SOCS

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

IST-2001-32530

Deliverable D13: Verifiable Properties of Societies of Computees

Project number:	IST-2001-32530
Project acronym:	SOCS
Document type:	D (deliverable)
Document distribution:	PP (restricted to the GC Programme)
CEC Document number:	IST32530/ICSTM/057/D/PP/b1
File name:	1057-b1[d13].pdf
Editor:	U. Endriss and F. Sadri
Contributing partners:	ALL
Contributing workpackages:	WP5
Estimated person months:	45
Date of completion:	10 March 2005
Date of delivery to the EC:	18 March 2005
Number of pages:	76

ABSTRACT
This deliverable reports the work conducted within the SOCS project for workpackage WP5 on verifiable properties of computees and their societies.

Deliverable D13: Verifiable Properties of Societies of Computees

M. Alberti,[6] F. Athienitou,[4] A. Bracciali,[2] F. Chesani,[5] U. Endriss,[1] M. Gavanelli,[6] A. Kakas,[4] E. Lamma,[6] W. Lu,[3] P. Mancarella,[2] P. Mello,[5] F. Sadri,[1] K. Stathis,[3] F. Toni,[1] and P. Torroni[5]

[1]Department of Computing, Imperial College London
Email: {ue,fs,ft}@doc.ic.ac.uk

[2]Dipartimento di Informatica, Università di Pisa
Email: {braccia,paolo}@di.unipi.it

[3]Department of Computing, City University, London
Email: {lue,stathis}@soi.city.ac.uk

[4]Department of Computer Science, University of Cyprus
Email: {cspgat1,antonis}@cs.ucy.ac.cy

[5]DEIS, Università di Bologna
Email: {fchesani,pmello,ptorroni}@deis.unibo.it

[6]Dipartimento di Ingegneria, Università di Ferrara
Email: {malberti,mgavanelli,elamma}@ing.unife.it

ABSTRACT
This deliverable reports the work conducted within the SOCS project for workpackage WP5 on verifiable properties of computees and their societies.

Contents

1 Introduction

This deliverable reports the work conducted within the SOCS project for workpackage WP5 on verifiable properties of computees and their societies. In the first year of the project we developed declarative models for computees and societies of computees. In the second year we developed the respective computational models and individual and integrated prototype implementations for them. In the third year we have formalised and proved various properties of the models (in WP5), and conducted practical experiments, with the implemented models, based on several scenarios (in WP6). A companion document, deliverable D14 [2], reports the work on these experiments.

The models of computees and societies of computees are both based on computational logic. Even the control module of the computees is formalised as computational logical theories, more specifically as logic programs with priorities. These design principles have already paid dividends in the development of the computational models and the proximity of the prototype implementations to the declarative and computational models. A further advantage of the use of computational logic is that it ensures that the models, both the declarative and computational models, lend themselves well to the specification and verification of formal properties. This is the topic for WP5 and this deliverable.

1.1 Choice of Properties

The choice of the properties to consider for WP5 has been guided by the research area of Global Computing. This choice has been deliberated and refined incrementally and progressively within the SOCS consortium, as reported in the evaluation criteria given in deliverable D3 [50], the Provisional List of Properties [30] and the progress report in deliverable D12 [3]. The final choice is aimed at the following main objectives:

(O1) *Effectiveness:* To show the effectiveness of our computational logic approach in modelling computees and their societies, in the sense of facilitating formalisation of properties and prediction of behaviour without the need to resort to empirical methods.

(O2) *Consequences:* To explore the consequences of some of the detailed design choices: The computee model is based on a modular collection of capabilities and transitions integrated within cycle theories for control, incorporating various selection functions. The society model is based on a social infrastructure that incorporates social knowledge and protocols. In the detailed designs of both computees and societies many individual choices have had to be made. One of our objectives in WP5 has been to formally explore the consequences of these choices. This part of our investigations has aimed at:

- Showing the appropriateness of the design decisions: for example some properties show the soundness of the proof procedures developed for computees and societies, and others show some benefits for example in terms of computees' behaviour towards their goals and plans that result directly as a consequence of the design choices.
- Identifying any shortcomings or errors in the detailed design.

(O3) *Versatility:* To explore the scope and versatility of the computee and society models: for example we have investigated how we can specify different profiles of behaviour in computees and how such profiles could alter the behaviour of computees and their success in achieving their goals.

(O4) *Guidance:* To give guidelines to developers who may use the PROSOCS platform (PROSOCS [4] is the implemented platform for computees and societies): for example formal results that show the relative merits of different behaviour profiles with respect to parameters characterising features of an application domain or environment can provide useful guidelines for those who wish to develop applications using PROSOCS.

(O5) *Scenarios:* To explore aspects of the chosen scenarios: for example we have studied various aspects of protocol conformance and allocation of resources.

We present the properties in four categories:

- Properties of Proof Procedures
- Properties of Individual Computees
- Properties of the Social Infrastructure
- Properties Related to Protocol Conformance

Each property in these four categories addresses at least one, and more often than not, several of the objectives O1–O5. For each property we give its informal description, significance, brief formal description, approach taken in proving it and the related work, if any. Detailed descriptions of the properties and proofs are given in the reports included in the annex and various published documents (see page 76).

Apart from the properties shown in this deliverable there are many features of the SOCS computee and society models that make them particularly suited to open and dynamic environments of the kind the Global Computing initiative addresses. These features require no proofs and become evident from an examination of the models. For example,

- *Heterogeneity:* This is possible at almost every level of the models, starting from the individual computee knowledge bases, priorities, and control strategies, to interaction protocols and society knowledge bases.
- *Adaptability:* The control strategies of computees' "normal" behaviour is designed in such a way that the computee can be interrupted from its "normal" activities to make observations in the environment and record them in its knowledge base. It then uses this information to make any necessary adjustments to its goals and plans in order to adapt to the perceived changes in the environment. Such changes include receiving messages from other computees.
- *Openness:* The social infrastructure caters for open societies where there are no restrictions on which or how many societies a computee joins.

The work on WP5 is complemented by that of WP6, which aims at conducting practical experiments with the PROSOCS platform and the implemented models of computees and societies of computees. The experiments are designed to explore a number of issues:

- To test the formal definitions of some of the componments of the models, for example experiments are conducted in testing some of the behaviour profiles (described in Section 5.1 in this deliverable) and the computee planning capability and transition (described in detail in deliverable D8 [36]).

- To explore empirically some of the properties of the behaviour profiles.
- To explore empirically the application of the fully integrated models and the PROSOCS platform on example scenarios from the domain of combinatorial auctions.

The two workpackages WP5 and WP6 have provided valuable input to one another, each helping to enhance and refine the work conducted in the other.

1.2 Deliverable Overview

The rest of this deliverable is structured as follows. In Section 2 we give an overview of the declarative and computational models of individual computees and societies of computees. In Section 3 we outline and classify the properties that are addressed in D13. We also explain the uniform format that is used for the summaries given for each property in the rest of the body of the deliverable. Using this format, in Sections 4, 5, 6, and 7 we outline the properties of the proof procedures, individual computees, the social infrastructure and protocol conformance, respectively. In Section 8 we conclude, and following the list of references we list all the documents that are annexed to D13.

2 Background on the SOCS Approach

In this section we briefly review the computee and the society models and proof procedures developed within SOCS. The purpose of this section is to make this deliverable as self-contained as possible. The models and the proof procedures were presented in detail in deliverables D4 [42], D5 [54] and D8 [36].

2.1 The KGP Model of Individual Computees

We summarise the main components and concepts of the KGP model of computees. Further details may be found in deliverable D8 [36].

2.1.1 Knowledge, Goals and Plan

Internal state. KGP computees have an internal state which is a tuple $\langle KB, Goals, Plan, TCS \rangle$, where:

- KB is the knowledge base of the computee, and describes what the computee believes of itself and the environment. KB consists of several modules supporting different reasoning capabilities. These modules are:
  - KB_{plan}, for Planning,
  - KB_{pre}, for the Identification of Preconditions of actions,
  - KB_{TR}, for Temporal Reasoning,
  - KB_{GD}, for Goal Decision,
  - KB_{react}, for Reactivity, and

– KB_0, for holding the (dynamic) knowledge of the computee about the external world in which it is situated (including past communications). It records, for example, observations of fluents holding or not holding in the environment at given times, and actions executed by the computee or observed to be executed by other computees at given times. KB_0 is included in all other knowledge bases.

- *Goals* is the set of objectives that the computee wants to achieve. Each goal has an associated time which can be fully instantiated or a (possibly constrained) variable.

- *Plan* is a set of actions scheduled in order to satisfy goals. Each action also has an associated time.

- TCS is a set of constraint atoms (referred to as *temporal constraints*) in some given underlying constraint language with respect to some structure $\Re$ equipped with a notion of constraint satisfaction $\models_{\Re}$. Temporal constraints bind the time variables of goals in *Goals* and actions in *Plan*, thus implicitly defining when goals are expected to hold and when actions should be executed. Via the temporal constraints, actions are partially ordered.

  In the sequel, given a set C of sentences built from constraint atoms, $\models_{\Re} C$ will stand for C is $\Re$-satisfiable, and $\not\models_{\Re} C$ will stand for C is $\Re$-unsatisfiable.

Goals and actions are uniquely identified by their associated time, which is implicitly existentially quantified within the overall state. To aid revision and partial planning, *Goals* and *Plan* form a *tree*, whose root is represented by $\bot$. *Top-level* goals and actions are children of the root of the tree.

Valuation of temporal constraints. Given a state $S = \langle KB, Goals, Plan, TCS \rangle$, we denote by $\Sigma(S)$ (or simply Σ, when S is clear from the context) the valuation:

$$\Sigma(S) = \{t = \tau \,|\, executed(a[t], \tau) \in KB_0\} \cup \{t = \tau \,|\, observed(l[t], \tau) \in KB_0\}$$

Intuitively, Σ extracts from KB_0 the instantiation of the (existentially quantified) time variables in *Plan* and *Goals*, derived from having executed (some of the) actions in *Plan* and having observed that (some of the) fluents in *Goals* hold (or do not hold). KB_0 provides a "virtual" representation of Σ.

2.1.2 Capabilities and Transitions

Capabilities. Computees have a collection of reasoning capabilities, including Goal Decision, Planning, Reactivity and Temporal Reasoning. In the computational counterpart of the formal KGP model, the last three of these have been realised using the CIFF proof procedure for abductive logic programming [24], while the Goal Decision capability builds on the LPwNF (Logic Programming without Negation as Failure) framework, implemented in the Gorgias system [20, 45]. Another important capability is the Sensing capability to check whether a given property holds in the environment.

Transitions. The state of a computee evolves by applying transition rules, which employ capabilities and constraint satisfaction. The transitions are:

- Goal Introduction (GI), replacing *Goals* with a new set of top-level goals, and using the Goal Decision capability.
- Plan Introduction (PI), changing *Goals* and *Plan*, and using the Planning and Introduction of Preconditions capabilities.
- Reactivity (RE), changing *Goals* and *Plan*, and using the Reactivity capability.
- Sensing Introduction (SI), changing *Plan* by introducing new sensing actions for checking the preconditions of actions already in *Plan*, and using the Sensing capability.
- Passive Observation Introduction (POI), changing KB_0 by introducing unsolicited information coming from the environment, and using the Sensing capability.
- Active Observation Introduction (AOI), changing KB_0, by introducing the outcome of (actively sought) sensing actions, and using the Sensing capability.
- Action Execution (AE), executing all types of actions (physical, sensing, communicative), and changing KB_0.
- State Revision (SR), revising *Goals* and *Plan*, and using Temporal Reasoning and Constraint Satisfaction.[1]

Transitions are represented as $T(S, X, S', \tau)$, where T is the name of the transition, S is the state before the transition is applied and S' is the state after, X is the (possibly empty) input taken by the transition, and τ is the time of application of the transition.

2.1.3 Cycle Theories

Transitions are integrated within cycle theories that specify the control component of computees.

Cycle theory. Formally, a cycle theory $\mathcal{T}_{cycle}$ consists of the following parts:

- An *initial* part $\mathcal{T}_{initial}$, that determines the possible transitions that the agent could perform when it starts to operate (*initial cycle step*). More concretely, $\mathcal{T}_{initial}$ consists of rules of the form

  $$*T(S_0, X) \leftarrow C(S_0, \tau, X), now(\tau),$$

  which we refer to via the "name" $\mathcal{R}_{0|T}(S_0, X)$. These rules sanction that, if the conditions C are satisfied in the initial state S_0 at the current time τ, then the initial transition should be T, applied to state S_0 and input X, if required. Parameter X may be absent if the given transition requires no input.

  The notation $*T(S, X)$ in the head of these rules, meaning that the transition T can potentially be chosen as the next transition, is used in order to avoid confusion with the notation $T(S, X, S', \tau)$, introduced earlier to represent the actual application of T.

[1] We assume here that the two transitions of Goal Revision and Plan Revision in the original KGP model presented in [42, 36, 43] are combined into a single State Revision transition (see also Section 2.3.1).

