which is not, because Δ is finite and ground. Thus, the body of any clause containing universally quantified variables is false in $SOKB$.

The corresponding rewritten clause is

$$a \leftarrow p(c)$$

where c is a new constant symbol. The body of this clause can be true only if Δ contains $p(c)$ or $p(X)$ for some variable X, which is not.

The proof is similar for the universally quantified atoms occurring in the goal, or in the $\mathcal{IC}_S$.

Moreover, atoms $\neg\mathbf{H}$ are all *false* in the rewritten program. In the original society instance, since it is open, atoms $\neg\mathbf{H}$ are new positive literals without definition (see Section 3.1), so they are false as well.

□

6.3 Proof of open soundness: Case without universally quantified abducibles

We consider the case of a (possibly non-ground) goal, expectation sets without universally quantified variables, and do not consider CLP constraints in the program.

By the following Lemma 10, we prove that for this class of programs, any $\mathcal{S}$CIFF open successful derivation has a counterpart in an IFF derivation computed on the IFF-like rewritten program.

Lemma 10. *Let $\mathcal{S}_{\mathbf{HAP}^i}$ be $\langle SOKB, \mathcal{E}, \mathcal{IC}_S\rangle$. Let (Δ, σ) be the answer extracted from an open successful derivation ($\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim_{\Delta}^{\mathbf{HAP}^f} G$) for an initial goal G and an initial society instance $\mathcal{S}_{\mathbf{HAP}^i}$ evolving to a proper extension (see Definition 7) $\mathcal{S}_{\mathbf{HAP}^f}$ such that Δ does not contain universally quantified variables.*

Then (Δ, σ) is an IFF computed answer for G for the program $\langle SOKB^* \cup \mathbf{HAP}^f, \mathcal{E}, \mathcal{IC}_S^*\rangle$.

Proof. We construct a successful IFF derivation from the given successful (open) $\mathcal{S}$CIFF derivation, by mapping every step except non-happening, fulfillment, happening, closure, violation, and propagation onto itself. Propagation is slightly different in the IFF and in the $\mathcal{S}$CIFF proof procedures: in the $\mathcal{S}$CIFF it also performs a copy of the abducible. Let us consider the new transitions, namely non-happening, fulfillment, happening, closure, violation, and propagation.

1. Non-happening transition cannot occur along an open successful derivation (by definition of Non-happening);

2. Violation generates two nodes. The former leading to failure (and therefore not present along a successful open derivation) the latter reproducing the parent node plus a new inequality constraint. Therefore this transition possibly reduces the set of computed substitutions in $\mathcal{S}$CIFF compared to the IFF proof procedure.

3. Happening transition can be removed from the computation thanks to Lemma 7 by considering the equivalent open successful derivation in $\mathcal{S}$CIFF starting from $\mathbf{HAP}^f$ and leading to the same final node.

4. Closure transition generates two nodes, the former identical to its parent, the latter identical to its parent except for the history which is closed. Therefore the latter node cannot occur along an open successful derivation.

5. Fulfillment. Since the derivation is open, fulfillment can be applied only to positive expectations and generates two nodes where $\mathbf{EXP} \cup \mathbf{FULF}$ is identical to the parent node, plus, respectively, a new equality or inequality constraint. Therefore this transition does not change the set of computed substitutions with respect to the IFF proof procedure.

6. Propagation. The only difference between propagation in $\mathcal{S}$CIFF and in C-IFF is the copy: in the IFF proof procedure *Propagation* is applied to an atom and an implication. In the $\mathcal{S}$CIFF proof procedure, first a copy of the atom is performed. The only difference stands in the case of universally quantified variables in abduced atoms (in fact, copy does not perform anything significant if the atom does not contain universally quantified variables). Since we assume that there are no universally quantified atoms in the final Δ, from Lemma 4 we know that no literal has been abduced with universally quantified variables in the derivation. Therefore, copy has no effect on the derivation in this case.

□

We can now prove Open Soundness (stated in Proposition 3) in the case without universally quantified abducibles

Proposition 6.1. Open Soundness: *case without universally quantified abducibles.*
Let $\mathcal{S}_{\mathbf{HAP}^i}$ be an open society instance such that

$$\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim^{\mathbf{HAP}^f}_{\mathbf{EXP}\cup\mathbf{FULF}} G.$$

Let $(\mathbf{EXP}\cup\mathbf{FULF}, \sigma)$ *be the corresponding expectation answer (Definition 18) such that* $\mathbf{EXP}$ *and* $\mathbf{FULF}$ *do not contain universally quantified variables. Then*

$$\mathcal{S}_{\mathbf{HAP}^f} \mid\!\approx_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

i.e., given Definition 14, $Comp(SOKB \cup \Delta\sigma) \cup \mathbf{HAP}^f \cup CET \models G\sigma$.

Proof. Let us consider the proof for an atomic goal (the extension to other structures of the formula G is trivial). Let us suppose that:

$$\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim^{\mathbf{HAP}^f}_{\mathbf{EXP}\cup\mathbf{FULF}} G$$

with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$, and prove that:

$$\mathcal{S}_{\mathbf{HAP}^f} \mid\!\approx_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

Proving this latter condition corresponds to proving the following ones, separately:

(i) $SOKB \cup \mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \models G\sigma$;

(ii) $SOKB \cup \mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \models \mathcal{IC}_S$;

(iii) $\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ ($\neg$-consistency for $\mathbf{E}$ atoms);

(iv) $\{\mathbf{NE}(p), \neg\mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ ($\neg$-consistency for $\mathbf{NE}$ atoms);

(v) $\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ ($\mathbf{E}$-consistency);

(vi) $\mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models \bot$ (fulfillment).

For the case of an open society, we rely upon the three-valued completion [7] of $SOKB$ and expectation sets (i.e., the set $\mathbf{HAP}$ is not completed, since the society instance is open with respect to the happening of events).

Thanks to Lemma 10, conditions (i) and (ii) hold on the basis of the soundness results of IFF [2] for the rewritten program; i.e., $SOKB^* \cup \mathbf{FULF} \cup \mathbf{EXP} \models G^*$ and $SOKB^* \cup \mathbf{FULF} \cup \mathbf{EXP} \models \mathcal{IC}_S^*$. Since the declarative reading of the rewritten program is the same, in this case, as the society instance (Lemma 9), conditions (i) and (ii) hold.

Notice that soundness of IFF is given with respect to a (three-valued) completion semantics of the theory adopted by the proof procedure. This is not the case in our $\mathcal{S}$CIFF open derivation, since history $\mathbf{HAP}$ has not been completed. But since negative literals of kind $\neg\mathbf{H}()$ are viewed as new positive predicates, they are never propagated as in the IFF corresponding derivation.

Let us consider the other conditions.

Conditions (iii), (iv) and (v) hold thanks to the enforcing of E-consistency and ¬-consistency, that generate at most two nodes. In particular, conditions (iii) and (iv) are necessary because we deal with negation of abducible atoms differently from the IFF proof procedure: recall that $\neg\mathbf{E}$ and $\neg\mathbf{NE}$ are considered as new positive atoms.

By contradiction, let us assume that $\mathbf{E}(p)$ and $\neg\mathbf{E}(p)$ belong to Δ (i.e., Δ is not ¬-consistent). In this case however, the requirement of ¬-consistency would lead to failure (and therefore that $\mathbf{E}(p)$ and $\neg\mathbf{E}(p)$ would not be present at the same time into a node along a successful open derivation). Analogously for $\mathbf{NE}(p)$ and $\neg\mathbf{NE}(p)$, and $\mathbf{E}(p)$ and $\mathbf{NE}(p)$ ($\mathbf{E}$-consistency).

Condition (vi) (fulfillment) holds thank to transitions Fulfillment and Violation. By contradiction, let us assume that condition (vi) does not hold. In a three-valued setting this can happen only if there exists an expectation $\mathbf{NE}(p)$ and the corresponding event $\mathbf{H}(p)$. In this case however, transition violation $\mathbf{NE}$ would apply leading to a node not along a successful open derivation.

□

6.4 Proof of closed soundness: case without universally quantified abducibles

As in the open case, we consider the case of a (possibly non-ground) goal, expectation sets without universally quantified variables, and do not consider CLP constraints in the program.

By the following Lemma 11, we prove that for this class of programs, any $\mathcal{S}$CIFF closed successful derivation has a counterpart in an IFF derivation computed on the IFF-like rewritten program.

Lemma 11. *Let $\mathcal{S}_{\mathbf{HAP}^i}$ be $\langle SOKB, \mathcal{E}, \mathcal{IC}_S\rangle$ where $\mathcal{IC}_S$ does not contain $\neg\mathbf{H}$ literals. Let (Δ, σ) be the answer extracted from a closed successful derivation ($\mathcal{S}_{\mathbf{HAP}^i} \vdash_{\Delta}^{\overline{\mathbf{HAP}^f}} G$) for an initial goal G and an initial society instance $\mathcal{S}_{\mathbf{HAP}^i}$ evolving to a proper extension $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$ such that Δ does not contain universally quantified variables.*

Then (Δ, σ) is an IFF computed answer for G for the program $\langle SOKB^* \cup \overline{\mathbf{HAP}^f}, \mathcal{E}, \mathcal{IC}_S^*\rangle$.

Proof. We construct a successful IFF derivation from the given successful (closed) $\mathcal{S}$CIFF derivation, by mapping every step except non-happening, fulfillment, happening, closure, violation, and propagation onto itself. Propagation is slightly different in the IFF and in the $\mathcal{S}$CIFF proof procedures: in the $\mathcal{S}$CIFF it also performs a copy of the abducible.

Let us consider the new transitions, namely violation, happening, closure, fulfillment, and propagation.

1. Violation generates two nodes. The former leading to failure (and therefore not present along a successful open derivation) the latter reproducing the parent node plus a new inequality constraint. Therefore this transition possibly reduces the set of computed substitutions in $\mathcal{S}$CIFF compared to the IFF proof procedure.
2. Happening and Closure transitions can be removed from the computation thanks to Lemma 8 by considering the equivalent open successful derivation in $\mathcal{S}$CIFF starting from $\overline{\mathbf{HAP}^f}$ and leading to the same final node.
3. Fulfillment. In the closed case, fulfillment can be applied both to positive and negative expectations. It generates two nodes where $\mathbf{EXP}\cup\mathbf{FULF}$ is identical to their parent node, plus, respectively, a new equality or inequality constraint. Therefore this transition does not change the set of computed substitutions with respect to the IFF proof procedure.
4. Propagation. Same discussion as for the open case (Lemma 10).

□

As in the open case, we first prove the closed soundness (stated as Proposition 4) in the case without universally quantified abducibles:

Proposition 6.2. Closed Soundness: *Case without universally quantified variables in abduced atoms.*
Given a closed society instance $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$*, if*

$$\mathcal{S}_{\mathbf{HAP}^i} \vdash^{\overline{\mathbf{HAP}^f}}_{\mathbf{EXP}\cup\mathbf{FULF}} G$$

with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$*, where* $\mathbf{EXP}$ *and* $\mathbf{FULF}$ *do not contain variables universally quantified, then*

$$\mathcal{S}_{\overline{\mathbf{HAP}^f}} \models_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

Proof. For the case of a closed society instance, we rely upon the 3-valued completion [7] of $SOKB$, expectation sets and the set $\mathbf{HAP}$ too, since the society instance is now closed with respect to the happening of events.

Let us consider the proof for an atomic goal (the extension to other structures of the formula G is trivial). Let us suppose that:

$$\mathcal{S}_{\mathbf{HAP}^i} \vdash^{\mathbf{HAP}^f}_{\mathbf{EXP}\cup\mathbf{FULF}} G$$

with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$, and prove that:

$$\mathcal{S}_{\overline{\mathbf{HAP}^f}} \models_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

Proving this latter condition correspond to prove the following ones, separately:

(i) $SOKB \cup \mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \models G\sigma$;

(ii) $SOKB \cup \mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \models \mathcal{IC}_S$;

(iii) $\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ ($\neg$-consistency for $\mathbf{E}$ atoms);

(*iv*) $\{\mathbf{NE}(p), \neg\mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ ($\neg$-consistency for **NE** atoms);

(*v*) $\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF} \cup \mathbf{EXP}]\sigma$ (**E**-consistency);

(*vi*) $\mathbf{HAP}^f \cup [\mathbf{FULF} \cup \mathbf{EXP}]\sigma \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models \bot$ (fulfillment).