The following initial rule, for example, expresses that the first transition may be a Plan Introduction provided there are goals available to plan for:

$$\mathcal{R}_{0|PI}(S_0, Gs) : *PI(S_0, Gs) \leftarrow Gs = c_{GS}(S_0, \tau), Gs \neq \emptyset, now(\tau)$$

- A *basic* part $\mathcal{T}_{basic}$ that determines the possible transitions (*cycle steps*) following other transitions, and consists of rules of the form

  $$*T'(S', X') \leftarrow T(S, X, S', \tau), EC(S', \tau', X'), now(\tau'),$$

  which we refer to via the "name" $\mathcal{R}_{T|T'}(S', X')$. These rules sanction that, after the transition T has been executed, starting at time τ in the state S and ending at the current time τ' in the resulting state S', and the conditions EC evaluated in S' at τ' are satisfied, then transition T' could be the next transition to be applied in the state S' with the input X', if required (again, parameter X' may be absent). The conditions EC are called *enabling conditions* as they determine when a cycle-step from the transition T to the transition T' can be applied. In addition, they determine the input X' of the next transition T'. Such inputs are determined by calls to the appropriate selection functions.

  For example, the following basic rule expresses that a State Revision may be followed by an Action Execution provided that the set of actions resturned by the action selection function is not empty:

  $$\mathcal{R}_{SR|AE}(S', As) : *AE(S', As) \leftarrow SR(S, S', \tau), As = c_{AS}(S', \tau'), As \neq \emptyset, now(\tau')$$

- A *behaviour* part $\mathcal{T}_{behaviour}$ that contains rules describing dynamic priorities amongst rules in $\mathcal{T}_{basic}$. Rules in $\mathcal{T}_{behaviour}$ are of the form

  $$\mathcal{R}_{T|T'}(S, X') \succ \mathcal{R}_{T|T''}(S, X'') \leftarrow BC(S, X', X'', \tau), now(\tau)$$

  with $T' \neq T''$, which we will refer to via the "name" $\mathcal{P}^{T}_{T' \succ T''}$. These rules in $\mathcal{T}_{behaviour}$ sanction that, at the current time τ, after transition T, if the conditions BC hold, then we prefer the next transition to be T' over T'', namely doing T' has *higher priority* than doing T'', after T. The conditions BC are called *behaviour conditions*. They are *heuristic* conditions, which may be defined in terms of *heuristic selection functions* (see [42] for details). For example, the heuristic action selection function may choose those actions in the agent's plan whose time is close to running out amongst those whose time has not run out.

- An *auxiliary part* including definitions for any predicates occurring in the enabling and behaviour conditions, and in particular for selection functions (including the heuristic ones, if needed).

- An *incompatibility part*, including rules stating that all different transitions are incompatible with each other and that different calls to the same transition but with different input items are incompatible with each other.

Operational trace. Cycle theories are interpreted as logic programs with preferences and the respective entailment operator is denoted by $\models_{pr}$. A cycle theory $\mathcal{T}_{cycle}$ induces an *operational trace*, namely a (possibly infinite) sequence of transitions

$$T_1(S_0, X_1, S_1, \tau_1), \ldots, T_i(S_{i-1}, X_i, S_i, \tau_i), T_{i+1}(S_i, X_{i+1}, S_{i+1}, \tau_{i+1}), \ldots$$

(where each of the X_i may be absent), such that

- S_0 is the given initial state;
- for each $i \geq 1$, τ_i is the time of the execution of transition T_i, with $\tau_i < \tau_{i+i}$;
- (*Initial Step*) $\mathcal{T}^0_{cycle} \wedge now(\tau_1) \models_{pr} *T_1(S_0, X_1)$, where $\mathcal{T}^0_{cycle}$ is $\mathcal{T}_{cycle} - \mathcal{T}_{basic}$;
- (*Cycle Step*) for each $i \geq 1$:

  $\mathcal{T}^s_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge now(\tau_{i+1}) \models_{pr} *T_{i+1}(S_i, X_{i+1})$,

  i.e. each (non-final) transition in a sequence is followed by the most preferred transition, as specified by $\mathcal{T}^s_{cycle}$, where $\mathcal{T}^s_{cycle}$ is $\mathcal{T}_{cycle} - \mathcal{T}_{initial}$. If the most preferred transition determined by $\models_{pr}$ is not unique, we choose one arbitrarily.

We refer to a state S_i in an operational trace as $\langle KB^i, Goals^i, Plan^i, TCS^i \rangle$.

2.1.4 The CIFF Proof Procedure

The computational counterpart of the formal KGP model builds on two proof procedures: Logic Programming without Negation as Failure (LPwNF), implemented in the Gorgias system, and CIFF for abductive reasoning with constraint predicates. We only include a brief description of the latter here, to provide a reference point for the results on CIFF presented later on (see Sections 4.4 and 4.5).

CIFF extends the IFF procedure of Fung and Kowalski [33] in two ways. One is that CIFF incorporates a constraint solver to deal with constraint predicates. The second concerns allowedness. The original IFF procedure requires inputs to meet a number of allowedness conditions to be able to guarantee the correct operation of the procedure. For CIFF, our approach is to tackle the issue of allowedness *dynamically*, i.e. at runtime, rather than adopting a static and overly strict set of conditions. To this end, the CIFF procedure includes a *dynamic allowedness rule* which is triggered whenever the procedure encounters a particular formula it cannot manipulate correctly due to a problematic quantification pattern. If the dynamic allowedness rule detects a formula with a quantification pattern that cannot be handled by the other proof rules then that node is labelled as *undefined*. This ensures (i) that CIFF will never compute an incorrect answer, and (ii) that an answer different from *undefined* is given whenever possible (in comparison, the overly strict static allowedness condition given by Fung and Kowalski means that their procedure is not defined over some types of input where computing an answer would still be possible). Our approach also allows us to run the procedure without first checking whether or not the input meets some suitable allowedness conditions (in comparison, the original IFF procedure could return an incorrect answer in such a case).

The input to the CIFF procedure consists of a (selectively completed) theory *Th*, a set of integrity constraints *IC*, and a query Q. There are three possible outputs: (1) the procedure succeeds and indicates an (abductive) answer to the query Q; (2) the procedure fails, thereby indicating that there is no answer; and (3) the procedure reports that computing an answer is not possible, because a critical part of the input is not allowed.

The CIFF procedure manipulates a set of formulas that are either atoms, implications, or disjunctions of atoms and implications. The theory *Th* is kept in the background and is only used to *unfold* defined predicates as they are encountered. Whenever a disjunction is encountered, the *splitting* rule may be applied, i.e. disjunctions give rise to different branches in the proof search tree. A node containing $\bot$ is called a *failure node*. If all branches in a derivation terminate with failure nodes, then the derivation is said to fail (the intuition being

that there exists no answer to the query in question). A node to which no more proof rules can be applied is called a *final node*. A final node that is not a failure node and which has not been labelled as *undefined* is called a *success node*. An *extracted answer* for a final success node N is a triple $\langle \Delta, \Phi, \Gamma \rangle$, where Δ is the set of abducible atoms, Φ the set of equalities and disequalities, and Γ the set of constraint atoms in N. Full details are given in [24].

2.2 The Society Model

We provide here a summary of the main features of the declarative and operational semantics of the society. Our description is a synthesis of the relevant parts of deliverable D8 [36].

2.2.1 Syntax

The society knowledge consists of the 4-tuple $\langle SOKB, SEKB, \mathcal{IC}_S, \mathcal{G} \rangle$, where:

- $SOKB$ is the *Social Organisation 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 *Goal* of the society.

Social environment knowledge base. The $SEKB$ dynamically evolves and consists of:

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

The happened events are represented as ground atoms of the form **H**(*Event*[, *Time*]), while expectations can be either of the form **E**(*Event*[, *Time*]) or of the form **EN**(*Event*[, *Time*]). Expectations can contain variables, with the following interpretation:

- variables in **E** atoms are always existentially quantified with scope the entire set of expectations;
- the other variables that occur only in **EN** atoms are universally quantified.

Social organisation knowledge base. The SOKB is a logic program, consisting of clauses that can contain expectations and CLP constraints [41].

Goal. The goal $\mathcal{G}$ of the society is a conjunction of expectations, CLP constraints and literals of predicates defined in the SOKB.

Social integrity constraints. These have the form of implications and can involve predicates defined in the SOKB, happened events, expectations, and CLP constraints. These integrity constraints can be used to describe social protocols of interaction.

2.2.2 ALP Interpretation of the Society Model

A society instance, i.e. a society grounded on a given history (a set of happened events), has an abductive interpretation, presented in previous deliverables [54, 36].

Definition 2.1 (Society instance) *An* instance $\mathcal{S}_{\mathbf{HAP}}$ *of a society* $\mathcal{S}$ *is represented as an Abductive Logic Program, 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* $\mathcal{S}$ *(the* $\mathbf{E}$ *and* $\mathbf{EN}$ *atoms);*
- $\mathcal{IC}_S$ *are the social integrity constraints of* $\mathcal{S}$*.*

Definition 2.2 (Society extension) *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 2.3 (Closed instance) *Given an instance* $\mathcal{S}_{\mathbf{HAP}}$ *of a society* $\mathcal{S}$*, the instance is* closed *iff it is assumed that no more events can happen. We denote a closed instance as* $\mathcal{S}_{\overline{\mathbf{HAP}}}$*.*

In the following, we indicate a closed history by means of an overline: $\overline{\mathbf{HAP}}$. In a closed instance, we assume that no further event can occur.

2.2.3 Declarative Semantics

The semantics of a society instance is given by identifying sets of expectations which, together with the society's knowledge base and the happened events, imply an instance of the goal and *satisfy* the integrity constraints. We rely upon a notion of entailment in a three-valued logic. In particular, for open instances we refer to a three-valued completion where only the history of events has not been completed. For closed instances the history of events is completed as well.

Definition 2.4 (Admissibility) *Given a (closed/open) society instance* $\mathcal{S}_{\mathbf{HAP}}$*, a set of expectations* $\mathbf{EXP}^\star$ *is called*

- $\mathcal{IC}_S$*-consistent iff* $SOKB \cup \mathbf{HAP} \cup \mathbf{EXP}^\star \models \mathcal{IC}_S$
- *E-consistent iff for each (ground) term* p*,* $\{\mathbf{E}(p), \mathbf{EN}(p)\} \not\subseteq \mathbf{EXP}^\star$
- $\neg$*-consistent iff for each (ground) term* p*,* $\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq \mathbf{EXP}^\star$ *and* $\{\mathbf{EN}(p), \neg\mathbf{EN}(p)\} \not\subseteq \mathbf{EXP}^\star$

Given a closed (respectively, open) society instance, a set of expectations is called closed *(resp.* open*)* admissible *if it is* $\mathcal{IC}_S$*-, E- and* $\neg$*-consistent.*

Definition 2.5 (Fulfilment) *Given a (closed/open) society instance* $\mathcal{S}_{\mathbf{HAP}}$*, a set of social expectations* $\mathbf{EXP}^\star$ *is* fulfilled *if and only if for each (ground) term* p*:*

$$\mathbf{HAP} \cup \mathbf{EXP}^\star \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{EN}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models false$$

Definition 2.6 (Violation) *Given a (closed/open) society instance* $\mathcal{S}_{\mathbf{HAP}}$*, a set of social expectations* $\mathbf{EXP}^\star$ *is* violated *if it is not fulfilled.*

Definition 2.7 (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 $\mathcal{S}_{\mathbf{HAP}} \not\approx_{\mathbf{EXP}^\star} G$) iff there exists an (open) admissible and fulfilled set of social expectations $\mathbf{EXP}^\star$, such that:*

$$SOKB \cup \mathbf{HAP} \cup \mathbf{EXP}^\star \models G$$

Definition 2.8 (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 $\mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^\star} G$) iff there exists a (closed) admissible and fulfilled set of social expectations $\mathbf{EXP}^\star$, such that:*

$$SOKB \cup \overline{\mathbf{HAP}} \cup \mathbf{EXP}^\star \models G$$

2.2.4 The Society Proof Procedure $\mathcal{S}$CIFF

The $\mathcal{S}$CIFF proof procedure [36], another extension of the IFF procedure [33] developed within WP3, 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 and 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

The set of social expectations, $\mathbf{EXP}^\star$, is partitioned into the sets **EXP**, **FULF**, and **VIOL** (i.e., $\mathbf{EXP}^\star = \mathbf{EXP} \cup \mathbf{FULF} \cup \mathbf{VIOL}$).

Initial node and success. A derivation D is a sequence of nodes

$$T_0 \rightarrow T_1 \rightarrow \cdots \rightarrow T_{n-1} \rightarrow T_n,$$

obtained as follows. Given a goal G and a set of integrity constraints $\mathcal{IC}_S$, we build the first node as follows:

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

The other nodes are obtained by applying the transitions defined in deliverable D8 [36], until no further transition can be applied (*quiescence*).

Definition 2.9 (Open successful derivation) *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 CS *is consistent. In that case, we write:*

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

Definition 2.10 (Closed successful derivation) *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* $\mathbf{EXP}$ *contains only negative literals* $\neg\mathbf{E}$ *and* $\neg\mathbf{EN}$*. In such a case, we write:*

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

2.3 Updates

In this section, we report on two recent updates to the formal and computational models presented in previous deliverables.

2.3.1 The New State Revision Transition

We have merged the revision transitions of the KGP model, i.e. Goal Revision (GR) and Plan Revision (PR), into a new single revision transition SR. The reason for this is that it became clear very quickly (and through the investigation of behaviour profiles and the experimentation) that it is always preferable to apply one revision transition (GR or PR) immediately after the other.