Thanks to Lemma 11, conditions (*i*) and (*ii*) hold on the basis of the soundness results of IFF [2], in particular condition (*ii*) holds when no $\neg\mathbf{H}()$ literal occurs in the body of social integrity constraints in $\mathcal{IC}_S$. We have then proved that condition (*ii*) above holds even when literals of kind $\neg\mathbf{H}()$ occur in the body of social integrity constraints. The IFF proof procedure handles negation in the body of integrity constraints in a different manner: in particular, negated literals are turned into positive ones, and moved to the head of the constraint as additional disjunct. We apply, instead, constructive negation [9] to $\neg H()$ literals (see transition *Non-happening*), and therefore benefit of the soundness results of this procedure.

Conditions (*iii*), (*iv*), and (*v*) hold for the same reasons explained in the open case (Section 6.3).

Condition (*vi*) (fulfillment) holds thank to transitions Fulfillment and Violation. By contradiction, let us suppose that condition (*vi*) does not hold. This can happen, as in the open case, if there exists an expectation $\mathbf{NE}(p)$ and the corresponding event $\mathbf{H}(p)$. In this case however, transition *violation* **NE** would apply leading to a node not along a successful closed derivation.

In the closed case, condition (*vi*) could fail to hold because there is an $\mathbf{E}(p)$ atom without a matching $\mathbf{H}(p)$. In his case, however, transition *Violation* **E** would apply, again leading to a node that cannot stand along a successful derivation.

□

6.5 Soundness with universally quantified abducibles

We are now able to introduce the proof of soundness in the general case, in which abduced atoms can contain universally quantified variables. This is a novel result, achieved in year 3 of the project.

We first provide a lemma (Lemma 12) that will be used as a backbone for the full proof of soundness.

As will be clear soon, in the proof of Lemma 12, we use a slightly different rule for *Unfolding* than the one used in the IFF and in the $\mathcal{S}$CIFF proof procedures. Let us call this transition *Unfolding**. Recall [4] that Unfolding is applicable to

1. an atom in a conjunct and a clause;
2. an atom in an implication and a set of clauses.

Transition *Unfolding** coincides with Unfolding in the first case and is defined as follows in the second:

Definition 20. *If*

$$PSIC_k = \{Atom, BodyIC \rightarrow HeadIC\} \cup PSIC',$$

and if the clauses $H_1 \leftarrow B_1$, ..., $H_n \leftarrow B_n$ belong to the SOKB, and $H_1, \ldots, H_n$ unify with Atom, Unfolding* *selects a clause $H_i \leftarrow B_i$ $(1 \leq i \leq n)$ and produces the following node:*

$$\begin{aligned} PSIC_{k+1} = \quad & \{Atom, BodyIC \rightarrow HeadIC, \\ & Atom' = H_i, B_i, BodyIC' \rightarrow HeadIC'\} \cup PSIC' \end{aligned}$$

where $Atom', BodyIC' \rightarrow HeadIC'$ is a copy of $Atom, BodyIC \rightarrow HeadIC$.

The proof of soundness of the IFF proof procedure is based upon the following Proposition (called Proposition 4.1, page 74 in [1])[2]:

Proposition 6.3. (Fung) *Given an $ALP \equiv \langle P, Ab, IC\rangle$, a node N and a set of computable immediate successors S of N, we have:*

$$Comp(T, P - Ab) \cup IC \models N \leftrightarrow \textit{the disjunction of the nodes in } S$$

where P is the set of predicate symbols in the language of the program.

The proofs that follow such Proposition 4.1 in [1], up to the proof of soundness of the whole IFF proof-procedure, do not consider the various transitions anymore, but only rely on Proposition 4.1. Thus, by extending Proposition 6.3 also for *Unfolding**, we prove that the IFF proof-procedure is sound also if enlarged with the further transition *Unfolding**.

Proposition 6.4. *Given a node N and a set of computable immediate successors S of N computed by transition Unfolding*, we have:*

$$Comp(T, P - Ab) \cup IC \models N \leftrightarrow \textit{the disjunction of the nodes in } S$$

where P is the set of predicate symbols in the language of the program.

Proof. Let us consider a node N with an implication $Atom, BodyIC \rightarrow HeadIC$ and a predicate H defined by the clauses $H_1 \leftarrow B_1, \ldots, H_n \leftarrow B_n$.

Transition *Unfolding** produces the node

$$Atom, BodyIC \rightarrow HeadIC \bigwedge Atom' = H_i, B_i, BodyIC' \rightarrow HeadIC'$$

that is obviously logically equivalent to node N. □

Thus, the IFF proof procedure extended with transition *Unfolding** is sound.

Again, since the proof of soundness of the $\mathcal{S}$CIFF proof procedure without universally quantified abducibles, call it $\mathcal{S}\text{CIFF}_{\setminus\forall}$ (Sections 6.3 and 6.4) was based on soundness of the IFF proof procedure, also $\mathcal{S}$CIFF with *Unfolding** is sound. We can now base the proof of soundness of $\mathcal{S}$CIFF (*with* universally quantified abducibles) on the soundness of the $\mathcal{S}\text{CIFF}_{\setminus\forall}$ enlarged with transition *Unfolding**.

Lemma 12. *Consider an (open/closed) successful derivation D*

$$N_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_{n-1} \rightarrow N_n$$

with

$$N_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \mathbf{HAP}, \emptyset, \emptyset\rangle,$$

$$N_n \equiv \langle \emptyset, CS_n, PSIC_n, \mathbf{EXP}_n, \mathbf{HAP}, \mathbf{FULF}_n, \emptyset\rangle$$

in which transition Happening is not applied. Let $\mathbf{EXP}_n^{\forall} \subseteq \mathbf{EXP}_n$ and $\mathbf{FULF}_n^{\forall} \subseteq \mathbf{FULF}_n$ be the sets of (pending and fulfilled) abduced expectations of type $[\neg]\mathbf{NE}$. Let σ be the substitution

[2]In the conventions in [1], T is a theory (i.e., a logic program), $Comp(T, A)$ is the *Selective Completion* of the theory T to the predicates in A, i.e., the predicates whose heads do not belong to the set A are not completed.

applied to $\mathbf{EXP}_n \cup \mathbf{FULF}_n$ *by answer extraction on node* N_n*; by definition of answer extraction, variables in* $[\mathbf{EXP}_n \cup \mathbf{FULF}_n]\sigma$ *are universally quantified.*

Consider a society with

$$SOKB' = SOKB \cup [\mathbf{EXP}_n^{\forall} \cup \mathbf{FULF}_n^{\forall}]\sigma \tag{8}$$

(meaning that for each atom $A \in [\mathbf{EXP}_n^{\forall} \cup \mathbf{FULF}_n^{\forall}]\sigma$ *there is a clause* $A \leftarrow true$ *in* $SOKB'$*) in which the literals* $\mathbf{NE}$ *and* $\neg\mathbf{NE}$ *are considered as defined predicates*[3]*. In this case, there exists an (open/closed) successful derivation* D' *starting from the initial node*

$$N'_0 \equiv \langle \{G\}, \emptyset, \mathcal{IC}_S, \emptyset, \mathbf{HAP}, \emptyset, \emptyset \rangle$$

terminating in a node N'_m *such that*

$$\begin{aligned}
R'_m\sigma &= R_n\sigma \qquad [=\emptyset] && (9)\\
CS'_m &\subseteq CS_n && (10)\\
PSIC'_m\sigma &= PSIC_n\sigma && (11)\\
\mathbf{EXP}'_m\sigma &= \left[\mathbf{EXP}_n \setminus \mathbf{EXP}_n^{\forall}\right]\sigma && (12)\\
\mathbf{FULF}'_m\sigma &= \left[\mathbf{FULF}_n \setminus \mathbf{FULF}_n^{\forall}\right]\sigma && (13)\\
\mathbf{VIOL}'_m\sigma &= \mathbf{VIOL}_n\sigma \qquad [=\emptyset] && (14)
\end{aligned}$$

and the assignment σ *trivially satisfies the constraints* $CS_n \setminus CS'_m$*.*

Proof. Let θ be the substitution that binds each existentially quantified variable in each node of the derivation D to its final value in N_n/σ (thus, $\sigma \subseteq \theta$). Let $\Delta_n^{\forall} = [\mathbf{EXP}_n^{\forall} \cup \mathbf{FULF}_n^{\forall}]$.

We build the derivation D' from the derivation D; we show that for each node $N_i \in D$ there is a node $N'_j \in D'$ such that $N_i/\theta = N'_j/\theta$, except for the sets of abduced, for which

$$\begin{aligned}
[\mathbf{EXP}'_j]\theta &= [\mathbf{EXP}_i \setminus \mathbf{EXP}_n^{\forall}]\theta\\
[\mathbf{FULF}'_j]\theta &= [\mathbf{FULF}_i \setminus \mathbf{FULF}_n^{\forall}]\theta
\end{aligned}$$

By induction, we will assume that the thesis holds up to node N_i in D, and that there exists a corresponding node N'_j in D', and we prove that the thesis holds for $N_{i+1} \in D$, $N'_{j+d_j} \in D'$ for some $d_j \geq 0$.

We show that, given the transition Tr from N_i to N_{i+1}, there is one (or more) transition Tr' in D' applicable to the node N'_j leading to a node N'_{j+d_j} for which the thesis holds.

Transition Tr can be one of the following:

Unfolding. In this case, Tr' is also unfolding, applied to the same atom and clause. Since the thesis holds for N_i and N'_j, it holds also for N_{i+1} and N'_{j+1}.

Abduction. If the literal selected for abduction, L, is of type $[\neg]\mathbf{NE}$, then Tr' is Unfolding applied to the same literal L and to the clause C defined as follows.

- If L contains universally quantified variables, we know from Lemma 4 that L will be in all the descendant nodes, so it will be in $\Delta_n^{\forall}$, and there will be a corresponding

[3]Recall that literals $\neg\mathbf{NE}$ are mapped into new positive literals.

clause C in $SOKB'$. By definition of $SOKB'$ and θ, C is $A \leftarrow true$, where $A = L/\theta$. The selected clause is C.

In fact, Abduction gives a node N_{i+1} such that

$$N_{i+1} \equiv \langle R_i \setminus \{L\}, CS_i, PSIC_i, \mathbf{EXP}_i \cup \{L\}, \mathbf{HAP}, \mathbf{FULF}_i, \emptyset \rangle$$

Unfolding gives a node N'_{j+1} such that

$$N'_{j+1} \equiv \langle R'_j \wedge \{true\} \setminus \{L\}, CS'_j \cup \{L = A\}, PSIC'_j, \mathbf{EXP}'_j, \mathbf{HAP}, \mathbf{FULF}'_j, \emptyset \rangle$$

We can apply the logical equivalence $Q \wedge true \leftrightarrow Q$ to R'_{j+1}. We can then apply constraint solving steps to deal with the constraint $L = A$; since $A = L/\theta$, the constraint rewrites to $true$ and gives the substitution $\eta = \theta|_{vars(L)}$ to the (existentially quantified) variables in L. We reach the node:

$$N'_{j+2} \equiv \langle R'_j \setminus \{L\}, CS'_j, PSIC'_j, \mathbf{EXP}'_j, \mathbf{HAP}, \mathbf{FULF}'_j, \emptyset \rangle \eta$$

for which the thesis trivially holds (given that it holds for N'_j).

- If L does not contain universally quantified variables, $L/\theta \in N_n/\sigma$, so it will be in $\Delta_n^{\forall}$, and there will be a corresponding clause C in $SOKB'$. The selected clause is C, and the proof follows the scheme in the previous bullet.

If the selected literal L is not of type $[\neg]\mathbf{NE}$, then Tr' is abduction.