Informally speaking, the new transition SR revises a state by removing all timed-out goals and actions and all goals and actions that have become obsolete because one of their ancestors is already believed to have been achieved. The specification of the new transition is as follows.

$$(\mathbf{SR}) \qquad \frac{\langle KB, Goals, Plan, TCS\rangle}{\langle KB, Goals', Plan', TCS'\rangle}\,\tau$$

where $Goals' \cup Plan'$ is the biggest subset of $Goals \cup Plan$ consisting of all goals $G = \langle l[t], P\rangle \in Goals$ and actions $A = \langle a[t'], P', C\rangle \in Plan$ such that:

(i) $P \in Goals' \cup Plan' \cup \{\bot\}$ and $P' \in Goals' \cup Plan' \cup \{\bot\}$, and

(ii) there exists a total valuation σ such that $\sigma \models_{\Re} TCS' \wedge t > \tau$, and there exists a total valuation σ such that $\sigma \models_{\Re} TCS' \wedge t' > \tau$, and

(iii) it is not the case that $executed(a[t'], \tau') \in KB_0$, and

(iv) there is no total valuation σ such that $\sigma \models TCS' \wedge t \leq \tau$ and $KB \models_{TR} l[t]\sigma$, and

(v) for every sibling $G' = \langle l'[t'], p' \rangle$ of G (or A) in $Goals \cup Plan$, either G' is a sibling of G (or A) in $Goals' \cup Plan'$ or there is a total evaluation σ such that $\sigma \models TCS \wedge t' \leq \tau$ and $KB \models_{TR} l'[t']\sigma$.

Condition (ii) removes *timed-out* goals and actions, and (iv) removes goals that are already achieved, (iii) removes actions that have already been executed, (v) removes goals and actions whose siblings are already timed out (and thus deleted, by condition (ii)), and (i) removes actions and goals with ancestors which are got rid off (recursively). Conditions (i)–(iv) are directly borrowed from the original definitions of GR and PR, whereas condition (v) is novel.

2.3.2 Correction to the Action Selection Function

While investigating the Coherence Properties (see Section 5.2), we came across an error in the definition of the Action Selection function, given earlier in deliverable D8 [36]. Below we describe the error and correct the definition.

Specification of c_{AS}

Informally, the set of conditions for the core action selection function is as follows. Given a state $S = \langle KB, Goals, Plan, TCS \rangle$ and a time-point τ, the set of all actions selected by c_{AS} is defined as follows. Let $\mathcal{X}(S, \tau)$ be the set of all actions A in $Plan$ such that:

1. A is executable at τ, e.g. it is not timed out,
2. no ancestor or sibling of A in $Goals$ and $Plan$ is timed out at τ,
3. no ancestor of A in $Goals$ is already satisfied at τ, given S,
4. no precondition of A is known to be false at τ, given S,
5. A has not already been executed.

Then it should be the case that $c_{AS}(S, \tau) \subseteq \mathcal{X}(S, \tau)$ such that all actions in $c_{AS}(S, \tau)$ are executable concurrently at τ.

The error in D8 in the definition of action selection was that $c_{AS}(S, \tau) = \mathcal{X}(S, \tau)$ and thus the action selection function could return sets of actions executable in isolation but mutually incompatible. Note that condition 1 in the definition of $\mathcal{X}(S, \tau)$ is logically redundant, as it is also re-imposed by definition of $c_{AS}(S, \tau)$. However, this condition serves as a first filter, and is thus useful in practice.

We correct the definition of action selection as follows. We first give a formalisation of the 5 earlier conditions and then correct the definition of c_{AS}.

Formally, given a state $S = \langle KB, Goals, Plan, TCS \rangle$, and a time-point τ, the set of all actions selected by c_{AS} is defined as follows. Each action A in $Plan$ is represented as $\langle a[t], G, C \rangle$ where a is the action operator, t is the time associated with the action, G is the immediate goal for which the action has been planned (G is the parent of A), and C is ttheset of all the preconditions of A. Let $\mathcal{X}(S, \tau)$ be the set of all actions

$$A = \langle a[t], G, C \rangle \in Plan$$

such that:

1. there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t = \tau \wedge TCS \wedge \Sigma(S)$,
2. there exists no action $\langle a'[t'], G^*, C' \rangle \in Plan$ and there exists no goal $\langle l[t'], G^* \rangle \in Goals$ such that
   - $G^* = G$ or $G^* \in ancestors(G, Goals)$, and
   - there exists no total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \geq \tau \wedge TCS \wedge \Sigma(S)$,
3. there exists no $\langle l[t'], G^* \rangle \in Goals$ such that
   - $G^* = G$ or $G^* \in ancestors(G, Goals)$, and
   - there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \leq \tau \wedge TCS \wedge \Sigma(S)$ and $KB \models_{TR} l[t']\sigma$,
4. let $C = l_1[t] \wedge \ldots \wedge l_n[t]$; if $n > 0$, then it is not the case that for some $i = 1, \ldots, n$ there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} TCS \wedge t = \tau \wedge \Sigma(S)$ and $KB \models_{TR} \overline{l_i[t]}\sigma$,
5. $executed(a[t], t') \notin KB_0$.

Then, $c_{AS}(S, \tau) \subseteq \mathcal{X}(S, \tau)$, $c_{AS}(S, \tau) = \{\langle a_1[t_1], \_, \_ \rangle, \ldots, \langle a_m[t_m], \_, \_ \rangle\}$ (where $m \geq 0$), such that there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} TCS \wedge t_1 = \tau \wedge t_m = \tau \wedge \Sigma(S)$.

c^c_{AS}: computational counterpart of c_{AS}

The computational counterpart for this corrected selection function can be obtained by having computational counterparts for the conditions 1-5 appropriately combined (either sequentially or concurrently checked, as discussed in D8), and then applying the final filter to get $c_{AS}(S, \tau)$ from $\mathcal{X}(S, \tau)$. The computational counterparts of 1–5 are as follows:[2]

1. $\vdash_{\Re} t = \tau \wedge TCS$
2. for each $\langle a'[t'], G^*, C' \rangle \in Plan$ and $\langle l[t'], G^* \rangle \in Goals$ such that $G^* = G$ or $G^* \in ancestors(G, Goals)$:

$$\vdash_{\Re} t' \geq \tau \wedge TCS,$$

3. for each $\langle l[t'], G^* \rangle \in Goals$ such that $G^* = G$ or $G^* \in ancestors(G, Goals)$:

$$\vdash_{TR} \text{ (finitely) fails to prove that } l[t'] \wedge t' \leq \tau \wedge TCS$$

4. let $C = l_1[t] \wedge \ldots \wedge l_n[t]$; if $n > 0$, then for every $i = 1, \ldots, n$

$$\vdash_{TR} \text{ (finitely) fails to prove that } \overline{l_i[t]} \wedge TCS \wedge t = \tau,$$

[2]Here $\vdash_{\Re}$ representes the computational counterpart of $\models_{\Re}$ ($\Re$-satisfiability) and $\vdash_{TR}$ represents the computational counterpart of $\models_{TR}$ (the Temporal reasoning capability).

5. $executed(a[t], t') \notin KB_0$.

Then, $c_{AS}(S,\tau) \subseteq \mathcal{X}(S,\tau)$, $c_{AS}(S,\tau) = \{\langle a_1[t_1], \_, \_\rangle, \ldots, \langle a_m[t_m], \_, \_\rangle\}$ (where $m \geq 0$), such that there exists a total valuation σ for the variables in TCS such that $\sigma \vdash_{\Re} TCS \wedge t_1 = \tau \wedge t_m = \tau \wedge \Sigma(S)$.

Trivially, if $\vdash_{TR}$ and $\vdash_{\Re}$ are correct computational counterparts of $\models_{TR}$ and $\models_{\Re}$, then the given computational counterparts of the checks 1–5 in the specification of c_{AS} are correct wrt the specification itself, and thus an overall correct computation counterpart of c_{AS} can be obtained.

3 Classification of Verifiable Properties

3.1 Guidelines for Choosing Properties

In this section we introduce and classify the properties that have been investigated within workpackage WP5. Recall from the Introduction that our main objectives in the choice of properties have been:

(O1) Effectiveness of the (computational logic-based) approach

(O2) Consequences of the design choices made

(O3) Versatility of the computee and society models

(O4) Guidance for application developers using PROSOCS

(O5) Exploring the chosen scenarios

All the properties considered go some way towards achieving the first objective, as they all demonstrate how the declarative and computational models of computees and societies lend themselves to formal specification and proof of properties. Below we give a general description of each class of properties considered and outline which of the other objectives they achieve.

3.2 Catalogue of Properties

Our classification of verifiable properties of societies of computees is based on the catalogue of properties put forward in deliverable D12 [3], which in turn extends the list of properties identified halfway through the second year of SOCS in [30]. In addition to the classes of properties given in the catalogue in D12, we now also include a class specifically devoted to properties of proof procedures.

Hence, we divide our catalogue of properties into four parts: (*i*) properties of proof procedures; (*ii*) properties of individual computees; (*iii*) properties of the social infrastructure; and (*iv*) properties specifically related to protocol conformance. As discussed already in [30], this classification is not strict and other classifications would have been possible as well. Our classification is summarised in Table 1.

Class		Instances
[PP-1]	Soundness properties	Section 4.1; Section 4.4
[PP-2]	Completeness properties	Section 4.2; Section 4.5
[PP-3]	Termination properties	Section 4.3
[IC-1]	Behaviour profiles	Section 5.1
[IC-2]	Coherence properties of the KGP model	Section 5.2
[IC-3]	Success criteria and preferences	Section 5.3
[IC-4]	Adopting social expectations	Section 5.4
[SI-1]	Well-definedness of societies	Section 6.1
[SI-2]	Automatically verifiable properties	Section 6.2
[SI-3]	Microeconomic properties	Section 6.3
[SI-4]	Stability properties	Section 6.4
[PC-1]	On-the-fly conformance checking	Section 7.2
[PC-2]	A-priori conformance checking	Section 7.3
[PC-3]	Protocol competence	Section 7.4

Table 1: Classification of properties

3.2.1 Properties of Proof Procedures

Properties of proof procedures developed within SOCS are relevant to both workpackages WP3 (computational models) and WP5 (verifiable properties). The investigation of properties of proof procedures began during the second year as part of WP3 and, in some cases, has carried on throughout the third year of SOCS as well. Although these properties are not themselves properties of societies of computees, they are immediately relevant to these properties, which is why we include their discussion in this deliverable. This group of properties includes three classes:

- [PP-1] Soundness properties of proof procedures.
- [PP-2] Completeness properties of proof procedures.
- [PP-3] Termination properties of proof procedures.

These properties achieve objective O2 (investigation of consequences of design choices). Termination results also give guidance about the design of the knowledge bases and thus help with objective O4.

3.2.2 Properties of Individual Computees

Under the general heading of *properties of individual computees*, we have identified the following main classes of properties:

- [IC-1] Behaviour profiles.
  We identify interesting profiles of behaviour and characterise them first in terms of properties of the states of computees and then design cycle theories that will induce traces whose states have these properties. We argue why such profiles may be advantageous given specific conditions, for example on the environment and timeliness requirements on the part of the computee.

- [IC-2] Coherence properties of the KGP model.
  This class of properties has been added since the submission of D12. It comprises properties of the computee model that demonstrate the appropriateness of some of the choices made in its design.

- [IC-3] Success criteria and preferences.
  Different states of a computee, as determined by its Knowledge Base and its current Goals and Plan, may be desirable for a computee in varying degrees. Particular states may reflect that the computee has been successful, for example in achieving its goals. More generally, it is possible to define preferences over alternative states, for instance, by introducing a notion of (individual) welfare enjoyed by a computee in a given state. These preferences may, for instance, depend on the number of achieved (or unachievable) goals, or at a greater level of detail, on whether or not the computee is making "progress" towards achieving its goals.

- [IC-4] Adopting social expectations.
  In the SOCS models of computees and societies, social expectations are specified in the social infrastructure, and private policies are specified in the computees' knowledge bases and cycle theories. We explore how these expectations and policies can be combined in the domain of resource allocation.

The properties under [IC-1], Behaviour Profiles, are good examples of our work towards achieving objective O3—investigating scope and versatility of the SOCS models, in particular in this case the KGP model. They show how to increase the level of heterogeneity in computees and the consequences of doing so. As such they also help towards objective O4—providing guidelines to developers. Another class of properties that works towards O3 is [IC-4], Adopting Social Expectations. These properties also demonstrate the scope of integrating the society and the individual computee models.

The properties under [IC-2], Coherence Properties, work towards objective O2—exploring consequences of design choices. The properties under [IC-3], Success Criteria and Preferences, have two purposes. One is that they work directly towards objective O2, showing that the KGP model guarantees a certain level of individual welfare. Another purpose is that they provide useful definitions and criteria to be used in other properties, for example in comparing different profiles of behaviour.

3.2.3 Properties of the Social Infrastructure

The next group of properties may be described as *properties of the social infrastructure:*

- [SI-1] Well-definedness of societies.
  We show under what circumstances a society is well-defined in the sense that its Social Knowledge Base, Social Integrity Constraints and Goal are such that amongst all its (closed) instances there exists at least one for which the Goal is achievable. This property provides guidance to the designers of societies.

- [SI-2] Automatically verifiable properties.
  We show how some formal properties, for example properties of protocols, can be verified automatically. For this purpose we extend the $\mathcal{S}$CIFF proof procedure and show how it can be used to verify whether a property, expressed as an existentially or universally

quantified formula, is a logical consequence of the Social Integrity Constraints of a given society.