Propagation. Propagation is applied to a literal L and an implication. If L is not of type $[\neg]\mathbf{NE}$, transition Tr' is Propagation applied to the same elements.

Otherwise, Propagation is applied in the node

$$N_i \equiv \langle R_i, CS_i, PSIC_i, \mathbf{EXP}_i, \mathbf{HAP}, \mathbf{FULF}_i, \emptyset \rangle$$

to an implication $A, L_1, \ldots, L_n \rightarrow Q \in PSIC_i$ and a literal L of type $[\neg]\mathbf{NE} \in \mathbf{EXP}_i \cup \mathbf{FULF}_i$. Let us suppose that $L \in \mathbf{EXP}_i$, being the proof for the case $L \in \mathbf{FULF}_i$ very similar. The resulting node is

$$N_{i+1} \equiv \langle R_i, CS_i, PSIC_i \cup \{A = L, L_1, \ldots, L_n \rightarrow Q\}, \mathbf{EXP}_i, \mathbf{HAP}, \mathbf{FULF}_i, \emptyset \rangle.$$

Transition Tr' is *Unfolding**, applied to the literal A occurring in the body of an implication, and the clause C selected as follows. Since $L \in \mathbf{EXP}_i$ in derivation D, then $L/\theta \in N_n/\sigma$, and there will be a corresponding clause C in $SOKB'$. By definition of $SOKB'$ and θ, C is $H \leftarrow true$, where $H = L/\theta$. The selected clause is C. *Unfolding** generates a node

$$N'_{j+1} \equiv \langle R'_j, CS'_j, PSIC'_j \cup \{A = L, true, L_1, \ldots, L_n \rightarrow Q\}, \mathbf{EXP}'_j, \mathbf{HAP}, \mathbf{FULF}'_j, \emptyset \rangle$$

to which the logical equivalence $F \wedge true \leftrightarrow F$ is applicable, leading to node N'_{j+2} for which the thesis holds.

Splitting. Tr' is splitting, applied to the same disjunction.

Case Analysis. Tr' is case analysis, applied to the same elements.

Factoring. If applied to atoms different from $[\neg]\mathbf{NE}$, Tr' is factoring applied to the same abducibles. Otherwise, factoring generates two nodes; only one of the two children will be in the derivation D.

In the first, factoring unifies two abducibles L_1 and L_2. If this first node is in D, by definition of θ we know that $L_1/\theta = L_2/\theta$. Thus, in $\Delta_n^{\forall}$ they correspond to the same clause C, so the thesis already holds for the nodes N_{i+1} and N'_j (i.e., we do not introduce a transition in the derivation D' at this step).

In the second node, factoring imposes that $L_1 \neq L_2$. Thus, in $\Delta_n^{\forall}$ they correspond to different clauses C, so, again, the thesis already holds for the nodes N_{i+1} and N'_j and we do not introduce a transition in the derivation D'.

Equivalence Rewriting. Tr' is the same equivalence rewriting rule, applied to the same elements.

Happening. Is not considered.

non-Happening. Tr' is non-Happening, applied to the same elements.

Closure. Tr' is closure.

Violation NE. Applies to a node N_i in which $\mathbf{NE}(X) \in \mathbf{EXP}_i$ and $\mathbf{H}(Y) \in \mathbf{HAP}_i$ such that X and Y are unifiable. Violation **NE** generates two nodes: in the first, X is unified with Y and a violation is raised (thus, this node cannot be in a successful derivation). In the second, $X \neq Y$ is imposed. Thus, X and Y will be bound to non unifiable terms by the substitution θ.

Fulfillment NE. In this case, we do not apply any transition; the thesis already holds for the pair of nodes N_{i+1} and N_j, as well as for N_i and N_j. Intuitively, Fulfillment **NE** simply moves an expectation from **EXP** to **FULF**. In the current lemma, this distinction is blurred, as all expectations of the type $[\neg]\mathbf{NE}$ are in the $SOKB'$, wether they were fulfilled or pending.

Fulfillment E, Violation E. Tr' is Fulfillment **E** (resp. Violation **E**), applied to the same elements.

We still have to show that D' is a successful derivation, i.e., N'_m is a node of quiescence. Concerning quiescence, we know that N_n is a node of quiescence for the society with $SOKB$. By construction, we know that $[\mathbf{EXP}'_m]\sigma \subseteq [\mathbf{EXP}_n]\sigma$ (and $[\mathbf{FULF}'_m]\sigma \subseteq [\mathbf{FULF}_n]\sigma$), while the other elements of the tuple are the same. Thus, if a transition that only involves the elements in the tuple is applicable to N'_m, then it is also applicable in N_n. The only difference is the $SOKB$; the only transition applied to clauses in $SOKB$ is *Unfolding*.

Unfolding can be applied to an atom in a conjunct of R, or in the body of an implication. But, since N_n is a final node, $R_n = \emptyset$, thus $R'_m = \emptyset$, so the first case cannot be. In the second case, an atom of a predicate defined in the $SOKB$ occurs in the body of an implication. Since the only difference stands in $[\neg]\mathbf{NE}$ atoms, and R_n is a node of quiescence, the only possibility is that a literal L of type $[\neg]\mathbf{NE}$ occurs in the body of a PSIC. But, in order to apply Unfolding, there must be a clause C matching with L. In this case, by definition of $SOKB'$, we would have a corresponding atom $A \in \mathbf{EXP}_n \cup \mathbf{FULF}_n$ matching L, and this would make transition Propagation applicable to node N_n, which is not. Thus, also the second case is impossible.

□

We are now ready to prove the theorem of soundness, previously stated as Theorems 3 and 4.

Theorem 3. Open Soundness. *Given an open society instance* $\mathcal{S}_{\mathbf{HAP}^i}$*, if*

$$\mathcal{S}_{\mathbf{HAP}^i} \mid\!\sim^{\mathbf{HAP}^f}_{\mathbf{EXP}\cup\mathbf{FULF}} G$$

with expectation answer (Definition 18) $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$ *then*

$$\mathcal{S}_{\mathbf{HAP}^f} \mid\!\approx_{(\mathbf{EXP}\cup\mathbf{FULF})\sigma} G\sigma$$

i.e., given Definition 14, $Comp(SOKB \cup \Delta\sigma) \cup \mathbf{HAP}^f \cup CET \models G\sigma$.

Proof. By Lemma 12, we know that there exists a successful derivation D' with the $SOKB'$ defined by equation 8. We have to prove (as we did for Proposition 6.1) that the following conditions hold:

(i) $SOKB \cup \mathbf{HAP} \cup [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma \models G\sigma$;

(ii) $SOKB \cup \mathbf{HAP} \cup [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma \models \mathcal{IC}_S$;

(iii) $\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma$ ($\neg$-consistency for $\mathbf{E}$ atoms);

(iv) $\{\mathbf{NE}(p), \neg\mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma$ ($\neg$-consistency for $\mathbf{NE}$ atoms);

(v) $\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma$ ($\mathbf{E}$-consistency);

(vi) $\mathbf{HAP} \cup [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models \bot$ (fulfillment).

Let us prove the conditions separately:

(i) D' is a successful derivation; by Proposition 6.1, we know that it is sound, so, in particular, condition (i) holds for derivation D':

$$Comp(SOKB' \cup [\mathbf{EXP}'_m \cup \mathbf{FULF}'_m]\sigma) \cup \mathbf{HAP} \cup CET \models G\sigma.$$

Thus, by definition of $SOKB'$ (Eq. 8):

$$Comp(SOKB \cup [\mathbf{EXP}^{\forall}_n \cup \mathbf{FULF}^{\forall}_n]\sigma \cup [\mathbf{EXP}'_m \cup \mathbf{FULF}'_m]\sigma) \cup \mathbf{HAP} \cup CET \models G\sigma;$$

$$Comp(SOKB \cup [\mathbf{EXP}^{\forall}_n \cup \mathbf{EXP}'_m]\sigma \cup [\mathbf{FULF}^{\forall}_n \cup \mathbf{FULF}'_m]\sigma) \cup \mathbf{HAP} \cup CET \models G\sigma.$$

By Equations 12 and 13 of Lemma 12,

$$Comp(SOKB \cup [\mathbf{EXP}_n \cup \mathbf{FULF}_n]\sigma) \cup \mathbf{HAP} \cup CET \models G\sigma.$$

(ii) Again, by Proposition 6.1, we know that D' is sound, so, in particular, condition (ii) holds for derivation D':

$$Comp(SOKB' \cup [\mathbf{FULF}'_m \cup \mathbf{EXP}'_m]\sigma) \cup \mathbf{HAP} \models \mathcal{IC}_S.$$

By Equation 8 of Lemma 12,

$$Comp(SOKB \cup [\mathbf{FULF}^{\forall}_n \cup \mathbf{EXP}^{\forall}_n]\sigma \cup [\mathbf{FULF}'_m \cup \mathbf{EXP}'_m]\sigma) \cup \mathbf{HAP} \models \mathcal{IC}_S,$$

and, as before, by Eq. 12 and 13,

$$Comp(SOKB \cup [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma) \cup \mathbf{HAP} \models \mathcal{IC}_S.$$

(*iii*) The condition holds for derivation D', so

$$\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq [\mathbf{EXP}'_m \cup \mathbf{FULF}'_m]\sigma.$$

By definition of $\mathbf{EXP}^{\forall}_n$ and $\mathbf{FULF}^{\forall}_n$ (Equations 12 and 13), there is no literal $[\neg]\mathbf{E} \in (\mathbf{EXP}^{\forall}_n \cup \mathbf{FULF}^{\forall}_n)$.

(*iv*) By definition of successful derivation, node N'_m is a quiescence node (no transition is applicable to it).

If there existed a term p such that both $\mathbf{NE}(p)$ and $\neg\mathbf{NE}(p) \in [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma$, then $\mathbf{NE}(p), \neg\mathbf{NE}(p) \in [\mathbf{FULF}^{\forall}_n \cup \mathbf{EXP}^{\forall}_n]\sigma$. This would mean that Unfolding would be applicable (twice) to the predicates defining $\mathbf{NE}$ and $\neg\mathbf{NE}$ in $SOKB'$ and the integrity constraint:

$$\mathbf{NE}(T), \neg\mathbf{NE}(T) \rightarrow false,$$

and this contradicts the fact that N'_m is a quiescence node.

(*v*) Analogous to (*iv*).

(*vi*) Since D' is a successful derivation, the following condition holds:

$$\mathbf{HAP} \cup [\mathbf{FULF}'_m \cup \mathbf{EXP}'_m]\sigma \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models false,$$

Thus, expectations of type $\mathbf{E}$ are fulfilled also in the final node N_n of the original derivation D.

We still have to show that $\forall p$,

$$\mathbf{HAP} \cup [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models false.$$

By contradiction, let us suppose that $\exists p$ such that $\mathbf{NE}(p) \in [\mathbf{FULF}_n \cup \mathbf{EXP}_n]\sigma$ and $\mathbf{H}(p) \in \mathbf{HAP}$. In this case, transition *Violation* $\mathbf{NE}$ would have been applicable, which contradicts the fact that D is a successful derivation (Lemma 12).

□

Theorem 4. Closed Soundness. *Given a closed society instance* $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$, *if*

$$\mathcal{S}_{\mathbf{HAP}^i} \vdash^{\overline{\mathbf{HAP}^f}}_{\mathbf{EXP} \cup \mathbf{FULF}} G$$

with expectation answer $(\mathbf{EXP} \cup \mathbf{FULF}, \sigma)$ *then*

$$\mathcal{S}_{\overline{\mathbf{HAP}^f}} \models_{(\mathbf{EXP} \cup \mathbf{FULF})\sigma} G\sigma$$

Proof. Similar to the previous proof (Theorem 3). □

7 Discussion and planned future activity

In this report, we have extended the proof of soundness of the $\mathcal{S}$CIFF proof procedure that was given in Deliverable D8 [4]. In Deliverable D8, the proof was limited to instances in which the final node does not contain abducibles with universally quantified variables. In the current report, this limitation is removed, and we prove soundness for the general case.

In future work we have to prove also completeness of the $\mathcal{S}$CIFF proof procedure.

Part II
Termination

8 Introduction

Xanthakos [10] proved the termination of the IFF proof procedure. In this document, we prove the termination of the $\mathcal{S}$CIFF proof procedure, by following the same path used by Xanthakos. We follow, for this document, the same structure of [10].

The basic changes to the proof of termination given by Xanthakos stand on the following facts:

Constraint Solving. In Constraint Logic Programming [5] constraint solving is intended as an extension of *unification*. We have taken the same viewpoint in the operational semantics of $\mathcal{S}$CIFF [4], and inserted the rules for equality rewriting in the constraint solving group of transitions. In the proof of termination of $\mathcal{S}$CIFF, constraint solving takes the place that was of equality rewriting in the proof of termination of the IFF.

Case Analysis. In the IFF, case analysis generates two branches, one of which simply inserts a new disequality, in the form $t = s \rightarrow false$. Such a disequality either is not used at all, or it makes the derivation fail, so for the purpose of proving termination, it is very easy. In $\mathcal{S}$CIFF, disequalities are passed to the constraint solver, which will typically use them for constraint propagation (including pruning in CLP(FD)), so we need to be more careful.

Dynamic occurrence of events. We first prove termination of the $\mathcal{S}$CIFF without considering dynamically incoming events. In other words, we rule out the transitions of *dynamically growing history* [4], and prove the termination of a "static" $\mathcal{S}$CIFF. We then extend the proof of termination for the full $\mathcal{S}$CIFF proof procedure in Section 13.

9 New restrictions on the $\mathcal{S}$CIFF

We give here the equivalent of the restrictions proposed by Xanthakos. Some of them are already stated in previous documents for the $\mathcal{S}$CIFF. Moreover, we need to make some assumptions on the behaviour of the Constraint Solver.

9.1 Splitting

Citing Xanthakos:

The first new restriction we enforce is the (exhaustive) application of splitting on any disjunctions in a node (i.e. whenever possible, splitting should be applied after an unfolding, propagation, case analysis, or previous splitting step). Then, any execution tree is an or-tree where any node is a conjunction of literals, implications and at most one disjunction $D_1 \vee \cdots \vee D_n$, where each D_i is a conjunction of literals and implications.

9.2 Equality rewriting and logical simplification

The second restriction that we pose is that we give logical simplification and equality rewrite rules the highest priority, i.e. they should be applied whenever possible. [10]

Note that the proof of soundness also relied on this assumption [3, Lemma 4.5].

Equality rewriting is substituted in the $\mathcal{S}$CIFF proof-procedure by more general transition rules, called *Constraint Solving*. We impose that *Constraint Solving* transitions are applied (together with logical simplification) before the other transitions (i.e., they have highest priority).

9.3 Case Analysis

Some equalities in the body of implications are not dealt with by equality rewrite rules, but by case analysis. Our third restriction is that case analysis is given the highest priority (after equality rewriting and logical simplification have been performed) when an implication is selected. Similarly to equality rewriting, we enforce that the left-most equality is selected first. This restriction simplifies the implications in a node and may also reduce the computational cost [10]

We take the same restriction proposed by Xanthakos. Notice that in the $\mathcal{S}$CIFF proof-procedure, Case Analysis can also be applied to a constraint in the body of an implication.

We coherently extend the definition of *Analysed for of an implication* [10, Definition 4.1.3, page 62] as follows:

Definition 21. *Given an implication I, we define $A(I)$, the* analysed form *of I, as the implication that is produced after exhaustively applying constraint solving and logical simplification on I, having extended constraint solving rules to include:*

- *If c is a constraint occurring in the body of an implication, and all the variables occurring in c are flagged, then delete c from the implication, and add it to the constraint store*

If $I = A(I)$, we say that I is in analysed form.

This definition extends the definition by Xanthakos for the case with constraints. Xanthakos uses the analysed form noticing that, concerning termination, Case Analysis can often be simplified. In fact, one of the branches generated by Case Analysis cannot introduce new atoms (except equality atoms). In our case, Case Analysis cannot introduce new atoms, except constraint atoms. However, constraint atoms cannot introduce other atoms, except constraints, so the same reasoning of Xanthakos can be applied also in this case.

In the following, we will use the name "atomic conjunct" as follows:

Definition 22. *We will call an* Atomic Conjunct *any literal occurring in a node in R,* **EXP***,* **FULF***,* **VIOL***.*

9.4 Assumptions on the Constraint Solver

As in Constraint Logic Programming [5], the Constraint Solving is not completely specified in the $\mathcal{S}$CIFF proof procedure. In order to prove termination, we need to make some assumption on the Constraint Solver.

Definition 23. *Assumptions on the Constraint Solver*

- *The constraint solving process always terminates*
- *The constraint solving process cannot generate an infinite constraint store*
- *If the constraint solving process generates a disjunction of constraints $CS = (c_1 \vee c_j) \wedge CS'$ then splitting can be applied. We require that the alternation of Constraint Solving and splitting always terminates.*
- *The constraint solving process will not change the quantification of a variable (a variable universally quantified will not become quantified existentially and vice-versa).*
- *The constraint solving process can change a literal L into L', but the new version, L' must be an instance of the previous version, L.*

Thanks to these assumptions, we can now state the following lemma:

Lemma 13. *Constraint Solving steps cannot cause other transitions, except*

- *Case Analysis*
- *failing transitions.*

Moreover, an infinite sequence of case analysis and constraint solving steps is impossible.

Proof. Constraint Solving steps can

- remove a constraint from the resolvent (transition *Constrain*)
- change the constraint solver (transitions *Infer*)
- possibly, fail (transition *Consistent*).

They do not change any element in the tuple except constraints and the variables that occur in the Constraint Store [4, pag 166].

The following transitions do not depend on a variable being instantiated or constrained:

- unfolding
- abduction
- propagation
- splitting of a disjunction in R
- case analysis on an equality in an implication
- happening

- non-happening
- closure
- fulfillment **E**, violation **NE**

The following transitions depend on the constraint store:

- splitting of a disjunction in the Constraint Store. However, in this case, the constraint store is changed, so we can assume that Constraint Solving is applied again. From a previous assumption (Definition 23), any sequence of Constraint Solving and Splitting always terminates. Moreover, both these transitions only modify the constraint store, so they can be considered as a unique transition for the purposes of proving termination.
- factoring depends on the quantification of variables of abducibles; however, we know that the rules for equality and disequality cannot change the quantification of a variable in an abducible [3, Lemma 4.5, pag 14]. It is consistent to assume that the constraint solver will not change the quantification of a variable.
- case analysis on a constraint depends on a variable being flagged, and the flagging of a variable can be changed by the constraint solver (Example **??**). However, an infinite sequence of constraint solving and case analysis is impossible, as constraint solving does not create new implications with constraints in the body and the number of constraints in the body of a node is finite.
- violation **E** (but gives a violation node, so the computation terminates)
- fulfillment **NE** (but the check can also be avoided, by using a given order of application of transitions [4, pag 163])

□

10 Acyclicity for $\mathcal{S}$CIFF programs

Definition 24. *Given a SOKB, an atom L* depends *on a literal M wrt SOKB if*

- *an instance of a clause in SOKB is $L\theta \leftarrow K, M$, or*
- *an instance of a clause in P is $L\theta \leftarrow K \wedge N$ and N depends on M*

where K is a conjunction of literals, possibly true, and $L\theta$ is an instance of L.
Given a logic program P, an atom L weakly depends *on a literal M wrt SOKB if*

- *L is M, or*
- *L depends on M wrt SOKB.*

The following note by Xanthakos will be used in the proof of termination:

> Applying equality rewriting, logical simplification and case analysis after unfolding an atom L (occurring as an atomic conjunct or in the body of an implication) guarantees that the introduced literals (occurring as atomic conjuncts or in an implication) are such that L depends on them.

We report here some definitions given by Xanthakos, adapted to our terminology.

Definition 25. *Given a SOKB, two literals L, M are* related *wrt an atom N if an instance of a clause in P is $N\theta \leftarrow K, L', M'$ (where K is a conjunction of literals, possibly true, and $N\theta$ is an instance of N) and L' weakly depends on L and M' weakly depends on M.*

Intuitively, two literals are related wrt a goal, if if a sequence of unfolding steps for the goal can lead to the introduction of a node with both literals.

Definition 26. *Given a SOKB, a level mapping $||$ is a function that maps all ground atoms in B_{SOKB} (where B_{SOKB} is the Herbrand base of the logic program SOKB) to $\mathbb{N} \setminus \{0\}$ and false to 0. Also, $||$ is extended to map a ground negative literal $\neg A$ to $|A|$, where $A \in B_{SOKB}$.*

We can give now the definition of acyclic implication restated in our terminology:

Definition 27. (Acyclic implication). *Given a society with $SOKB$ acyclic wrt a level mapping $||$, a ground implication, say $L_1, \ldots, L_n \rightarrow H_1 \vee \cdots \vee H_m$, is called acyclic wrt SOKB and $||$, if for every non-constraint atom L_i, $i = 1, \ldots, n$, for every ground atom K which L_i weakly depends upon wrt SOKB,*

- *$|K| > |H_r|$, $r = 1, \ldots, m$ and*
- *$|K| > |N|$, for every non constraint atom N such that some L_j, $j = 1, \ldots, i-1, i+1, \ldots, n$ depends upon the negative literal $\neg N$ and*
- *$|K| > |N|$, for every non equality atom N such that K is related to $\neg N$ wrt L_i.*

An implication is called acyclic *wrt $SOKB$ and $||$ if every ground instance of it is acyclic wrt $||$.*

An implication is called acyclic *wrt $SOKB$ if it is acyclic wrt some level mapping.*

The definition of Acyclic Implication considers CLP constraints as an extension of the concept of unification (as is usual in CLP [5]). In other words, constraints are not assigned a level; this is reasonable, because they do not depend upon definitions, nor upon integrity constraints, but their semantics is defined only by the underlying constraint theory.

Notice that Definition 27 of acyclicity is independent from the presence of $\mathbf{H}$ literals in Social Integrity Constraints:

Proposition 10.1. *Consider an Abductive Logic Program $ALP \equiv \langle SOKB, \mathcal{E}, \mathcal{IC}_S \rangle$. Consider the Abductive Logic Program $ALP^1 \equiv \langle SOKB, \mathcal{E}, \mathcal{IC}_S^1 \rangle$ in which every literal $[\neg]\mathbf{H}$ in $\mathcal{IC}_S$ is replaced by true.*

The two following conditions are equivalent:

(i) ALP is acyclic wrt some level mapping $||$ and all the implications are bounded wrt $||$

(ii) ALP^1 is acyclic wrt some level mapping $||^1$ and all the implications are bounded wrt $||^1$

Proof. $(ii) \Rightarrow (i)$. Since $||^1$ is bounded, let N be the maximum level of an atom in ALP^1. We define $||$ as follows:

- $|L| = |L|^1$ for all literals L different from $[\neg]\mathbf{H}$;
- $|L| = N + 1$ if L is of type $[\neg]\mathbf{H}$

By the syntax of Social Integrity Constraints (see Deliverable D8), literals $[\neg]\mathbf{H}$ can only occur in the body of an ic_S in $\mathcal{IC}_S$, so ALP is trivially acyclic and bounded wrt the level mapping $||$.