- [SI-3] Microeconomic properties.
  Problems of distributed resource allocation are central to Global Computing in general and to SOCS in particular. Given, in addition to this, the importance of the notion of society within SOCS, it is natural to study microeconomic properties of societies of computees, for instance, by analysing the social welfare of a society in relation to the individual welfare enjoyed by its members. We show under what circumstances computees can reach optimal resource allocation, while varying the notion of optimality, class of deals and utility functions.

- [SI-4] Stability properties.
  Under this category we collect properties that abstract away from the internal details of agents. These properties concern collections of agents and result from the interaction amongst the agents and of individual agents with the environment in which they are situated. The main property is that of stability—all other properties in this class are defined in terms of stability.

The properties in [SI-1], Well-definedness of Societies, and [SI-4], Stability, address objective O4, and those in [SI-2], Automatically Verifiable Properties, address objectives O2 and O3. [SI-3], Microeconomic Properties, addresses objective O5.

3.2.4 Properties Related to Protocol Conformance

Interaction protocols are of central importance within SOCS, which is why we list properties related to *protocol conformance* separately:

- [PC-1] On-the-fly conformance checking.
  On-the-fly conformance is a property of an interaction (between two or more computees) with respect to a given society protocol. It assesses, at runtime, whether or not the interaction is legal according to the protocol. We show how this can be done automatically using the $\mathcal{S}$CIFF proof procedure.

- [PC-2] A-priori conformance checking.
  A-priori conformance is a property of a computee with respect to a protocol. It assesses, at design time, whether or not a computee can be guaranteed to always respect the protocol. We show how a-priori conformance can be checked by analysing the relationship between the computees' private policies and the public protocols. The results give guidelines for the design of computees.

- [PC-3] Protocol competence.
  By protocol competence we understand any property assessing how well a computee is able to "use" a protocol beyond the basic requirement of being able to conform. An example would be the ability to react to an incoming message in all of the ways foreseen by the protocol under appropriate circumstances (as opposed to an incompetent computee that may, for instance, have to reject any incoming proposal, however advantageous, due to a limited knowledge base). We show how a particular aspect of protocol competence, namely the ability to reach any given "state" in a given protocol, can be checked by analysing the relationship between the specifications of a group of computees and a given

protocol. The results can provide guidelines on how to "customise" protocols to adapt them to the needs and abilities of (possibly not fully competent) computees.

A further differentiation of the concept of protocol conformance is discussed in Section 7.1. The three classes of properties given in this section all work towards objective O3, and the last two also work towards objective O4.

3.3 Other Properties and Related Work

As was mentioned in the Introduction the list of properties that we have addressed in WP5 has been compiled through a process of deliberation and refinement of earlier ideas reported in D3 [50], the Provisional List of Properties [30], and D12 [3]. We have added a number of classes of properties to these earlier lists. These include `[IC-2]` Coherence Properties and `[SI-4]` Stability Properties. We have also removed a few of the earlier properties. These include *termination of social interaction*, *fairness* and *termination, determinism and exhaustiveness*.

Properties related to *termination of social interaction* include, for instance, the termination of negotiation dialogues in the context of a resource allocation scenario (and in this specific context we have addressed it in our study of microeconomic properties). Most importantly for SOCS, however, termination issues at the social level are closely linked to termination properties of the society proof procedure. This is why we have removed this class of properties in favour of the newly introduced termination properties of proof procedures.

The property of *fairness* has been dropped from our list, because it appears not to be of specific relevance to societies of computees (beyond its general relevance to multiagent systems). Nevertheless, fairness is at least *related* to some of the work carried out in SOCS. For instance, our work on egalitarian agent societies (see Section 6.3) is concerned with the development and characterisation of negotiation mechanisms for the fair allocation of resources in societies of autonomous agents [28]. Furthermore, our work on the customisation of protocols (reported in Section 7.4 under protocol competence) may prove useful for developing a methodology for adapting existing interaction protocols to the characteristics of specific computees (who may not be competent users of the full protocols) in a fair manner [22].

Finally, the entry on *termination, determinism and exhaustiveness* listed in earlier draft classifications of properties we were interested in [3, 30] includes properties of a computee's reaction to observations caused by actions of other computees in a society. But given that these issues have already been addressed by members of the SOCS Consortium [68, 74] prior to the commencement of the project (for an agent model that may be considered a simplified variant of the KGP model), we decided to prioritise our work on WP5 by including other new properties instead.

We should stress that the type of property typically investigated in the context of modal logic-based approaches to modelling intelligent agents within a BDI architecture do not address what we have aimed for in SOCS. Rao and Georgeff's seminal paper in this area [63], in particular, is concerned with the *axiomatisation* of simple properties such as "if an agent intends α then he will also believe α" and the identification of the corresponding model-theoretic properties (relationships between the different accessibility relations in the Kripke frame). This is a worthwhile enterprise in multi-dimensional modal logics (such as BDI logics), where the basic logic does not sufficiently determine the relationship between the different modalities (belief, desire, intention). In the KGP model, on the other hand, this issue simply does not arise. The connections between the different "dimensions" have been fixed as part of our definition of the KGP transitions. Nevertheless, our *coherence properties* (Section 5.2) are broadly related to

this, as they also concern properties inherent in the model. The difference is that we show how these coherence properties *follow* as a consequence of design choices made in the specification of the KGP model, while the type of BDI properties addressed by Rao and Georgeff are in fact themselves part of the design process.

3.4 Presentation of Results

Our concrete results regarding properties of societies of computees are presented in Sections 4–7. We have used a *common template* throughout to report these properties. Here, we explain the template.

Property template header. The header of each entry includes information on the *name* of the property, the *classification* of the property with respect to our catalogue of properties presented earlier, and the names of the SOCS *partners* involved in working on this particular property. Naturally, some properties will be relevant to more than one of the classes of properties we have identified. In such cases, the most important class is given in the header and other relevant classes are referred to in the body of the text.

The header also includes references to the most *relevant papers*. These are either published papers, previous SOCS deliverables, or technical reports included in the annex of this deliverable. We have aimed at keeping the presentation of these results short and succinct in the templates. Therefore, any proofs as well as formal definitions requiring an extensive technical apparatus have been omitted. These may be found in the referenced papers.

Finally, the header also includes information on the *status* of the property, i.e. whether the property has been fully *proved* or fully *formalised*, or whether the reported results are of a *preliminary* nature.

Property template body. The body of each entry includes the following information:

- *Informal statement of the property.* We start by giving a short self-contained description of the property that should be comprehensible without having an intimate knowledge of the formal tools used in SOCS. Where appropriate, we have also included a short introductory paragraph to introduce the wider context of the property before giving the informal statement.

- *Significance.* Here we comment on the significance of the property in the context of SOCS. Important arguments for the significance of a property include, for instance, (*i*) that it demonstrates the effectiveness of the SOCS models, (*ii*) that it provides guidelines for developers using our platform, (*iii*) that it is relevant to the evaluation criteria given in deliverable D3 [50], or (*iv*) that it contributes to one of the research areas addressed by SOCS, including Global Computing, Computational Logic and Multiagent Systems, in a significant way.

- *Formal statement of the property.* If possible, we provide a short self-contained statement. Otherwise we refer to the relevant papers that provide a full formalisation and proofs (where appropriate).

- *Approach.* In this paragraph we describe the general approach followed in obtaining the result. In particular, again, we do not give formal proofs here but rather refer to the relevant papers listed in the header of the entry.

- *Related work.* Here we briefly comment on related work to the specific property in question. In some cases, where we have considered several closely related properties, related work is discussed in a central place before the templates for that group of properties.

In Sections 5.1.3–5.1.8 on behaviour profiles a slightly different template is used, as this is more appropriate. The template there is introduced in Section 5.1.1.

Overview of results. Table 1 on page 19 provides an overview of the results achieved. For each of the classes of properties identified earlier, it provides a reference to the concrete properties studied by giving the appropriate section numbers.

4 Proof Procedures

In this section we present properties of the proof procedures developed within SOCS. These procedures have been used to implement the capabilities of individual computees and the reasoning mechanisms available in the society model and thereby directly influence higher-level properties of both computees and societies of computees.

4.1 $\mathcal{S}$CIFF Soundness

Property classification:	[PP-1] Properties of proof procedures
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Deliverable D8 [36], Gavanelli et al. [34]
Status:	Proved

Informal statement of the property. Soundness of a proof procedure refers to the fact that if an answer is obtained by the proof procedure, then it is also logically entailed by the declarative semantics. In the context of computees this means that if the $\mathcal{S}$CIFF proof procedure has a successful derivation, then the computees have behaved according to the protocols defined in the Society Knowledge Bases and the expectations have not been violated.

Significance. Soundness is one of the most important properties of a proof procedure, and is the main link between declarative and operational semantics. This result shows that the $\mathcal{S}$CIFF proof procedure can be used for checking (both on-the-fly and a-posteriori) that computees behave in accordance with protocols defined by the Social Integrity Constraints. Importantly, this checking is done without necessarily knowing anything about the computees' internals (in fact the society could be composed of a mixture of computees and agents of other kind, or it could even include no computees at all). This result helps in several respects, including checking the correct behaviour of a society of computees in a highly heterogeneous and dynamic setting. This is an original and significant result of the project.

Formal statement of the property. Soundness has been stated both for the open and for the closed instances of a society in D8 [36]. The following theorem relates the operational notion of open successful derivation with the corresponding declarative notion of goal achievability.

Theorem 4.1 (Open soundness) *If* $\mathcal{S}_{\mathbf{HAP}^i}$ *is an open society instance, and*

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

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

$$\mathcal{S}_{\mathbf{HAP}^f} \mathrel{|\!\approx}_{(\mathbf{EXP} \cup \mathbf{FULF})\sigma} G\sigma$$

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 4.2 (Closed soundness) *If* $\mathcal{S}_{\overline{\mathbf{HAP}^f}}$ *is a closed society instance, and*

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

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 $(\mathbf{EXP} \cup \mathbf{FULF})\sigma$).

The property has been proved for some restricted cases in deliverable D8 [36]. The proof has been extended to the general case in a report annexed to deliverable D12 [35].

Approach. The proof of soundness of the $\mathcal{S}$CIFF proof procedure reuses the proof of soundness of the IFF proof procedure [32]. Concerning dynamically incoming events, a lemma [36] links the on-the-fly conformance checking with the a-posteriori one, showing that the success nodes for on-the-fly verification with a dynamically growing history, are the same obtained by the $\mathcal{S}$CIFF proof procedure with the whole history given a priori. A further useful lemma proved in D8 states 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 these lemmas, in [36], we were able to use soundness 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 *terminate* in a node without any universally quantified abducible. Finally, we extended the proof of soundness to derivations in which universally quantified abducibles can occur [34]. In such cases, we showed that the derivation is sound relying on the soundness of a second $\mathcal{S}$CIFF derivation in which universally quantified abducibles are moved to the static part of the Society Knowledge Base.

Related work. The proof of soundness of the $\mathcal{S}$CIFF proof procedure is based on the corresponding proof of soundness of the IFF proof procedure [32].

4.2 $\mathcal{S}$CIFF Completeness

Property classification:	[PP-1] Properties of proof procedures
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Gavanelli et al. [34]
Status:	Formalised

Introduction. The completeness property of the $\mathcal{S}$CIFF proof procedure is directly relevant to our approach of verifying protocol properties automatically (see Section 6.2). Work on a completeness proof is currently under way; we outline the property and its significance here.

Informal statement of the property. Completeness of the proof procedure refers to the fact that if a result is logically entailed by the declarative semantics, then it is also an answer obtainable by the proof procedure.

Significance. Completeness can be used to affirm that if the $\mathcal{S}$CIFF proof procedure has a successful derivation, then the computees are behaving according to the protocol. It can be considered symmetrical to soundness. In the closed case, it is important also for automatically proving other properties, since, as in model checking, the automatic verification of some properties is done by adopting the negation of properties as goals (see Section 6.2).

Formal statement of the property. Completeness was already stated in deliverable D8 [36], both for open and closed instances of the society. Completeness in the open case states that if goal G is achievable in an open society instance under the expectation set $\mathbf{EXP}^\star$, then an open successful derivation can be obtained for G, possibly computing a set $\mathbf{EXP}^{\star\prime}$ of the expectations whose grounding (according to the expectation answer) is a subset of $\mathbf{EXP}^\star$. We hope to be able to prove the following result:

Theorem 4.3 (Open completeness) *Given an open society instance $\mathcal{S}_{\mathbf{HAP}}$, and a (ground) goal G, for any set of ground expectations, $\mathbf{EXP}^\star = \mathbf{EXP} \cup \mathbf{FULF}$, such that $\mathcal{S}_{\mathbf{HAP}} \not\approx_{\mathbf{EXP}^\star} G$ then $\exists \mathbf{EXP}^{\star\prime}$ such that $\mathcal{S}_\emptyset \not\sim^{\mathbf{HAP}}_{\mathbf{EXP}^{\star\prime}} G$ with an expectation answer $(\mathbf{EXP}^{\star\prime}, \sigma)$ such that $\mathbf{EXP}^{\star\prime}\sigma \subseteq \mathbf{EXP}^\star$.*

More important is completeness in the closed case, in particular for the automatic verification of properties (Section 6.2). Completeness in the closed case states that if goal G is achieved in a closed society instance under the expectation set $\mathbf{EXP}^\star$, then a closed successful derivation can be obtained for G, possibly computing a set $\mathbf{EXP}^{\star\prime}$ of the expectations whose grounding (according to the expectation answer) is a subset of $\mathbf{EXP}^\star$. We hope to be able to prove the following result:

Theorem 4.4 (Closed completeness) *Given a closed society instance $\mathcal{S}_{\overline{\mathbf{HAP}}}$, a (ground) goal G, for any set of ground expectations, $\mathbf{EXP}^\star = \mathbf{EXP} \cup \mathbf{FULF}$ such that $\mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^\star} G$ then $\exists \mathbf{EXP}^{\star\prime}$ such that $\mathcal{S}_\emptyset \vdash^{\mathbf{HAP}}_{\mathbf{EXP}^{\star\prime}} G$ with an expectation answer $(\mathbf{EXP}^{\star\prime}, \sigma)$ such that $\mathbf{EXP}^{\star\prime}\sigma \subseteq \mathbf{EXP}^\star$.*

Approach. We plan to exploit the completeness proofs for the IFF proof procedure [32, 79]. By exploiting the termination result (see Section 4.3), we plan to prove strong completeness, i.e. completeness independent of the order of applications of the rewriting steps.