$(i) \Rightarrow (ii)$. Viceversa, we define $||^1$ as the restriction of $||$ to the literals with functor different from $\mathbf{H}$. □

We now extend the notion of acyclicity to the society knowledge (which is the definition of acyclic ALP [10, Def 4.2.5 pag 65] rewritten in our terminology)

Definition 28. • *Given a logic program P that is acyclic wrt a level mapping $||$, a negative defined literal $\neg N$ is called acyclic wrt P and $||$ if the implication $N \rightarrow false$ is acyclic wrt P and $||$. A negative defined literal is called acyclic wrt P if it is acyclic wrt some level mapping.*

- *A* society $\mathcal{S}$ is acyclic w.r.t. a level mapping $||$ *if*
  1. *$SOKB$ is acyclic wrt $||$*
  2. *all negative defined literals in $SOKB$ are acyclic w.r.t $SOKB$ and $||$*
  3. *every implication in $\mathcal{IC}_S$ is acyclic wrt $SOKB$ and $||$.*

  A society $\mathcal{S}$ is called acyclic *if it is acyclic w.r.t. some level mapping.*

- *A query G to a society $\mathcal{S}$ where the $\mathcal{S}$ is acyclic wrt some level mapping $||$, is called acyclic w.r.t. $\mathcal{S}$ and $||$ if every negative defined literal in G is acyclic w.r.t. $SOKB$ and $||$. $\mathcal{S}$ and G are then called acyclic w.r.t. $||$. A society $\mathcal{S}$ and a query G are called acyclic if they are acyclic w.r.t. some level mapping.*

Notice that the definition of *acyclic negative literal* is slightly different from the IFF, because the $\mathcal{S}$CIFF proof procedure does not rewrite *all* negative literals $\neg N$ to $N \rightarrow false$, but only the negative defined literals, while abducibles have explicit negation [4], and constraints depend on the solver (for example, $\neg(A < B)$ is typically rewritten as $A \geq B$). Thus, literals $\neg\mathbf{E}$, $\neg\mathbf{NE}$ and $\neg c$ (where c is a constraint) are always acyclic.

11 Termination

In this section we give a result for the termination of the $\mathcal{S}$CIFF proof-procedure. We report the definitions given by Xanthakos, for the sake of understandability.

First, we extend the notion of bounded literals to implications.

Definition 29. (Xanthakos) Bounded implication. *An implication I is called bounded with respect to a level mapping $||$, if all the literals in the body and head of I are bounded wrt $||$.*

We state the theorem of termination for a *"static version* of $\mathcal{S}$CIFF proof-procedure, i.e., for a version of $\mathcal{S}$CIFF that does not have Happening, non-Happening, and closure transitions. In other words, we prove termination for a version of $\mathcal{S}$CIFF provided with a static history. We will then extend, in Section 13, the proof for the dynamic case.

Theorem 5. (Termination of static $\mathcal{S}$CIFF) *Let G be a query to a society $\mathcal{S}$, where $SOKB$, $\mathcal{IC}_S$ and G are acyclic wrt some level mapping, and G and all implications in $\mathcal{IC}_S$ bounded wrt the level-mapping. Then, every $\mathcal{S}$CIFF derivation for G, where transitions Happening, non-happening, and closure are not applied, for each instance of $\mathcal{S}$ is finite.*

We report the definitions of *causes* and *bounded node* from [10]:

Definition 30. Causes *(Xanthakos). Given two execution steps E and E', E* causes *E' if E is applicable in a node N producing a node N', E' is not applicable in N and E' is applicable in N'.*

Definition 31. Bounded node *(Adapted from [10])*

- *A disjunction $D_1, \dots, D_k$, $k > 0$, where D_i, $1 \leq i \leq k$, is a conjunction of literals and implications, is called bounded w.r.t. a level mapping $||$ if all non-constraint conjuncts of every D_i are bounded w.r.t. $||$.*

- *A node N in an execution tree is called bounded w.r.t. a level mapping $||$ if all its non-constraint conjuncts are bounded w.r.t. $||$.*

As done by Xanthakos, we consider nodes in canonical form, i.e., after applying two types of transitions. Xanthakos considers equality rewriting and logical equivalence. As we extend the language with constraints, we consider a node in canonical form after application of constraint solving and logical simplification. Thus, we consider the executions steps to be augmented with constraint solving and logical simplification. We need the following lemma to ensure that this assumption is safe (extended from [10, Lemma 4.3.1]):

Lemma 14. *Any sequence of Constraint Solving or logical simplification steps, performed on a finite node, is finite.*

Proof. The lemma holds for Equality Rewriting and Logical Simplification [10, Lemma 4.3.1, pag 69]. By assumption (Definition 23), constraint solving always terminates. Since the node is finite, the number of constraints in the node is finite, therefore any sequence of constraint solving steps is finite. Also, any sequence of logical simplification steps in a finite node is finite [10, Lemma 4.3.1]. Constraint Solving steps can cause logical simplification steps, but the reverse does not hold. Since any sequence of Constraint Solving steps in a finite node is finite, we conclude that any sequence of Constraint Solving or logical simplification steps is finite. □

The proof of Theorem 5 follows the same proof given by Xanthakos for the IFF proof-procedure. We first assume that transitions for dynamically growing history (i.e., *happening, non-happening* and *closure*) will not be applied. Then, we extend the proof of termination to the full proof-procedure in Section 13.

11.1 Non-introducing steps

In this section, we prove that any sequence of non-introducing steps is finite. By non-introducing step, we take a slightly different definition w.r.t. Xanthakos [10].

Definition 32. *An* introducing step *is a transition E that, after application of constraint solving and logical equivalence steps, leads from a node N to N' such that a new literal is introduced in $R \cup \mathbf{EXP} \cup \mathbf{FULF} \cup \mathbf{VIOL}$.*

A non-introducing step *is a transition E that, after application of constraint solving and logical equivalence steps, leads from a node N to N' such that no new literal is introduced in $R \cup \mathbf{EXP} \cup \mathbf{FULF} \cup \mathbf{VIOL}$.*

The definition has different consequences w.r.t. the definition by Xanthakos. For example, adding a constraint c to the set R is a non-introducing step; in fact, after constraint solving, c will be moved to the constraint store. Adding a literal $\neg \mathbf{E}$ is an introducing step. Adding a literal $\neg p$, where p is a defined literal (there exists a definition for p in $SOKB$), is a non-introducing step, because Logical Equivalence will remove $\neg p$ from R and add $p \rightarrow false$ to PSIC.

We prove the following lemma, that is the extension to the $\mathcal{S}$CIFF, of Lemma 4.3.2 by Xanthakos [10, Lemma 4.3.2]

Lemma 15. *Consider a sequence $N_1, N_2, \ldots$ of nodes in T, where any N_{i+1}, $i = 1, 2 \ldots$, is a successor of N_i produced by an execution step E. If no execution step E introduces a non-constraint atomic conjunct in N_{i+1}, then the sequence is finite.*

The sequence of successive nodes defines a sequence of execution steps, none of which introduces a non-constraint atomic conjunct. We are interested in sequences of the following execution steps:

1. Constraint Solving or logical simplification.
2. Case analysis that does not introduce a non-constraint atomic conjunct.
3. Factoring.
4. Splitting that does not introduce a non-constraint atomic conjunct.
5. Fulfillment and Violation.
6. Unfolding that does not introduce a non-constraint atomic conjunct.
7. Propagation that does not introduce a non-constraint atomic.

We will prove that any sequence of the five first types of steps is finite. To cater for the two latter cases, we define the auxiliary level of an implication and the auxiliary level of a node. These notions are only used in this section and are very similar to the corresponding definitions given by Xanthakos. They are different from the level of implication and level of node defined in later sections.

Definition 33. *Consider a node N in the execution tree T, an implication I in N and an atomic conjunct L in N. We define*

- *the* auxiliary level *of I, denoted as $|I|_a$, as the multiset of the levels of the non-constraint atoms in $A(I)$, if the analysed form $A(I)$ of I is neither $true$ nor $false$.*
- *the* propagation degree *of I in N, denoted as $P(I, N)$, as the number of propagation steps involving I that are applicable in N.*
- *the* auxiliary level *of L, denoted as $|L|_a$, as the multiset $\{|L|\}$.*
- *the* extension of a node N*, denoted as $E(N)$, as the node that contains all conjuncts of N, where all variables in $E(N)$ are quantified as in N, plus $P(I, N)$ copies of every implication I in N.*

- *the* auxiliary level of a simple node N*, denoted as* $|N|_a$*, as the multiset of the auxiliary levels of all non-constraint conjuncts of* $E(N)$*, other than implications whose analysed form is* $true$ *or* $false$*.*

- *the auxiliary level of a node* $N = [(D_1 \vee \cdots \vee D_k) \wedge Rest]$*, denoted as* $|N|_a$*, where* $k \geq 1$ *and* D_i*,* $Rest$ *denote conjunctions of atoms and implications, as* $max(|N_1|, \ldots, |N_k|)$*, where* N_i*,* $1 \leq i \leq k$*, is the simple node* $N_i = [D_i \wedge Rest]$*.*

Proof. of Lemma 15. Again, this proof follows the steps of the corresponding proof given by Xanthakos [10, Lemma 4.3.2], extending it for constraint solving and Fulfillment-Violation. The proof is organised in three parts. First, we prove that any sequence of steps of type (1), (2), (3), (4) or (5) (as identified at the beginning of this subsection) is finite. Next we show that for any node N' that is the successor of N, produced by the application of an execution step E on N, where E is of type (1), (2), (3), (4) or (5), we have that $|N|_a = |N'|_a$. Finally, we prove that for any node N' that is the successor of N, produced by the application of an execution step E on N, where E is of type (6) or (7), we have that $|N|_a > |N'|_a$.

For the first part, consider a sequence of type (1), i.e. Constraint Solving or logical simplification steps. Any such sequence is finite (Lemma 14).

Consider a sequence of interleaving steps of type (1) or (2). Any sequence of steps of type (2), i.e. case analysis, has to be finite because any node has a finite number of implications, and any implication has a finite number of constraints in its body (reduced by one every time a case analysis step in applied on it). Steps of type (2) can cause steps of type (1), but not vice versa. Therefore, a sequence of interleaving steps of type (1) or (2) cannot be infinite, because that would imply an infinite chain of steps of type (1).

Consider a sequence of interleaving steps of type (1), (2) or (3). Factoring produces two nodes, where the first contains an equality, while the second contains a disequality. In the first node the only new steps that may be performed are constraint solving and logical simplification (which may be performed a finite number of times). In the second node, a disequality is imposed. In the IFF proof-procedure, a disequality cannot propagate anything, and can only fail. In the $\mathcal{S}$CIFF, it can also start a constraint propagation; but in such a case, again the new applicable steps are constraint solving and logical simplification. Therefore, a sequence of interleaving steps of type (1) or (2) or (3) cannot be infinite, because that would imply an infinite sequence of steps of type (1) or (2).

Consider a sequence of interleaving steps of type (1), (2), (3) or (4). Any sequence of splitting steps on a disjunction is finite, as the number of disjunctions in a node is finite, reduced by one after every splitting step. Splitting a disjunction can cause steps of type (1) and (3), but not vice versa. Therefore, a sequence of interleaving steps of type (1), (2), (3) or (4) cannot be infinite, because that would imply an infinite sequence of steps of type (1) or (3).

Any sequence of splitting steps on an implication is finite, because the number of disjunctions in an implication and the number of implications in a node are finite and are reduced by one after every splitting step. Splitting an implication $I = [R \rightarrow H \vee L]$ in a node $N = [I \wedge Rest]$ produces two nodes $N' = [J \wedge Rest]$, where $J = [R \rightarrow H]$, and $N'' = [L \wedge Rest']$.

Consider N'. Any step that is applicable on N' is also applicable on N, therefore no step is caused in this case. Now, consider N''. If L is a non-constraint atom, then N'' cannot be part of the sequence of nodes we consider. If L is a constraint, only steps of type (1) are caused. Therefore, a sequence of interleaving steps of type (1), (2), (3) or (4) cannot be infinite, because that would imply an infinite sequence of steps of type (1).

Consider a sequence of interleaving steps of type (1), (2), (3), (4) or (5). Fulfillment and violation transitions have the following behaviours

- Fulfillment **E**, violation **NE** produce two nodes, one with an equality and one with a disequality. We can apply the same reasoning used for factoring.
- Violation **E**, if applicable, leads to a violation node
- Fulfillment **NE** cannot cause other transitions

For the second part, it is easy to see that for any step of type (1), (2), (3) or (5) applied on a node N and producing a node N', $|N|_a \geq |N'|_a$.[4] Constraint solving steps may change the arguments of a literal L in R, **EXP**, **FULF**, **VIOL**, however the changed literal L' is an instance of L (by assumption), therefore based on the definitions of the level mapping and auxiliary level, $|L|_a \geq |L'|_a$.

In the sequel, when an execution step introduces an equality, equality rewriting may change $Rest$ to $Rest'$, R to R' and H to H'. Extending the reasoning of the previous paragraph, we have that $|Rest|_a \geq |Rest'|_a$, $|R|_a \geq |R'|_a$ and $|H|_a \geq |H'|_a$.

Concerning type (4) (splitting), suppose an implication $I = [R \rightarrow H \vee L]$ in a node $N = [I \wedge Rest]$, where L is an atom with only flagged variables, is split. Then, two nodes are produced, namely $N' = [J \wedge Rest]$, where $J = [R \rightarrow H]$ and $N'' = [L \wedge Rest']$.

Consider node N'. Clearly, $|I|_a = |J|_a$, therefore $|N|_a > |N'|_a$.

Consider node N''. If L is a non-constraint atom, then it cannot be in the sequence of nodes we consider. If it is a constraint, since constraints do not contribute to the level of a node, $|N|_a \geq |N''|_a$.

For the third part:

- Step of type (6) (unfolding)

unfolding a conjunct Suppose an atom $L \in R$ in node $N = \langle R, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL} \rangle$ is unfolded, introducing no literals in R', **EXP**', **FULF**', and **VIOL**'. I.e. suppose unfolding introduces only constraints in CS' and negative literals rewritten as denials in PSIC'; thus the produced node is $CS' = CS \cup \{c\}$, $PSIC' = PSIC \cup \{[M_1 \rightarrow false], \dots, [M_o \rightarrow false]\}$. Since L depends on $\neg M_j$, $j = 1, \dots, o$, we have that $|L| > |\neg M_j| = |M_j|$. Moreover, L cannot be false (because it was unfolded), thus $|L| > 0 = |false|$. Therefore, $|L|_a = \{|L|\} > \{|M_j|, 0\} = \{|M_i|, |false|\} = |M_j \rightarrow false|_a$. Thus, $|N|_a > |N'|_a$.

unfolding an implication Suppose an atom L in an implication $I = [L \wedge R \rightarrow H]$ in node $N = I \wedge Rest$ is unfolded, introducing no non-constraint atomic conjuncts in the produced node N', i.e. $N' = J_1 \wedge \cdots \wedge J_o \wedge Rest$. Let $A(J_i) = L_1 \wedge \cdots \wedge L_n \wedge R' \rightarrow H' \vee M_1 \vee \cdots \vee M_k$. Since L depends on all $L_1, \dots, L_n, M_1, \dots, M_k$, we have that $|I|_a > |J_i|_a$. Thus, $|N|_a > |N'|_a$.

[4]In fact

1. Constraint Solving can transform literals/implications by making them more specific (the new version is an instance of the previous), Logical Equivalence (that does not introduce a literal) ...
2. Case Analysis either removes an implication, or it adds it without a constraint (and constraints do not have a level)
3. Factoring adds a constraint (which does not have a level)
4. Fulfillment and Violation can only add constraints (that do not have a level).

- Step of type (7) (propagation) Suppose an atom $p(t) \in R \cup \Delta$ and an implication $I = [p(s) \wedge R \rightarrow H]$ in a node $N = p(t) \wedge I \wedge Rest$ are selected for a propagation step, producing the node $N' = [p(t) \wedge I \wedge J \wedge Rest']$, where $A(J) = R' \rightarrow H'$.

  Clearly, $|I|_a > |J|_a$. Also, $P(I, N) > P(I, N')$, because the propagation step on N that produced N' is not applicable in N'. Thus, $|N|_a > |N'|_a$.

We proved that any sequence of execution steps of type (1), (2), (3), (4) or (5) is finite. Moreover, if the first node of such a sequence is N_1 and the last one is N_k, then $|N_1|_a \geq |N_k|_a$. We also proved that any execution step of type (6) or (7) on a node N produces a node N' such that $|N|_a > |N'|_a$. Thus, an infinite sequence of non-introducing steps entails an infinite decreasing sequence, which contradicts that the standard multiset ordering is well-founded. Therefore, any sequence of non-introducing execution steps is finite.

□

11.2 Level of an implication

Definition 34. *Given an acyclic $\mathcal{S}$, for a bounded implication I where $A(I)$ is neither true nor $false$, we define*

- *$PA(I)$, the potential atoms of I, as the multiset $\{H_1, \dots, H_m, M_1, \dots, M_k\}$, where $H_1, \dots, H_m$ are the non-constraint atoms in the head of $A(I)$ and M_j, $j = 1, \dots, k$ is any non-constraint atom such that some atom in the body of $A(I)$ depends on some defined literal $\neg L$, and L depends on M_j w.r.t. SOKB;*

- *$|I|$, the level of I, as $max(|E_1|, ..., |E_n|)$, where $E_i \in PA(I)$, $i = 1, \dots, n$.*

With respect to Xanthakos's definition [10, def 4.3.5], we had to add the literal $\neg L$. In fact, the $\mathcal{S}$CIFF proof procedure does not move to the head literals representing expectations, but it moves defined literals. See the following example:

Example 1. *Consider a society with the following integrity constraints:*

$$p \wedge q \rightarrow r$$

and the following SOKB:

$$\begin{array}{lcl} q & \leftarrow & \neg \mathbf{E}(a(X)) \\ p & \leftarrow & \neg s \\ s & \leftarrow & \mathbf{E}(a(Y)) \end{array}$$

Unfolding q does not move $\mathbf{E}(a(X))$ to the head:

$$p \wedge \neg \mathbf{E}(a(X)) \rightarrow r$$

while unfolding p moves s to the head; so, unfolding s, we obtain:

$$\neg \mathbf{E}(a(X)) \rightarrow r \vee \mathbf{E}(a(Y))$$

Notice the different behaviour for defined literals and expectations.

11.3 Level of a node

Citing Xanthakos:

> Our aim is to define the level of a node N so that if a non-introducing step is performed on N, producing N', the level of N should be greater or equal to the level of N'. To achieve this, we compute a priori all the conjuncts that may be introduced by any sequence of non-introducing steps, i.e. execution steps involving an implication J (in N or a subsequent node) and possibly an atomic conjunct L of N (but not atomic conjuncts introduced in subsequent nodes). These conjuncts may be introduced by unfolding an atom in the body of J, by propagating J and L, by splitting J, or any combination of the three. To define these conjuncts, we introduce the notion of family of an implication in a node. Note that the analysed form of an atomic conjunct is the atomic conjunct itself.

We propose an equivalent notion of family of an implication. Intuitively, the family of an implication contains the atoms that can be inserted by the implication by using only the atomic conjuncts that are already in the node N. It is not the full $\mathcal{S}$CIFF proof-procedure, because it does not recursively consider the atoms that are not in N (but only in subsequent nodes):

Definition 35. *Given a node N of the execution tree T and an implication I in N, we define $F(I, N)$, the* Family of I in N *as the multiset of the analysed forms $A(J)$ of the implications J (such that $A(J)$ is neither true nor false), where J is*

- *I or*
- *produced by unfolding of an atom in the body of an implication in $F(I, N)$ or*
- *produced by the propagation of an implication in $F(I, N)$ and an atomic conjunct in N or*
- *produced by splitting the head of an implication in $F(I, N)$.*

We refer to a conjunct C that is not in a node N but is in the family of an implication I in a node N as a *potential conjunct* of N. Note that for any conjunct C, $C \in F(I, N)$, we have that $|I| \geq |C|$, based on the definition of the level of an implication.

Xanthakos also proved that the family of an implication is finite [10, Lemma 4.3.3]. The proof also holds (with minimal adjustments) also for the $\mathcal{S}$CIFF proof-procedure. We report it here for the sake of completeness:

Lemma 16. *Given a node N of the execution tree T and an implication I in N, $F(I, N)$ is finite.*

Proof. The computation of $F(I, N)$ corresponds to an execution tree T' with root node the node N, where the only execution steps performed are the ones given in the definition of $F(I, N)$. To prove that $F(I, N)$ is finite, it is enough to prove that T' is finite. Since the maximum branching degree of T' (and indeed of any execution tree) is 2, it is enough to prove that T' has no infinite branches.

Consider an arbitrary branch B of T'. B defines a sequence S of execution steps.

Suppose all execution steps in S are non-introducing steps (i.e. do not introduce any non-constraint atomic conjuncts). Based on Lemma 15, we have that S is finite.

Suppose that all execution steps in S are introducing steps. Suppose the introducing step E is performed on N, producing the node N_1, where the non-constraint conjunct L was introduced in N_1. Suppose also that the introducing step E_1 was performed on N_1, producing the node N_2, where the non-constraint conjunct L' was introduced in N_2.

$$T' : \qquad N \xrightarrow{E} N_1 \xrightarrow{E_1} N_2$$

Since the execution steps allowed in T' cannot involve L, the execution step E_1 is applicable on N. This argument holds for any introducing execution step E_i in S, i.e. E_i is applicable on N. It is easy to see that the number of execution steps that are applicable on a finite node is finite. Thus, the number of introducing execution steps in S is finite, i.e. S is finite.

With similar reasoning, we see that any subsequence S' of execution steps in S consisting of non-introducing steps only (we refer to this type of sequences as type 1) or introducing steps only (we refer to this type of sequences as type 2) is finite.

Suppose S is infinite. Then, there must be an infinite number of finite subsequences of type 1 and type 2 that are interleaved in S. Note that no introducing step permitted in T' causes a non-introducing step permitted in T' . Therefore, an infinite sequence of type 1 can be constructed from S by removing all subsequences of type 2 in S. This contradicts Lemma 15.

Therefore, S is finite, i.e. the arbitrary branch B of T' is finite. Therefore, T' is finite because the maximum branching degree of T' is two. We conclude that $F(I, N)$ is finite and can be computed in a finite number of steps. □

The contribution of an atomic conjunct A to the level of a node N is not simply the level of the atom A itself, but also includes the level of the atoms that may be inserted in successive nodes because of the presence of A in N. To cater for this, Xanthakos proposed to include two occurrences of all levels of implications and potential conjuncts in he level of a node. Intuitively, this will be useful when comparing the level of two nodes: given an implication $a \rightarrow b$ in a node N, we count twice the level of b, so, if, later on, b is actually introduced in a node, the level of such a node will be less than the level of N.

We rewrite the definition of level of a node in our syntax:

Definition 36. *Consider a bounded simple node* $N = \langle R, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL}\rangle$*, with* $PSIC = \{I_1, \ldots I_m\}$*, and* $L_1, \ldots, L_n \in R \cup \mathbf{EXP} \cup \mathbf{HAP} \cup \mathbf{FULF}$*. We define:*

1. $Pt(N)$*, the* potential of N*, as the multiset* $F(I_1, N) \uplus \cdots \uplus F(I_m, N)$ *(where* $\uplus$ *is multiset union, i.e., multiple occurrences are preserved).*
2. $|N|$*, the* level of N*, as the multiset* $|N| = \{|L_1|, \ldots, |L_n|, |E_1|, |E_1|, \ldots, |E_r|, |E_r|\})$*, where* $E_1, \ldots, E_r$ *are all implications in* $Pt(N)$ *whose analysed form is neither true nor false.*

The level of a (possibly not simple) node N is the maximum of the levels of the simple nodes that can be derived by N by splitting.

Definition 37. *Consider a bounded node* $N = \langle R, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL}\rangle$*, where* $R = D_1 \vee \cdots \vee D_k$*. We define* $|N|$*, the level of* N*, as* $max(|N_1|, \ldots, |N_k|)$*, where* N_i *is the simple node with* $\langle D_i, CS, PSIC, \mathbf{EXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL}\rangle$*.*

Remember that $\mathcal{S}$CIFF also applies *splitting* to disjunctions in the constraint store. However, the constraint store can only contain constraints, that do not have a level.