Related work. The IFF proof procedure was proven sound and complete by Fung [32]. However, the statement of completeness was of a weak form showing only that some order of application of the rewriting steps obtain the required answer. Xanthakos [79], after proving termination, was able to strengthen this statement to *strong completeness* where any order of application of the rewriting steps obtains the required answer.

4.3 $\mathcal{S}$CIFF Termination

Property classification:	`[PP-1]` Properties of proof procedures
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Gavanelli et al. [34]
Status:	Proved

Informal statement of the property. Under some syntactic restrictions, namely, acyclicity of the society knowledge bases, the $\mathcal{S}$CIFF proof procedure terminates.

Significance. Termination is one of the key steps for the practical usability of an algorithm. Of course, one would like a proof procedure to terminate for every possible input; however, as it is obvious from the $\mathcal{S}$CIFF syntax, the user can write infinite loops. Termination is therefore important not only for the purpose of proving properties of the proof procedure, but also to guide the design of programs and knowledge bases. Through termination, one can prove other formal properties of the proof procedure, for example using termination and completeness a strong completeness property [79] can be proved.

A termination result and the syntactic restrictions which it relies upon can also guide the user in writing the Society Knowledge Bases, being able to distinguish those that allow termination and those that may not. Finally, the proof of termination provided useful guidelines for the implementation of the proof procedure, such as heuristics not specified by the operational semantics. In particular, it suggested a preferred order of application of the rewriting steps given in the operational semantics.

Formal statement of the property. Termination is proven, as for SLD resolution, for *acyclic* knowledge bases and *bounded* goals and implications. For definitions of boundedness and acyclicity for the Society Knowledge Bases, please refer to annexed document [34].

Theorem 4.5 (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 $\mathcal{IC}_S$ are bounded wrt. the level-mapping. Then, every $\mathcal{S}$CIFF derivation for G for each instance of $\mathcal{S}$ is finite.*

Approach. Xanthakos [79] proved the termination of the IFF proof procedure. Since $\mathcal{S}$CIFF extends IFF, our proof of termination roughly follows the same overall structure as the proof by Xanthakos. First, we prove that the presence of happened events inside social integrity constraints does not change the fact that a program is acyclic. Since the $\mathcal{S}$CIFF proof procedure, as Constraint Logic Programming [41], considers constraint solving as an extension of unification (or equality rewriting, in IFF), we extend the proofs regarding equality rewriting to constraint solving. More precisely, we show that the properties of equality rewriting rewriting steps that are exploited in the proof of termination for IFF also either hold for constraint solving rewriting steps, or are reasonable assumptions for a constraint solver. Finally, we give a proof along the

lines of that given by Xanthakos, extending it for the $\mathcal{S}$CIFF proof procedure. The complete proof may be found in the annexed document [34].

Related work. Naturally, the closest work is the proof of termination of the IFF proof procedure [79]. The $\mathcal{S}$CIFF proof of termination extends the proof by Xanthakos to deal with the new features of $\mathcal{S}$CIFF.

4.4 CIFF Soundness

Property classification:	`[PP-1]` Properties of proof procedures
Partners involved:	ICSTM, DIPISA
Relevant papers:	Endriss et al. [23, 24]
Status:	Proved

Informal statement of the property. The CIFF proof procedure for abductive logic programming is sound in the sense that any derived answer gives rise to a correct answer according to the completion semantics (soundness of success). Furthermore, whenever the procedure fails, there exists no correct answer according to the semantics (soundness of failure).

Significance. The soundness property is central to any proof procedure. In the context of SOCS, our soundness result ensures the correctness of the computational models of the planning, reactivity and temporal reasoning capabilities, all of which are based on the CIFF procedure.

Formal statement of the property. The input to CIFF consists of a theory Th (a selectively completed logic program), a set of integrity constraints IC, and a query Q. A correct answer to a query Q with respect to an abductive logic program $\langle Th, IC\rangle$ is a pair $\langle \Delta, \sigma\rangle$, where Δ is a finite set of ground abducible atoms and σ is a substitution for the free variables occurring in Q, such that

$$Th \cup Comp(\Delta) \models IC \wedge Q\sigma$$

Our soundness results for the CIFF proof procedure are as follows:

Theorem 4.6 (CIFF soundness of success) *If there exists a successful derivation for the input $\langle Th, IC, Q\rangle$, then the extracted answer gives rise to a correct answer for that input.*

Theorem 4.7 (CIFF soundness of failure) *If there exists a derivation for the input $\langle Th, IC, Q\rangle$ that terminates and where all final nodes are failure nodes, then there exists no correct answer for that input.*

These results have been published in [24]; full proofs may be found in [23].

Approach. The proof of both of the above theorems relies on the fact that all CIFF proof rules are equivalence-preserving in the sense that a node in a derivation is always logically equivalent to the disjunction of its immediate successor nodes. It follows that any final node in any branch of the proof tree logically entails the initial node, which is the conjunction of the query and the integrity constraints. Another important lemma establishes the correctness of the CIFF answer extraction procedure. It shows that the formulas in a final success node that are not directly represented by the extracted answer (i.e. any formulas that are not abducible

atoms) are in fact logical consequences of the extracted answer. Our two soundness theorems then follow from these intermediate results.

In [24], we also note that the CIFF dynamic allowedness rule, which labels nodes containing formulas with non-allowed quantification patterns as undefined on-the-fly (to avoid floundering), is appropriate in the sense that such nodes would either lead to infinite abductive answers (which would not conform to the chosen semantics) or at least require the enumeration of *all* the solutions to a set of CLP constraints. Also, while the original IFF procedure would generate a wrong answer for a non-allowed input (with respect to the definition of allowedness used in [33]), CIFF will either state explicitly that the output is undefined or the answer will be both well-defined and correct.

Related work. Our soundness proof is based on and extends the original proof for the soundness of the IFF procedure [32].

4.5 CIFF Completeness

Property classification:	`[PP-1]` Properties of proof procedures
Partners involved:	ICSTM, DIPISA
Relevant papers:	Endriss et al. [24]
Status:	Preliminary results

Introduction. Work on possible completeness results for the CIFF procedure is currently under way. Here we present a simple result that applies to inputs for which the procedure can be guaranteed to terminate.

Informal statement of the property. CIFF is complete for a limited class of inputs for which termination can be guaranteed and for which there are no problems with quantification patterns (allowedness).

Significance. Completeness is a highly significant property for any proof procedure, albeit proving completeness can often be difficult in practice. In the context of SOCS and with regard to the applications of CIFF to implement KGP capabilities, soundness is the central property of interest, as it ensures, for instance, that any plan that does get computed will be a correct plan. Completeness would also allow us to draw conclusions about a computee's ability to compute a plan whenever that is possible at all, but this turns out to be less significant in practice. Furthermore, for both planning and reactivity, in the implementation of the PROSOCS system, only the first answer computed by CIFF will be fed into the respective capability. Finally, for temporal reasoning, only the question whether or not a query will succeed matters. This is covered by the simple completeness result presented here (i.e. the question whether or not *all* abductive answers will be found is irrelevant).

Formal statement of the property. For any class of inputs (consisting of a theory, a set of integrity constraints, and a query) that are known to be *allowed* (in the sense of never triggering the *dynamic allowedness rule* [24]) and for which termination can be guaranteed, the CIFF procedure will terminate successfully whenever there exists a correct answer according to the completion semantics for abductive logic programming with constraints.

Approach. This property is an immediate consequence of the two soundness theorems given in the previous section. In fact, the *soundness of failure* result presented earlier may itself be regarded as a kind of completeness result.

Note that both the requirement that the input is known to be allowed and that it is known to lead to a terminating CIFF derivation are very strong assumptions. We hope to address the question of identifying interesting classes of input programs for which termination can be guaranteed (for instance, by imposing suitable acyclicity conditions [79]) in our future work. This would make the above result more applicable.

Related work. In the context of the IFF proof procedure, Fung [32] proves a similar completeness result as the one presented here as well as a more general result that relies on the three-valued completion semantics. The work of Xanthakos on the termination of IFF for acyclic logic programs is also relevant [79].

5 Individual Computees

In this section we present our results on properties of individual computees.

5.1 Behaviour Profiles

At the level of individual computees, we have identified and analysed interesting and useful profiles of behaviour and their corresponding cycle theories. Section 5.1.1 gives a brief introduction to behaviour profiles and discusses related work. Sections 5.1.2–5.1.8 then present specific profiles of behaviour that we have identified and that we have formalised both in terms of describing properties of states of computees following such profiles and by giving cycle theories that will induce such behaviour. The study of behaviour profiles is intended to explore the versatility of the computee model and to provide guidelines for developments within the PROSOCS architecture.

5.1.1 Introduction to Profiles

The KGP model of individual computees allows for a variety of different *profiles of behaviour*, depending on the choices made for the rules of a computee's cycle theory. After a joint effort in identifying interesting profiles of behaviour, we undertook the task of formalising these profiles and of defining suitable cycle theories that would induce such behaviours. Our results are reported in Sections 5.1.2–5.1.8. In these sections we present the specific profiles, and for each we give (1) a formalisation of the properties of states of computees equipped with the profile, (2) the cycle theories that will induce such states, (3) summary characterisations of environments and situations where computees equipped with the profile will perform either better or at least as well as other profiles such as what we have defined as the "normal profile", and (4) some concrete examples of the possible consequences of the behaviours induced by the profile.

For both (3) and (4) we often concentrate on comparing how well computees do in terms of achieving their goals, whether or not they manage to achieve any goals, and if they do, then how many. This links our work on profiles with the work on *individual welfare*. As is described in Section 5.3 one of the notions of welfare we have defined is in terms of the number of goals

achieved. So in the work on profiles we have attempted to study the circumstances under which one profile may lead to increased *individual welfare* compared to another.

The objectives of the study of profiles are twofold:

- To explore the scope and versatility of the computee model: we explore to what extent we can provide heterogeneity in the behaviour of computees by changing their control (cycle) theories in a modular way, and we explore the consequences of this heterogeneity.

- To give guidelines to developers who may use the PROSOCS platform: our identification of the circumstances, for example environmental factors, such as time criticality and dynamism of the environment, under which one profile has advantages over another can guide application developers in their choice of profile.

The choice of profiles we have studied was guided primarily by the nature of Global Computing environments and by our interest in exploring how the behaviour of computees changes as a consequence of changing various parameters. The Cautious, Actively Cautious and Objective profiles were motivated by the highly dynamic nature of Global Computing environments, where the environment external to the computee changes unexpectedly and frequently. So these profiles explore ways in which the computee attempts to keep itself better informed about the changes in the environment that affect the successful execution of the computees' actions. The Focussed, Punctual and Impatient profiles are motivated by the time-critical nature of Global Computing environments. These profiles explore ways in which a computee can improve its chances of achieving at least some of its goals, even when achieving all its goals is not feasible. Finally, the Careful profile studies the advantages of a computee which frequently examines and updates its commitments, in terms of its goals and plans, in its internal state.

The theoretical work reported here on profiles has motivated and is complemented by the experimental work on profiles reported in deliverable D14 [2].

Related work. Here we briefly summarise related work that is relevant to all the different profiles presented in subsequent sections. There are two main aspects to consider, namely (*i*) programming agent control as in 3APL and (*ii*) attitudes of agents.

Profiles of computees are an attempt to address the need that different applications require different deliberation processes which therefore should be controlled by the designer/programmer. In 3APL [40] a meta-language that refers to goals, actions, plans and rules together with constructs from an imperative programming is used in order to implement how to process the object-level entities in the application. Our approach shares the aim of 3APL to make the agent cycle programmable and the selection mechanisms explicit, but it goes beyond it. Indeed, the approach of [40] can be seen as relying upon a catalogue of fixed cycles according to some criteria, whereas we drop the concept of fixed cycle completely and replace it with fully programmable cycle theories. Our approach allows us to achieve enhanced flexibility and adaptability in the operation of a computee.