11.4 Acyclicity and propagation

The following lemma, adapted from [10, Lemma 4.3.4], proves that propagation steps will preserve acyclicity and, moreover, that any propagation step involving a PSIC and an atom A will only introduce atoms with level less than $|A|$.

Lemma 17. *Let L be a happened event or an expectation, and I a PSIC in a node N in the execution tree T, for the query Q wrt an acyclic society knowledge base. Then we have:*

(1) *$A(I)$ is acyclic wrt SOKB and $||$.*

(2) *for any $J \in F(I, N)$, J is acyclic*

(3) *$|L| > |E|$, where $|E|$ is any conjunct[5] produced by propagating $|L|$ and some $J \in F(I, N)$.*

Proof. We prove part (1) by induction on the nodes of the execution tree. In the initial node, all the implications are acyclic by hypothesis.

Suppose that the analysed forms of all implications in node N_k are acyclic. We need to prove that whatever inference rule is applied on N_k, the analysed forms of all implications in the resulting node N_{k+1} are also acyclic. Obviously, we only need to consider execution steps that introduce new implications (and do not introduce false).

We consider the following execution steps.

1. Unfolding

   (a) Unfolding an atom $L = p(T)$ in the body of a PSIC $I = p(T) \wedge R \rightarrow H$ in a node $N_k = [I \wedge Rest]$ we get the node $N_{k+1} = [I_1 \wedge \cdots \wedge I_r \wedge Rest]$. Consider one of the new implications I_j, where $A(I_j) = [p_1(s_1) \wedge \cdots \wedge p_n(s_n) \wedge R \rightarrow H \wedge m_1(r_1) \wedge \cdots \wedge m_q(r_q)]$. As we discussed in Section 10 (page 36), $p(T)$ depends on $(p_1(s_1), \ldots, p_n(s_n), \neg m_1(r_1), \ldots, \neg m_q(r_q))$.
   To prove that $A(I_j)$ is acyclic, we need to prove that all its ground instances are acyclic. Consider a ground instance I'_j of $A(I_j)$ that we get after applying a substitution θ of ground terms. Suppose $I'_j = [p'_1 \wedge \cdots \wedge p'_n \wedge R' \rightarrow H' \vee m'_1 \vee \cdots \vee m'_q]$. Let p' the ground instance $p(t)\theta$. p' depends upon p'_i $(i = 1 \ldots n)$ and m'_o $(o = 1 \ldots q)$. Also, p'_i and $\neg m'_o$ are related wrt p'.
   Assume K and $\neg M$ are two (ground) literals conjoined in R' and H_i is a non-constraint atom in H'. Also, assume S is an atom that p'_i weakly depends upon - therefore S and $\neg m'_o$ are related wrt p'. Finally, assume T is an atom that K weakly depends upon and $\neg N$ is a negative literal that K depends upon. The following schema summarises the names of the literals in I'_j (where $\Downarrow$ should be read as *"(weakly) depends upon"*):[6]
   $$\begin{array}{cccccccccccc} p'_1 & \wedge \cdots \wedge & p'_i & \wedge \cdots \wedge & p'_n & \wedge & K \wedge \neg M & \rightarrow & H_i & \vee m'_1 & \vee \cdots \vee & m'_o \ldots m'_q \\ & & \Downarrow & & & & \Downarrow & & & & & \\ & & S & & & & T, \neg N & & & & & \end{array}$$
   Since I is acyclic, we have that
   - $|S| > |H_i|$, $|S| > |M|$, (because p'_i weakly depends on S),

[5]note that E is an implication.

[6]The schema is oversimplified. In fact, H may also contains other literals (and in the schema only contains H_i), and R may contain other literals (not only K and $\neg M$).

- $|S| > |m'_o|$ (because S is related to $\neg M$ wrt p'),
- $|T| > |H_i|$, $|T| > |M|$ (because K weakly depends on T),
- $|T| > |m'_o|$ (because K weakly depends on T and p'_i depends on $\neg m'_o$),
- $|S| > |N|$ (because p'_i weakly depends upon S and K depends on $\neg N$).

Given that the implication $R \to H$ is acyclic, the above formulas show that the arbitrary I'_j is acyclic. Therefore, $A(I_j)$ is acyclic.

(b) Unfolding an atomic conjunct $L' \in R_k$ in a node N_k may introduce a negative literal $\neg L$. If L is a constraint or an expectation, it cannot change the acyclicity. If L is a defined literal, we get (after logical simplification) the implication $I = [L \to false]$ in the node N_{k+1}, which is already in analysed form ($A(I) = I$). $\neg L$ is in the body of some clause in the *SOKB*, thus, based on the definition of acyclicity (item 2 of Definition 28), $A(I)$ is acyclic.

Furthermore, the unfolding of L may introduce in node N_{k+1} a constraint. By applying constraint solving, some implication E in $PSIC_k$ may be changed to $E\theta$, where θ is a substitution. If $A(E)$ is acyclic, then all its ground instances are acyclic. It is easy to see that the set of ground instances of $A(E\theta)$ is a subset of the set of ground instances of $A(E)$, therefore $A(E\theta)$ is also acyclic.

2. Executing a propagation step using an atom $L = p(t)$ and an implication $I = [p(s) \wedge R \to H]$ in a node $N_k = [L \wedge I \wedge Rest]$ produces the node $N_{k+1} = [L \wedge I \wedge I' \wedge Rest]$, where $I' = [t = s \wedge R \to H]$. Since case analysis has higher priority than propagation, we have that $I = A(I)$. Since I is acyclic, $R \to H$ is also acyclic, therefore every instance of it is acyclic. $A(I')$ is an instance of $R \to H$, therefore it is acyclic.

3. Splitting the head of an implication $I = [R \to H \vee A]$ in a node N_k results in two nodes, none of which contains I. One of the two nodes contains the atomic conjunct A and the other contains the implication $I' = [R \to H]$. Since case analysis has higher priority than splitting the head of an implication, we have that $I = A(I)$. It is therefore easy to see that $I' = A(I')$. Obviously, since I is acyclic, $A(I')$ is also acyclic.

   If A is a constraint, some implication E may change to $E\theta$. As we have already discussed in (1b), as $A(E)$ is acyclic, $A(E\theta)$ is also acyclic.

4. Applying case analysis on an equality in the body of an implication $I = [c \wedge R \to H]$ in node $N = [I \wedge Rest]$ results in two nodes, none of which contains I. One of the two nodes contains the denial $\neg c$ (which is trivially acyclic), and the other contains the implication $I' = R' \to H'$, where R', H' are R, H possibly after the application of the substitution θ for the generated by propagating the constraint c.[7]. It is easy to see that $A(I) = A(I\theta)$, therefore since $A(I)$ is acyclic, $A(I\theta)$ is also acyclic.

   The substitution θ may change some implication E in $PSIC$ to $E\theta$, but since $A(E)$ is acyclic, $A(E\theta)$ is also acyclic as argued earlier in (1b).

5. Applying factoring in node $N_k = [p(t) \wedge p(s) \wedge Rest]$ does not introduce new implications in $\mathcal{S}$CIFF (differently from the IFF), but only new constraints.

[7]In case c is an equality $t = s$, θ corresponds to equality rewrite rule (g) for the newly introduced equality $t = s$

Constraint propagation may affect an implication E in *Rest*, but, as argued earlier in (1b), the implications produced after constraint propagation are also acyclic.

The same holds also for fulfillment and violation transitions.

6. Executing non-happening on an implication $I = [\neg\mathbf{H}(t) \land R \to H]$ in a node N_k with $PSIC_k = \{I\} \cup PSIC'$ produces the node N_{k+1} with $PSIC_{k+1} = \{I'\} \cup PSIC'$, where $I' = [t = s \land R \to H]$. Since case analysis has higher priority than non-happening, we have that $I = A(I)$. Since I is acyclic, $R \to H$ is also acyclic, therefore every instance of it is acyclic. $A(I')$ is an instance of $R \to H$, therefore it is acyclic.

For part (2), by the definition of $F(I,N)$, J is in analysed form, i.e. $J = A(J)$. Also, it is easy to see that J may be introduced in some later node, after a number of unfolding and propagation steps (otherwise J would not be in $F(I,N)$). Using part (1), part (2) follows.

For part (3), let $L = p(t)$. Assume $E = [EQ \land R \to H]$. Then, J must have the form $J = [p(s) \land R \to H]$. Note that since J is in $F(I,N)$, J is in analysed form (by the definition of family of implications) and is also acyclic (based on part (2)).

$A(E)$ has the form $A(E) = [R\theta \to H\theta]$, where θ is such that $p(t)\theta = p(s)\theta$. From the definition of the level mapping on predicates, we have $|p(t)| \geq |p(t)\theta|$, therefore $|p(t)| \geq |p(s)\theta|$.

Since J is acyclic, every instance of it is acyclic, therefore $J\theta$ is acyclic. Therefore, $|p(s)\theta|$ is greater than anything that can be propagated from $J\theta$, i.e. $|p(s)\theta| > |J\theta|$. It is easy to see that $|J\theta| \geq |A(E)|$. By definition, $|E| = |A(E)|$, therefore $|J\theta| \geq |E|$.

We conclude that $|p(t)| \geq |p(s)\theta| > |J\theta| \geq |E|$, i.e., $|L| > |E|$.

□

11.5 Comparing levels of successive nodes

We compare now the level of two successive nodes. We prove the following lemma (adapted from [10, Lemma 4.3.5])

Lemma 18. *Consider a node $|N_k|$ in the execution tree T and an execution step E performed on N_k that produces the node $|N|_{k+1}$. If E is a non-introducing step, then $|N_k| \geq |N_{k+1}|$. Else, if E is an introducing step, then $|N_k| > |N_{k+1}|$.*

Proof. We consider all possible execution steps:

1. Unfolding.

   (a) *Unfolding an atomic conjunct $L \in R_k$* in a node $N_k = [L \land Rest]$ we get the (possibly not simple) node N'. Suppose that the maximum level of the simple nodes that can be derived from N' is given by the simple node N_{k+1} ($|N_{k+1}| = |N'|$), which has the form $N_{k+1} = CS \land L_1 \land \cdots \land L_n \land [M_1 \to false] \land \cdots \land [M_k \to false] \land Rest'$ ($n \geq 0$, $k \geq 0$). L depends (w.r.t. *SOKB*) on $L_1, \ldots, L_n, \neg M_1, \ldots, \neg M_k$. As *SOKB* is acyclic, $|L| > |L_i|$, $i = 1, \ldots, n$ and $|L| > |M_o|$, $o = 1, \ldots, k$.

   $|N_k|$ contains $|L|$, while $|N_{k+1}|$ does not. However, $|N_{k+1}|$ may contain some new elements (other than $|L_i|$). We prove that $|L|$ dominates all new elements.

   In the level of a node, we have the families of the implications wrt the atoms in the node itself. Since R changes, the families change as well. The introduction of the atom L_i may cause the introduction of the level of a conjunct E' to $|N_{k+1}|$, where $E' \in F(E, N_{k+1})$ and E is the implication produced by a propagation step involving L_i and an implication (existing or potential) of N_{k+1}. Based on Lemma 17, $|L_i| > |E|$ and $|E| \geq |E'|$, we have $|L| > |E'|$.

Consider the implication $I = [M_o \rightarrow false]$. M_o depends wrt *SOKB* on every element $A \in PA(I)$, therefore, as *SOKB* is acyclic, $|M_o| > |A|$, i.e., $|M_o| > |I|$. Also, $|I| \geq |J|$ for any $J \in F(I, N_{k+1})$. Since $|L| > |M_o|$ and $|I| \geq |J|$, we have $|L| > |J|$.

The rest of the node N_{k+1}, $Rest'$, is the conjunction of literals *Rest* after applying constraint solving on the constraints in CS. As we observer earlier, $Rest'$ is *Rest* with a substitution. Consider an atom $p(u) \in Rest$. Then, there is an atom $p(v) \in Rest'$ where $v = u\theta$. Based on the definition of the level mapping on predicates, $|p(u)| \geq |p(v)|$. This argument holds for all atomic conjuncts and all atoms in all implications in *Rest*. Therefore, for any conjunct C' in $Rest'$, there exists a conjunct C in *Rest* such that $|C| > |C'|$ (the same holds for potential conjuncts).

We conclude that $|N_k| > |N_{k+1}|$ if a non-constraint literal is introduced, else $|N_k| \geq |N_{k+1}|$.

(b) *Unfolding an atom L in the body of an implication $I = [L \wedge R \rightarrow H]$ in a node $N_k = [I \wedge Rest]$* we get the node $N_{k+1} = [I_1 \wedge \cdots \wedge I_r \wedge Rest$, $r > 0$.

It is easy to see that $[F(I_1, N_{k+1}) \uplus \cdots \uplus F(I_r, N_{k+1})] \subset F(I, N_k)$ (I does not occur in N_{k+1}).

After constraint solving and logical simplification, one of the introduced implications, e.g. I_i, $1 \leq i \leq r$, may be rewritten as an atomic conjunct.

Suppose the introduced atom is not a constraint. Then, $|N_k|$ contains two occurrences of $|I_i|$ while $|N_{k+1}|$ contains only one. The second occurrence of $|I_i|$ in $|N_k|$ dominates all elements introduced in $|N_{k+1}|$ because of the presence of $|I_i|$ in N_{k+1}, therefore $|N_k| > |N_{k+1}|$.

If all introduced atomic conjuncts are constraints, then $|N_k| \geq |N_{k+1}|$, as argued in (1a).

If all atomic conjuncts in N_{k+1} are also in N_k, then we have that $Pt(N_{k+1}) \subset Pt(N_k)$. Therefore, $|N_k| \geq |N_{k+1}|$.

2. Propagation

(a) Propagation that introduces implication

Executing a propagation step using an atom $L = p(t)$ and an implication $I = [p(s) \wedge R \rightarrow H]$ in a node $N_k = [p(t) \wedge I \wedge Rest]$ produces the node $N_{k+1} = [p(t) \wedge I \wedge J \wedge Rest]$, where $J = [EQ \wedge R \rightarrow H]$.

It is easy to see that $(F(I, N_k) \uplus F(J, N_{k+1})) \subseteq F(I, N_k)$. Since all atomic conjuncts in N_{k+1} are also in N_k and implications in *Rest* are not affected, we have that $Pt(N_{k+1}) \subseteq Pt(N_k)$. Therefore $|N_k| \geq |N_{k+1}|$.

(b) Propagation that introduces an atomic conjunct.

Executing a propagation step using an atom $L = p(t)$ and an implication $I = [p(s) \rightarrow H_1 \vee \cdots \vee H_m]$ in a node $N_k = [p(t) \wedge I \wedge Rest]$ produces the node $N_{k+1} = [p(t) \wedge I \wedge J \wedge Rest]$, where $J = [t = s \rightarrow H_1 \vee \cdots \vee H_m]$, and (after constraint solving and logical simplification) we get the (possibly) non simple node $N_{k+1} = [p(t) \wedge I \wedge (H_1' \vee \cdots \vee H_m') \wedge Rest]$. Suppose that the maximum level of the simple nodes that can be derived from N_{k+1} is given by the simple node N' (i.e., $|N_{k+1}| = |N'|$), which has the form $N' = [p(t') \wedge I' \wedge H'_i \wedge Rest']$, $1 \leq i \leq m$.

In the special case that all new atomic conjuncts in H'_i are constraints, N' has the form $N' = [p(t)\theta \wedge I\theta \wedge H'_i \wedge Rest']$. There are no new potential implications caused by H'_i and for any conjunct C in N' there is a conjunct in N_k that has a level that is at least equal to $|C|$ - as explained in (1a) (the same holds for potential conjuncts).

We conclude that $|N_k| \geq |N_{k+1}|$.

Suppose that some H'_i is not an equality. It is easy to see that $F(I, N_{k+1}) \subset F(I, N_k)$. In more detail, $F(I, N_k)$ contains $A(J)$, while $F(I, N_{k+1})$ does not, because the same propagation step cannot be performed again. Therefore $|N_k|$ has two occurrences of $|J|$ that $|N_{k+1}|$ does not. By the definition of the level of an implication, we have that $|J| \geq |H'_i|$. $|N_{k+1}|$ has however some elements that $|N_k|$ does not have. $|N_{k+1}|$ has one occurrence of $|H'_i|$, as well as the levels of the conjuncts E', where $E' \in F(E, N_{k+1})$ and E is the result of propagating H'_i and any (existing or potential) implication in N_{k+1}. As argued in (1a), $|H'_i| > |E'|$. Therefore, the second occurrence of $|J|$ in $|N_k|$ dominates all the elements in $|N_{k+1}|$ that are not in $|N_k|$.

We conclude that $|N_k| > |N_{k+1}|$.

3. Splitting a disjunction. By definition of level of a non simple node.

4. Case analysis.

   Consider the implication $I = [C \wedge R \rightarrow H]$ containing the constraint C in a node $N_k = [I \wedge Rest]$. Case analysis produces two nodes; $N^1_{k+1} = [\neg C \wedge Rest']$ and $N^2_{k+1} = [C \wedge J \wedge Rest'']$, where $J = [R' \rightarrow H']$ (R' and H' are R, H after constraint solving and logical simplification have taken place). Obviously, $|N_k| \geq |N^1_{k+1}|$.

   Consider N^2_{k+1}. We distinguish two cases:

   - if R' is not empty (true), then no atom is introduced in N^2_{k+1}. ($A(I) = A(J)$, therefore) $F(I, N_k) = F(J, N_{k+1})$. Therefore, $|N_k| = |N^2_{k+1}|$
   - if R' is empty (true), then J is an atomic conjunct of N^2_{k+1}.

     If J is a constraint, then $|N_k| \geq |N^2_{k+1}|$, as argued earlier in (1a).

     If J is not a constraint, then $|N_k|$ contains two occurrences of $|I|$, while $|N^2_{k+1}|$ contains only one occurrence of $|H'|$ and the levels of the new potential implications E caused by H'. Since $|I| \geq |H'|$ and $|H'| > |E|$ (by Lemma 17), we have that $|N_k| > |N^2_{k+1}|$.

5. Factoring.

   Consider a node $N_k = [p(t) \wedge p(s) \wedge Rest]$, where p is abducible (i.e., either **E** or **NE**). Then, by factoring we get the two nodes $N^1_{k+1} = [p(t) \wedge p(s) \wedge t \neq s \wedge Rest']$ and $N^1_{k+1} = [p(t) \wedge t = s \wedge Rest'']$.

   As we have seen, for example, in (1a), adding a constraint cannot increase the level of a node, thus $|N_k| \geq |N_{k+1+}|$. Also, $|N_k| > |N_{k+1}|$ because an atom was 'lost' (and, again, the introduction of a constraint can only reduce the level of a node).

6. Fulfillment and Violation.

All these transitions are non-introducing steps. We can apply the same considerations done for Factoring also for Fulfillment **E** and Violation **NE**. Violation **E** leads to a failure node. Fulfillment **NE** does not change the level of the node, so $|N_k| = |N_{k+1}|$.

□

12 Static-$\mathcal{S}$CIFF Proof of termination

Relying on the previous lemmas, we are now able to prove Theorem 5 of termination of the static $\mathcal{S}$CIFF, which is identical to the proof of the corresponding theorem by Xanthakos.

Proof. In Lemma 18, we have proved that any execution step that introduces a non-constraint atom strictly reduces the level of the node that the step was performed on, and that all other steps cannot increase that level. Thus, based on Lemma 15, we conclude that any node N in an infinite chain of execution steps must be followed (after a finite number of steps) by a node N' such that $|N| > |N'|$.

Having started from a bounded node and remembering that the multiset ordering we use is well-founded, we conclude that there cannot be an infinite sequence of execution steps, i.e. any branch in the execution tree is finite. Therefore, as the maximum branching degree of any execution tree is finite (namely 2) we conclude that the execution tree is finite. □

13 Extension for dynamically growing history

We are now able to prove the complete termination theorem for $\mathcal{S}$CIFF. We will make two further assumptions. The first states that the new events will arrive only when the $\mathcal{S}$CIFF is in a stable state (i.e., new events are considered only if no other transition is applicable).

Definition 38. *A $\mathcal{S}$CIFF derivation has a* slow happening rate *if happening transitions apply only if no other transition is applicable.*

Non happening transitions are applicable only after closure of the history [4]. We will assume that after closure of the history, non happening is applied as soon as possible (this can be seen as a *preprocessing*):

Definition 39. *A $\mathcal{S}$CIFF derivation has* non happening high priority *if, whenever non happening is applicable, it is indeed applied.*

We can now state and prove our termination theorem for $\mathcal{S}$CIFF.

Theorem 6. (Termination of $\mathcal{S}$CIFF) *Let G be a query to a society $\mathcal{S}$, where $SOKB$, $\mathcal{IC}_S$ and G are acyclic wrt some level mapping, and G and all implications in $\mathcal{IC}_S$ bounded wrt the level-mapping.*

Then, every $\mathcal{S}$CIFF derivation with high priority for non happening and with slow happening rate for G, starting from an initial history $\mathbf{HAP}^i$ ending in a (possibly closed) finite final history $\mathbf{HAP}^f$ is finite.

Proof. We now add the transitions that were removed in the previous proof of Theorem 5, namely Happening, non-happening, and closure.

We divide the whole derivation in two phases: the open and the closed phases. The closed phase is the sub-derivation starting from the first node with a closed history. From the definition of closure transition, there can be only one closure transition that ends in a closed history.

In the open phase, happening can be applied. Since the derivation has slow happening rate, the static $\mathcal{S}$CIFF is applied between any two happening transitions. Since the final history $\mathbf{HAP}^f$ is finite, happening can be applied finitely many times. No transition can cause happening. Between two happening transitions, static $\mathcal{S}$CIFF is applied, and by theorem 5 it terminates. When the static $\mathcal{S}$CIFF terminates, closure transition becomes applicable, and it is applicable only once (before another happening transition), so closure will be applied finitely many times.

Non-happening can be applied only to implications with $\neg\mathbf{H}$, and it removes the $\neg\mathbf{H}$ literal. So, since implications are finitely many, and have finite number of literals, non-happening is applied finitely many times. No transition can cause happening. Between one happening and the other, $\mathcal{S}$CIFF is applied, and it terminates. After application of non-happening transitions, static $\mathcal{S}$CIFFis applicable, and by previous theorem, it terminates. $\square$

Index

References

[1] T. H. Fung. *Abduction by Deduction.* PhD thesis, Imperial College London, 1996.

[2] T. H. Fung and R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, Nov. 1997.

[3] M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. SCIFF: Full proof of soundness. Deliverable IST32530/DIFERRARA/401/D/I/b1, SOCS Consortium, Jun 2004.

[4] M. Gavanelli, E. Lamma, P. Torroni, P. Mello, K. Stathis, P. Moraïtis, A. C. Kakas, N. Demetriou, G. Terreni, P. Mancarella, A. Bracciali, F. Toni, F. Sadri, and U. Endriss. Computational model for computees and societies of computees. Technical report, SOCS Consortium, 2003. Deliverable D8.

[5] J. Jaffar and M. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19-20:503–582, 1994.

[6] J. Jaffar, M. Maher, K. Marriott, and P. Stuckey. The semantics of constraint logic programs. *Journal of Logic Programming*, 37(1-3):1–46, 1998.

[7] K. Kunen. Negation in logic programming. In *Journal of Logic Programming*, volume 4, pages 289–308, 1987.

[8] P. Mello, P. Torroni, M. Gavanelli, M. Alberti, A. Ciampolini, M. Milano, A. Roli, E. Lamma, F. Riguzzi, and N. Maudet. A logic-based approach to model interaction amongst computees. Technical report, SOCS Consortium, 2003. Deliverable D5.

[9] P. Stuckey. Negation and constraint logic programming. *Information and Computation*, 118(1):12–33, 1995.

[10] I. Xanthakos. *Semantic Integration of Information by Abduction.* PhD thesis, Imperial College London, 2003.