Most of the existing literature refers to behaviour at the level of social attitudes and personalities whereas our cycle profiles refer more to the level of problem solving strategies (given an application environment). For example, in [9] five profiles (agreeable, disagreeable, argumentative, open-minded, elephant child) are proposed to discriminate between different attitudes of agents with varying degree of willingness to cooperate. In a computee this aspect of its personal attitude can be captured as a part of its Goal Decision knowledge base, KB_{GD}, where it can influence its decisions (see e.g. [46, 44]). Nevertheless the two levels are linked as a

specific personal attitude can lead or induce a problem solving strategy. In [72] different agent architectures are studied where each one of these is based on a different combination of reactive/deliberative capabilities depending on the personal attitude to its current needs. Different survival behaviours emerge from the different architectures. In our computees such variety of behaviour can be achieved via a combination of suitable Goal Decision capabilities and cycle theories.

In the BDI framework three different *commitment strategies* have been defined to express relationships between current and future intentions [63]. These are *blind*, *single minded* and *open minded*. Informally, a blindly committed agent is one which maintains its intentions until it believes that it has achieved them. A single minded agent, on the other hand, maintains its intentions until it believes they are achievable, and an open minded agent maintains its intentions while they are still its goals. Rao and Georgeff then analyse the consequences of these strategies in terms of the agents' beliefs and intentions. Our work on profiles shares the objectives behind the study of commitment strategies but goes beyond that study in the range of profiles explored and their consequences.

Property template for behaviour profiles. In this section, we use a slightly different template to report results on properties than in the rest of the deliverable. The reason for this is that we typically report two different types of properties in each entry. The first are *correspondence properties*. They establish that a given cycle theory does indeed induce an operational trace that has the properties characterising the behaviour profile in question. The second type are formal properties that demonstrate *advantages* of the profile in question (in some cases such advantages are only demonstrated by means of suitable examples). For each profile identified we report the following:

- Informal description of the profile
- Significance
- Formal description of the profile
- Cycle theory
- Advantages of the profile
- Approach

Related work on profiles has been discussed above as well as, in more detail, in the annex [13].

5.1.2 The Cautious Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	DIPISA
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. A *cautious* profile of behaviour requires a computee to execute actions in its plan only if it believes that the preconditions of these actions are currently true.

Significance. The main advantage of the cautious profile is to minimise the risk of action execution failure, in the sense of actions not causing the expected effects, by checking their preconditions in advance.

Formal description of the profile. The cautious profile has been characterised in terms of properties of the states that may occur in the operational trace, as well as in terms of a cycle theory [13]. The states of the operational traces a cautious computee can exhibit can be characterised as follows: given an operational trace

$$T_1(S_0, X_1, S_1, \tau_1), \ldots T_i(S_{i-1}, X_i, S_i, \tau_i), \ldots$$

for all the i such that $T_i = AE$ (Action Execution), and for all the actions $\langle a[t], \_, C\rangle$ in X_i, the following holds:

$$KB_{TR}^{i-1} \models_{TR} C \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$$

That is, whenever the computee executes an action, then all the preconditions of that action are believed to hold.

Cycle theory. The cycle theory for a cautious profile simply contains the following basic rule:

$$\begin{aligned} Cautious_{T|AE}(S', Y) : *AE(S', Y) \leftarrow \quad & T(S, X, S', \tau), selected\_actions(S', X', \tau'), \\ & prec\_sat(X', Y, \tau'), now(\tau'), Y \neq \emptyset. \end{aligned}$$

where $selected\_actions/3$ implements the core action selection function and $prec\_sat/3$ selects in Y, amongst the actions in X', those whose preconditions can be proved to hold at the current time τ' (by calling the TR capability). In order for the transition to occur Y must not be empty. The above rule is required to be the only one in the cycle theory defining AE transitions. It has been proved that the traces induced by this cycle theory fulfil the above mentioned state properties.

Moreover, it has been shown how a cautious profile can also be realised by means of behavioural rules and proved that this implementation satisfies the state characteristics of the profile. These rules can be "injected" in any cycle theory, inducing in it a cautious behaviour as far as AE transitions are concerned.

Advantages of the profile. The features of the environment which could be relevant for the behaviour of the profile and its capability to improve individual welfare have been studied. It has been shown that under certain conditions a cautious computee is more likely to achieve the expected effects from the executed actions than a normal computee. Informally, this in particular happens when precondition truth values may change in the environment, before actions become timed-out. In this case, the cautious computee has the chance to wait for precondition to hold before executing the actions which depend on them.

Approach. This profile has been defined in order to improve the effectiveness of the actions performed by a computee, whose effects may be jeopardised by the fact that the preconditions of the executed actions fail to hold. In its basic formulation, a computee is required not to execute actions whose preconditions are not believed to be true. This profile is an extension of the normal one, obtained by restricting the set of executable actions to those whose preconditions

are known to be true. The actively cautious profile, see Section 5.1.3, extends the approach by requiring that the computee actively checks in the environment preconditions of actions before they are executed, by introducing suitable sensing actions.

An experiment with cautious computees is presented in deliverable D14 [2].

5.1.3 The Actively Cautious Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	UCY
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. An *actively cautious* profile of behaviour requires a computee to execute actions in its plan only if it believes that the preconditions of these actions are currently true. In addition, the actively cautious computee executes sensing actions in order to actively obtain the value of unknown preconditions.

Significance. The study of profiles of behaviour intends to provide guidelines for developers using PROSOCS. The actively cautious profile will be particularly useful in environments where it is important that the computee does not execute actions whose preconditions do not hold.

Formal description of the profile. The actively cautious profile is characterised by the following property of the operational trace: Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of the computee where $S_i = \langle KB^i, Goals^i, Plan^i, TCS^i \rangle$ and T_i is an action execution transition for a set of actions As, then

- For every action $A_j \in As$ where $A_j = \langle a_j[t], \_, C_j \rangle$ the following condition holds:

$$KB^{i-1}_{TR} \models_{TR} C_j \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$$

- and, if there exists an action $A' \in c_{AS}(S_i, \tau_i)$ where $A' = \langle a'[t], \_, C' \rangle$ and there exists a maximal non-empty set of preconditions $C'' \subseteq C'$ such that for each $c \in C''$:
  - $KB^{i-1}_{TR} \not\models_{TR} c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$
  - $KB^{i-1}_{TR} \not\models_{TR} \neg c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$

  then all actions in the set As are necessarily sensing actions.

In other words, whenever a computee executes a set of actions, two conditions hold. Firstly, all the preconditions of those actions are known to hold at the time of the execution. Secondly, the actions executed are all sensing actions, if there exist actions in the set of selected actions for the current state whose preconditions are unknown. Notice that this condition prevents the computee from executing non-sensing actions whose preconditions are known to hold, when there exist other actions in the set of selected actions with unknown preconditions. This does not necessarily mean that the computee will execute sensing actions for all the unknown preconditions of all the actions in its plan before it starts executing the actions. Its actual behaviour depends on the precise definition of the selection function and the extent to which this selects a maximal set of the actions in the current *Plan*. For example, if the selection function takes

into account the time ordering of actions and does not select together actions which are totally ordered at different times then if there exist two actions, A_1 and A_2, whose preconditions are not known to hold, and action A_1 must be executed before action A_2, then the computee will not sense for the preconditions of the action A_2 until A_1 is executed.

We study a special case where we assume that only one action is executed at a time. Note that this sets a constraint on the input parameters of the AE transition and not on the selection function. The results and proofs that follow are based on this assumption.

Cycle theory. The actively cautious profile can be captured through a suitable cycle theory. A short description of it is given below. The full cycle theory is given in [13].

An *actively cautious cycle theory* is *any* cycle theory where

- The following 3 rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:
  - $Actively\_Cautious\_\mathcal{P}^{PI}_{SI \succ T} : \mathcal{R}_{PI|SI}(S, Ps) \succ \mathcal{R}_{PI|T}(S, X)$
    for all transitions $T \neq SI$
    This rule gives higher priority to a SI transition over any other transition, when the last transition was PI.
  - $Actively\_Cautious\_\mathcal{P}^{T}_{AE \succ AE}$:
    $\mathcal{R}_{T|AE}(S, A) \succ \mathcal{R}_{T|AE}(S, X) \leftarrow A = \langle sense\_precondition(c[t]), \_, \_\rangle, \exists A' \in c_{AS}(S, \tau), A' = \langle a[t], \_, C\rangle, c \in C, unknown(c), now(\tau)$.
    for every T, X such that X is not a sensing action, where the predicate $unknown(c)$ is defined appropriately using the Temporal Reasoning capability of the computee.
    This rule gives higher priority to an AE transition where the action is a sensing action, over any other AE transition, when there exist actions in the set of selected actions with unknown preconditions.
  - $Cautious\_\mathcal{P}^{T}_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$,
    for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$, where the predicate $unknown\_pred(S, As)$ is defined appropriately using the Temporal Reasoning capability of the computee.
    This rule gives lower priority, over any other transition, to AE transitions with input parameter a set of actions which contains an action whose preconditions are not known to hold at the current state and time.
- The above three rules have higher priority than any other priority rule.
- The following rule is part of the $\mathcal{T}_{basic}$ part of the theory

  $$\mathcal{R}_{PI|SI}(S', Ps) : *SI(S', Ps) \leftarrow PI(S, Gs, S', \tau), Ps = c_{PS}(S', \tau'), Ps \neq \{\}, now(\tau')$$

  This rule enables a SI transition to follow a PI transition. We assume that the set of preconditions of each of the sensing actions added to the plan by this rule, is empty.

We have proved the following correpondence result:

- Proposition 1: The proposed actively cautious cycle theory induces operational traces that satisfy the characteristic feature of the actively cautious profile.

Advantages of the profile. We call an *underlying (non-actively cautious) cycle theory* any cycle theory obtained from an *actively cautious cycle theory* by removing the extra rules which were added to the theory in order to achieve the actively cautious behaviour.

We examine the following propositions:

- Proposition 2: A computee with an *actively cautious cycle theory* is as good as a computee with an *underlying cycle theory* with the same knowledge base KB, in the sense that whenever the latter succeeds in achieving a goal, so does the former, in environments where:
  - sensing actions always succeed in finding out the values of all the preconditions that were sensed for;
  - during the delay in executing an action, due to sensing for its preconditions, there is no change of the truth value of the preconditions via some unknown factor in the environment;
  - time is not very critical. Each action A_j in a plan $Plan$, where $A_j = \langle a_j[t], \_, C_j \rangle$, has at least time T_{A_j} before it times out, where $T_{A_j} = 1 + \Sigma_{j=0}^{|Plan|} |C_j|$, where $|Plan|$ denotes the number of actions in $Plan$ and $|C_j|$ denotes the number of precondition fluents in the set C_j.
- Proposition 3: In certain cases a computee with an *actively cautious cycle theory* is better than a computee with an *underlying cycle theory* in that the first succeeds in achieving a goal while the second fails to achieve the same goal.

Approach. All propositions are proved in the annex document [13]:

- Proposition 1: We prove that the proposed cycle theory induces operational traces that satisfy the required feature by proving that:
  - there does not exist an admissible set of rules which concludes an action execution transition for some action whose preconditions are not known, and
  - there does not exist an admissible set of rules which concludes an action execution which is not a sensing action, when there exist actions with unknown preconditions in the set of selected actions.
- Proposition 2: We prove that a computee with an *actively cautious cycle theory* achieves any goal that a computee with an *underlying cycle theory* achieves in the given environment by showing that in the given environment a computee with the actively cautious cycle theory is able to reproduce the successful operational trace of a computee with the underlying cycle theory.
- Proposition 3: We provide the following example to demonstrate that in certain cases an actively cautious computee succeeds but a computee with an underlying cycle theory fails. Suppose that the two computees have exactly the same knowledge base and that there exists a set of actions $A_1, \ldots, A_n$ in the set of selected actions whose preconditions are unknown. Also assume that actions $A_1, \ldots, A_{n-1}$ belong to plans for a set of goals Gs whereas action A_n belongs to a plan for a goal $G' \notin Gs$.

According to the definition of the actively cautious computee, it will execute sensing actions since there exist actions in the set of selected actions with unknown preconditions. After executing sensing the actively cautious computee finds out that the preconditions of the actions $A_1, \ldots, A_{n-1}$ do not hold whereas the preconditions of action A_n hold. Therefore the actively cautious computee does not execute any of the actions whose preconditions do not hold and executes next action A_n. The execution succeeds and goal G' is achieved.

A possible operational trace for the computee with the underlying cycle theory in the above scenario is to begin to, unsuccessfully, execute actions $A_1, \ldots, A_{n-1}$, since it does not know that their preconditions do not hold. While it is executing these actions, action A_n times out. Consequently the computee with the underlying cycle theory fails to achieve goal G'.

Therefore we conclude that in certain scenarios a computee with an actively cautious cycle theory succeeds where a computee with an underlying cycle theory fails.

5.1.4 The Punctual Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	DIPISA
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. A *punctual* computee selects urgent goals and urgent actions for planning and execution. This informal statement may be further refined according to two possible interpretations of the notion of urgency (in the following, the term *items* stands for both goals and actions):

1. Whenever planning or executing actions, select the items that are *more urgent* than the others (*simple punctual computee*).

2. If there are *very urgent* goals or actions, i.e. items close to their deadlines, give preference to planning and/or action execution rather than other transitions (*punctual computee*), possibly with a given preference amongst the two and determining the input items for the transition by choosing those that are more urgent, as in (1).

The notion of *deadline* has been introduced so as to impose an ordering actions and goals. The *more urgent items* are the maximal elements according to this ordering amongst those returned by the core selection functions. The *very urgent items* are those items, amongst those returned by the core selection functions, whose deadline is within a chosen time window from the current time.

Significance. The main advantage of this profile is not to let goals and actions become unachievable due to being timed-out. Moreover, the order induced by the notion of more urgent items can be used to guarantee temporal constraint satisfaction, for instance regarding the execution of actions. These features are expected to facilitate the satisfaction of (top-level) goals and to minimise failures due to a careless management of time, and temporal constraints.

As we mentioned in Section 2.3.2, while working on the Coherence Properties (Section 5.2) we came across a problem with the definition of the Action Selection Function, namely that it

allowed the selection of a set of actions with incompatible temporal constraints. In Section 2.3.2 we described how the definition can be corrected. The corrected definition, however, created one further issue, namely that an action could be selected although other actions with earlier times were available for selection. After many discussions we decided to allow for this in the core definition, as it was possible to construct examples where it would be advantagous to allow this out-of-order selection of actions. Instead, we decided to design the *punctual profile* in such a way that it not only solved the first problem mentioned above, but also ensured that actions are selected in the right order, according to their temporal constraints.

Formal description of the profile. The notion of *deadline* imposes an ordering on items that are not timed out, lifted from the order of their deadlines: $i[t_i] \rhd j[t_j]$ iff $now \leq d_i < t_j$ (an item i is more urgent than an item j if their deadlines are both not already timed out and the one for i is closer). The *more urgent items* are the maximal elements with respect to $\rhd$. The *very urgent items* are those whose deadline is closer than u time instants from now ($now + u \geq d$).

The profile has been characterised in terms of properties of the states that may occur in the operational traces. For instance, a trace of a simple punctual computee cannot have a state where an AE transition is performed without executing the more urgent actions, i.e. if there exists an i such that $T_i(S_{i-1}, Y_i, S_i, \tau_i)$ is a transition in the trace of a simple punctual computee and $T_i = AE$, then Y_i must contain the more urgent actions in the state of the computee.

Cycle theory. The simple punctual computee can be implemented by means of basic cycle rules, like

$$\begin{array}{l} Punctual_{T|AE}(S', Y): \\ \quad *AE(S', Y) \leftarrow \quad T(S, X, S', \tau), selected\_actions(S', X', \tau'), now(\tau'), \\ \qquad\qquad\qquad\quad more\_urgent\_actions(X', Y, \tau'), Y \neq \emptyset, \end{array}$$

which suitably restricts the selected actions to the more urgent ones (the predicate *selected_actions* implements the core action selection function). The punctual computee, other than rules like the one above, contains behaviour cycle rules, like

$$\begin{array}{l} Punctual^{T'}_{AE \succ T}(S', Y): \\ Punctual_{T'|AE}(S, Y) \succ \mathcal{R}_{T'|T}(S', X) \leftarrow \quad selected\_actions(S', X', \tau'), now(\tau'), \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad very\_urgent\_actions(X', Y, \tau', u), \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad Y \neq \emptyset, \end{array}$$

which, in the presence of very urgent actions, forces an AE transition. These rules are required to be the only ones defining a preference over AE and PI. Some care is needed to fix a priority amongst AE and PI. See [13], for further details. It has been proved that the traces induced by the punctual cycle theory fulfil the state properties mentioned above.

Advantages of the profile. It has been shown that, under given hypotheses and a suitable tuning of the time window width in the definition of very urgent items, a punctual computee is able to guarantee that the execution of the planned actions satisfies the temporal constraints. When planning and execution costs are taken into consideration, it may be useful to execute more actions, or plan for more goals, together in order to meet deadlines. This can be achieved by selecting the n items highest up in the order, instead of the only the maximal ones, as the more urgent items. This may spoil temporal constraints satisfaction.

Approach. This profile has been defined in order to improve the time responsiveness and the temporal constraint compliance of computees. This has been achieved by imposing a preference on the planning for urgent goals and the execution of urgent actions, with the aim of preventing goals and actions from becoming timed-out before they have been planned for or executed. Differently from the normal computee, a punctual computee can execute an AE or PI transition at any instant, whenever some actions or goals in its state are recognised as being urgent according to a suitable notion of urgency.

An experiment involving the punctual profile of behaviour is discussed in deliverable D14 [2].

5.1.5 The Impatient Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	UCY
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. An *impatient* profile of behaviour prevents a computee from executing an action, if an action of the *same type* was executed in the past *unsuccessfully*, that is, without producing the desired effect. We say that two actions $A_1 = \langle a_1[t_1], \_, \_\rangle$ and $A_2 = \langle a_2[t_2], \_, \_\rangle$ are of the same type iff $a_1 = a_2$. A more moderate version of this profile can be obtained by defining a time window after which the computee is allowed to execute the action. In our study of the profile, the computee never executes an action if an action of the same type was executed unsuccessfully in the past, unless no other transition is enabled.

Significance. The impatient profile will be particularly useful in environments where time is critical and we cannot afford to have the computee try executing actions when we know that they are likely to fail.

Formal description of the profile. The impatient profile is characterised by the following property of the operational trace:

Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of the computee such that $executed(a[t], \tau_i) \in KB_0^i$ for some action $A = \langle a[t], \_, \_\rangle$, where $S_i = \langle KB^i, Goals^i, Plan^i, TCS^i\rangle$, then

- either $KB^{i-1} \nvDash unsuccessful(a[t'])$
- or the only transitions possible from state S_{i-1} are of the form $AE(S_{i-1}, As)$, where for every action $A = \langle a[t], \_, \_\rangle \in As$, it holds that $KB^{i-1} \vDash unsuccessful(a[t'])$.

Where the predicate $unsuccessful(a[t'])$ is defined appropriately using the Temporal Reasoning capability of the computee and it means that an action $\langle a[t'], \_, \_\rangle$ was executed unsuccessfully in the past. This means that $executed(a[t'], \_) \in KB_0$, but the desired effects of the action do not hold at the current time and nothing happened between the time it was executed and the current time that could make the effects not hold.

In other words the computee does not execute an action if an action of the same type was executed unsuccessfully in the past, unless it has no other choice of transition.

Cycle theory. The impatient profile can be captured through the following suitable cycle theory where an action of the same type as an action which failed is given very low priority so that unless there is nothing else to be done, it will remain unexecuted and therefore it will time out.

An impatient cycle theory is *any* cycle theory where

- The following two rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:
  - $Impatient\_\mathcal{P}^T_{T1 \succ AE}$:
    $\mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As)) \leftarrow \exists A \in As, A = \langle a[t], \_, \_ \rangle, unsuccessful(a[t'])$
    for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$.
  - $Impatient\_\mathcal{MP}^T_{T1 \succ AE} : Impatient\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$,
    for every T and $T1$, such that $T1 \neq AE$.
- There is no rule which could enable $\mathcal{P}^T_{AE \succ T1} \succ Impatient\_\mathcal{P}^T_{T1 \succ AE}$ in the $\mathcal{T}_{behaviour}$ part of the cycle theory.

The purpose of the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ is to give lower priority over any other transition to transitions of Action Execution with input parameter a set of actions As when an action of the same type as an action in As was executed unsuccessfully in the past. The purpose of the rule $Impatient\_\mathcal{MP}^T_{T1 \succ AE}$ is to give the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ higher priority than any other priority rule which gives higher priority to an AE transition.

- Proposition 1: The proposed impatient cycle theory induces operational traces that satisfy the characteristic feature of the impatient profile.

Another definition of an impatient cycle theory is also given in [13] where we allow the computee to execute an action of the same type as an action which failed, if another action, which could affect its precondition, is executed.

Advantages of the profile. We call an *underlying (non-impatient) cycle theory* any cycle theory obtained from an *impatient cycle theory* by removing the extra rules which were added to the theory in order to achieve the impatient behaviour.

We examine the following propositions:

- Proposition 2: A computee with an *impatient cycle theory* is as good as a computee with an *underlying cycle theory* with the same knowledge base KB, in the sense that whenever the latter succeeds in achieving a goal, so does the former, in environments where the truth value of the preconditions of an action do not change by some unknown factor in the environment.
- Proposition 3: In certain cases a computee with an *impatient cycle theory* is better than a computee with an *underlying cycle theory* in that the first succeeds in achieving a goal while the second fails to achieve the same goal.

Approach. All propositions are proved in [13]:

- Proposition 1: We prove that the proposed cycle theory induces operational traces that satisfy the required feature by proving that there does not exist an admissible set of rules which concludes an action execution transition for some action of the same type as an action which was executed unsuccessfully in the past, if there are other transitions enabled.

- Proposition 2: We prove that a computee with an *impatient cycle theory* achieves any goal that a computee with an *underlying cycle theory* achieves in the given environment by showing that in the given environment a computee with the impatient cycle theory is able to reproduce the successful operational trace of a computee with the underlying cycle theory.

- Proposition 3: We provide the following example to demonstrate that in certain cases an impatient computee succeeds but a computee with an underlying cycle theory fails. Suppose that the two computees have exactly the same knowledge base and that the impatient computee executes an action A which belongs to a plan for some goal G. Let's say that the execution of A fails. According to the definition of the impatient cycle theory, the impatient computee will not try executing any action of the same type as A unless it has no other choice of transition or some action that could affect its preconditions is executed. Suppose that no such action is executed. Let's say that the computee executes next an action B which belongs to a plan for a goal G' where $G' \neq G$. The execution succeeds and goal G' is achieved.

  A possible operational trace for the computee with the underlying cycle theory in the above scenario is to execute action A unsuccessfully and then try executing some other action A' of the same type as A. While the computee with the underlying cycle theory is executing action A', action B times out, hence the computee with the underlying cycle theory fails to achieve goal G'.

  Therefore we conclude that in certain scenarios a computee with an impatient cycle theory succeeds where a computee with an underlying cycle theory fails.

5.1.6 The Careful Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	ICSTM
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. A computee endowed with a careful profile of behaviour examines and revises its current commitments frequently so as to recognise infeasible goals and actions (as well as goals that have been achieved already) as soon as possible.

Significance. The intuitive advantage of such a behaviour profile is that operations such as planning and reactivity are not hindered by superfluous items in the computee state.

Formal description of the profile. A careful computee is a computee that will never generate an operational trace with two consecutive transitions that are different from SR (State Revision). In fact, this condition is stronger than strictly necessary: as long as there are no redundant goals or actions no revision would be required. Nevertheless, from a pragmatic point of view, our definition provides us with an appropriate characterisation of careful computees. This is so, because *checking* whether or not a state includes redundant goals or actions to be revised is just as costly as performing a state revision in the first place.

Cycle theory. Any given cycle theory can be turned into a cycle theory for careful computees by first removing any basic rules that speak about two consecutive transitions both of which are different from SR and then adding the following basic rule for each transition T different from SR:

$$R_{T|SR}(S') : *SR(S') \leftarrow T(S, X, S', \tau)$$

Any such cycle theory is easily seen to induce the careful profile of behaviour.

As shown in [13], carefulness is in fact a very strong condition that requires an almost complete rewriting of a cycle theory. Arguably, the simplest approach is to ensure carefulness in behaviour purely by adding and deleting basic rules in the cycle theory appropriately. Behaviour rules can then be used to guide the computee's behaviour in aspects that go beyond the basic need to alternate SR with every other type of transition.

Advantages of the profile. The following two propositions (which hold under certain circumstances) demonstrate advantages of the careful profile of behaviour:

- Proposition 1: Careful computees will never generate a reaction via the reactivity transition to timed-out unachieved goals or timed-out unexecuted actions.
- Proposition 2: Careful computee will never generate a reaction via the reactivity transition to actions that may not be timed out yet but which are unexecuted and no longer necessary.

An example showing the advantage of the careful profile would be the following: Computee C believes that he has registered for a conference $conf05$ but wants not to be registered at the conference. He plans for the goal of not being registered and consequently generates an action in his $Plan$ to cancel his registration at $conf05$. He has a reactive rule in his KB_{react}that says:

> *If (observe that the deadline for cancellation for Conference has reached) and (an action of cancellation of registration at Conference is expected) then tell the bank to stop credit card payment to Conference.*

Suppose before the action of cancellation is executed the computee receives a message from the conference telling him that there was a problem with its initial attempt at registration and so he is not actually registered. So his goal of not being registered is achieved without the need for the cancellation action. A careful computee will not tell the bank to stop credit card payment (which is pointless anyway), but, under the same circumstances, a non-careful one might.

Approach. The above properties rely on the assumption that no action and no goal is timed out between an SR transition and its immediate successor if that is an RE transition. Details are given in [13].

5.1.7 The Focussed Profile of Behaviour

Property classification:	[IC-1] Behaviour profiles
Partners involved:	ICSTM
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. A *focussed* computee does not plan for more than one top-level goal at a time. It remains committed to the same top-level goal until that goal has either been achieved or is believed to be infeasible.

Significance. The advantages of the focussed profile come into effect in highly time-critical domains as well as domains where a computee has several goals for which no mutually compatible plans can be found. In such a situation, a focussed computee can be expected to achieve, at least, some of its goals, but an unfocussed computee may fail completely.

Formal description of the profile. A focussed computee is a computee that, under no circumstances, will generate an operational trace that includes a state with two top-level goals with children.

Cycle theory. Focussed behaviour can be achieved by adding to any basic rule in a computee's cycle theory that enables Plan Introduction an enabling condition that succeeds iff the selected goals all belong to the same top-level goal and no other goal in the state has got any children.

A cycle theory is called focussed iff the initial rule $r_{0|PI}$ and the basic rule $R_{T|PI}$ for any transition T include the enabling condition $\mathit{focussed}(Gs', S, Gs)$, where:

(i) S stands for the current state;

(ii) Gs' stands for the set of goals that is returned by the goal selection function and Gs stands for the set of goals to which PI will be applied; and

(iii) the predicate $\mathit{focussed}(Gs', S, Gs)$ succeeds iff $Gs \subseteq Gs'$ and all the goals in Gs are descendants of the same top-level goal (possibly including that top-level goal itself) and no other top-level goal has got any children.

We have shown that any focussed cycle theory induces a focussed profile of behaviour [13].

Advantages of the profile. If a focussed and a normal computee have a set of goals for which they have no compatible plans then the focussed computee may be able to achieve at least some of its goals while the normal computee may not be able to achieve any of the goals.

Note that this links the profile to the notion of welfare that is measured by the number of achieved goals (discussed in Section 5.3). Thus under the specified conditions the focussed profile provides the computee with an advantage in terms of its welfare.

Approach. In [13], we first introduce a characterisation of the focussed profile of behaviour that asserts that no state should ever include two distinct top-level goals with children, both of which are neither achieved nor infeasible. We then show how by insisting that the computee performs a State Revision before switching to a new top-level goal for planning, we can give an alternative state-based characterisation of the focussed profile that is much simpler: a computee is focussed iff it never induces a trace with a state where more than one top-level goal has got children. This definition is then naturally captured by the enabling condition given above.

The theorem demonstrating the advantage of the focussed profile, which is formalised and proved in the annex [13], shows under what conditions the focussed computee is guaranteed to achieve more of its goals compared to the normal computee. An experiment related to this property is discussed in deliverable D14 [2].

5.1.8 The Objective Profile of Behaviour

Property classification:	`[IC-1]` Behaviour profiles
Partners involved:	CITY
Relevant papers:	Athienitou et al. [13]
Status:	Proved

Informal description of the profile. A computee with an *objective* profile of behaviour always attempts to check immediately after the execution of an action, via Active Observation Introduction (AOI), if its desired effects are indeed established in the environment. The computee therefore attempts to obtain from the environment explicit information confirming the expected results of the execution of its actions.

Significance. This profile is suited to volatile environments where exogenous events can interfere with the execution and the result of an action. However, as AOI takes extra time to check the effect of actions, this profile may be less suitable for time-critical environments.

Formal description of the profile. Let $expected\_effects(S_i, X_i, \tau_i)$ be the set of fluents that are the expected effects of the execution of the actions in X_i in state S_i at time T_i.

Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of an objective computee with $T_i = AE$ and $expected\_effects(S_i, X_i, \tau_i) \neq \emptyset$, then there exists a transition $T_j(S_{j-1}, X_j, S_j, \tau_j)$ in the operational trace such that:

- $j > i$;
- T_j is an Active Observation Introduction, and $X_j = expected\_effects(S_i, X_i, \tau_i)$
- for all transitions T_k, $k = i+1, ..., j-1$, T_k is a POI transition.

Cycle theory. An objective cycle theory includes the following rules:

$$R_{AE|AOI}(S', X) : *AOI(S', X) \leftarrow AE(S, As, S', \tau), \\ X = expected\_effects(S', As, \tau), X \neq \emptyset$$
$$Objective\_\mathcal{P}^{AE}_{AOI \succ T'} : R_{AE|AOI}(S, X) \succ R_{AE|T'}(S, Y)$$

for every $T' \neq AOI$, where

$$expected\_effects(S', As, \tau) = \bigcup_{A=\langle a'[\_],G',\_\rangle \in X} Obj(S', a', \tau) \cup \{G'\} \backslash \{\bot\}$$

and

$$Obj(S, a, \tau) = \{f |\ S \models^{\tau}_{TR} initiates(a, \tau, f)\}\ \cup \{\neg f |\ S \models^{\tau}_{TR} terminates(a, \tau, f)\}$$

We have proved the following correpondence result:

- Proposition: The proposed objective cycle theory induces operational traces that satisfy the characteristic feature of the objective profile.

Advantages of the profile. In certain cases a computee with an objective cycle theory is better than a computee with a normal cycle theory in that the first succeeds in achieving a goal while the latter fails to achieve the same goal.

Approach. In [13], we prove that the proposed cycle theory, i.e. the normal cycle theory extended to capture the objective profile, induces traces that satisfy the required features. Our approach is based on argumentation-based preference reasoning. It uses preferences to define the behaviour of the objective profile and then proves that the normal cycle theory extended with the new preferences does have the above-mentioned property.

One type of example that demonstrates the advantage of the objective profile is one where the computee with the objective profile succeeds in achieving a goal because after executing an action towards it, it immediately gets feedback, through Active Observation, that the action has been unsuccessful, while there is still time to replan for the goal and try again. This is compared to the computee with the normal profile which will execute the action and will assume that it has been successful. When it finds out, through Passive Observation, some time later that the action has not been successful it is too late to replan for the goal as its time has passed.

Related work. Work related to behaviour profiles has been discussed in Section 5.1.1.

5.2 Coherence Properties

In this section we summarise our results concerning coherence properties of the model of individual computees developed in SOCS.

5.2.1 Coherence of Plan Introduction and Action Execution

Property classification:	`[IC-2]` Coherence properties of the KGP model
Partners involved:	ICSTM
Relevant papers:	Sadri and Toni [66]
Status:	Proved

Informal statement of the property. Computees do not attempt to execute actions that they believe are infeasible or unnecessary, and computees do not attempt to plan for goals if a plan is not needed or if it is too late to plan for them.

Significance. These properties show the suitability of the design of the KGP model in that computees are prevented from spending time on the activities of planning and acting if these activities are not appropriate or useful.

Formal statement of the property.

- Computees never attempt to execute actions that
  - are timed out, or
  - belong to a plan for a goal that is timed out or that they believe is already achieved.
- Computees never attempt to plan for a goal that
  - is timed out or that they believe is already achieved, or
  - belongs to a plan for a goal that is timed out or that they believe is already achieved.

Here, given a state, by plan for a goal we mean the set of all actions and goals that are descendents of that goal within the tree in that state.

Approach. Any cycle step rule showing that a transition is followed by Action Execution uses the action selection function in an enabling condition, and any cycle step rule showing that a transition is followed by Plan Introduction uses the goal selection function in an enabling condition. The required properties above follow from the definition of these two selection functions.

Related work. To the best of our knowledge no similar properties have been proven or identified in the literature.

5.2.2 Coherence of Action Execution with Temporal Incompatibility

Property classification:	`[IC-2]` Coherence properties of the KGP model
Partners involved:	ICSTM
Relevant papers:	Sadri and Toni [66]
Status:	Proved

Informal statement of the property. Temporally incompatible actions are never executed concurrently.

Significance. This result shows that computees will avoid concurrent execution of actions that they believe cannot be successfully executed together, because of the temporal constraints imposed upon them.

Formal statement of the property. If for some time τ and actions in a state S with operators $a_1[t_1], \ldots, a_n[t_n]$ there exists no total valuation of the variables $t_1, \ldots, t_n$ satisfying the temporal constraints in S, all equalities derived from KB_0 in S (as represented in $\Sigma(S)$) and $t_1 = \tau, \ldots, t_n = \tau$, then $a_1[t_1], \ldots, a_n[t_n]$ will never be all selected for execution at τ.

Approach. This property follows from the definition of the action selection function that will not select temporally incompatible actions at the same time for execution, given the constraint on their times.

Note that while proving this property a mistake in the definition of the action selection function was identified and corrected, as reported in Section 2.3.2, thus the proof has provided useful feedback about the model.

Related work. To the best of our knowledge no similar properties have been proven or identified in the literature.

5.2.3 Coherence of Action Execution with Incompatible Preconditions

Property classification:	[IC-2] Coherence properties of the KGP model
Partners involved:	ICSTM
Relevant papers:	Sadri and Toni [66]
Status:	Proved

Informal statement of the property. computee will never try to execute two actions at the same time if the actions have incompatible preconditions.

Significance. This result uses both the declarative and computational models of computees and shows that their design is such that computees will avoid concurrent execution of actions that they believe cannot be successfully executed together.

Formal statement of the property. Given the declarative and computational models of computees [36] if

- the $Plan^{nr}$ part of the computee state contains two actions $A1$ and $A2$ (i.e. $A1$ and $A2$ are non-reactive actions), and
- $A1$ has a precondition $p1$ and $A2$ has a precondition $p2$ such that the computee's KB_{plan} includes an integrity constraint

$$assume\_holds\_at(p1, T) \wedge \ assume\_holds\_at(p2, T) \rightarrow false$$

then the computee will never attempt to execute actions $A1$ and $A2$ at the same time.

Approach. The sketch of the proof is follows: As actions $A1$ and $A2$ are non-reactive actions they must have been generated in the computee state by a PI transition. The declarative and computational models of the computee ensure that the same PI transition which generates actions $A1$ and $A2$ will also generate a constraint $T1 \neq$T2 in the TCS part of the computee state where $T1$ and $T2$ are the proposed execution times of actions $A1$ and $A2$, respectively. Then the property follows from the proerty in Section 5.2.2.

Related work. To the best of our knowledge no similar properties have been proven or identified in the literature.

5.3 Individual Welfare-related Properties

Property classification:	[IC-3] Success criteria and preferences
Partners involved:	ICSTM, DIPISA, CITY, UCY
Relevant papers:	P. Mancarella et al. [52]
Status:	Preliminary results

Introduction. We consider several notions of individual welfare, some at a higher level of detail, in terms of top-level goals that are achieved and top-level goals that have become infeasible (timed out), and one at a lower level of detail in terms of incremental progress made towards achieving top-level goals. The former, referred to as the notion of *happiness*, considers only the achievement or unachievability of top-level goals, and the latter, referred to as the notion of *progress*, considers the whole state of the computee. We show some relationships between these different notions of welfare. We primarily consider a *subjective* approach to the notion of welfare, whereby achievement of a goal is assessed with respect to the computee's belief (knowledge base). We briefly address what might be needed in order to deal with an *objective* notion of welfare whereby achievement of a goal is assessed with respect to the computee's environment.

Informal statement of the properties. We define four preference relations over alternative computee states at a high level with respect to top-level goals:

- $S_1 \ll_1 S_2$ iff in S_2 at least as many top-level goals have been achieved as in S_1.
- $S_1 \ll_2 S_2$ iff in S_2 more top-level goals have been achieved than in S_1.
- $S_1 \ll_3 S_2$ iff in S_1 more top-level goals are infeasible (timed out) than in S_2.
- $S_1 \ll_4 S_2$ iff in S_2 more top-level goals have been achieved than in S_1, or in S_2 the same number of top-level goals have been achieved as in S_1 and in S_1 more top-level goals are infeasible (timed out) than in S_2.

Here by a goal being achieved we mean (roughly) that it is entailed from the computee KB_{TR} via the Temporal Reasoning capability.

We define a notion of $\ll$-improving, given any notion $\ll$ of preference between states. Informally, this notion states that a computee is $\ll$-improving iff given any state S in the sequence of states corresponding to any operational trace induced by the computee's cycle theory, there is a later state S' in the sequence such that $S \ll S'$. We have established the following properties (they all assume that no GI transition has been performed within the operational trace we are considering):

- Proposition 1: Any computee is $\ll_1$-improving.
- Proposition 2: No computee is $\ll_3$-improving.
- Proposition 3: A computee is $\ll_4$-improving iff it is $\ll_2$-improving.

We also define a notion of welfare in terms of "progress towards achievement of goals" and accordingly define another preference ordering $\prec$. The full definition is long and given in [52]. Informally and briefly, $S_1 \prec S_2$ iff in S_2 some progress has been made, for example by planning for a goal or by making "useful" observations.

- Proposition 4: The maximal state with respect to $\ll_2$ is the maximal state with respect to $\prec$.

Details of the orderings and welfare notions, formal definitions and the necessary underlying assumptions are given in [52].

Significance. Our notions of welfare are all intuitive ones and amount to assessing how effective a computee is in achieving its goals or at least in working towards achieving its goals. This work, therefore, is significant, in allowing us to analyse the effectiveness of the KGP model in these terms. Also by studying the environmental or internal conditions that would help, guarantee or hinder improvement of welfare, it may be possible to give guidelines to designers of computees.

Formal statement of the properties. The welfare notions and associated orderings are formalised and the propositions are stated formally and proved in [52].

Approach. We state the orderings using the formal language of the transitions and capabilities, and use their definitions to prove the properties.

Related work. As for the notion of *happiness*, which assesses the well-being of a computee in terms of the number of achieved (and possibly also unachievable) goals, there are clear connections to modelling the preferences of agents using utility functions [65]. Indeed, the number of achieved goals does induce a very simple kind of utility function. For our interpretation of individual welfare as *progress*, on the other hand, there are no direct links to such decision-theoretic concepts.

5.4 Adopting Social Expectations for Resource Allocation

Property classification:	`[IC-4]` Adopting social expectations
Partners involved:	UCY, UNIBO
Relevant papers:	Kakas et al. [46]
Status:	Preliminary results

Introduction. This property is an example for computees adopting social expectations as a means of achieving a desired outcome of interaction. We have studied the general issue of adopting social expectations in the context of a scenario where computees negotiate over resources, i.e. this property is also related to the class of *microeconomic properties* `[SI-4]`.

Informal statement of the property. In a resource reallocation domain, there exist social protocols and individual policies (within the Goal Decision knowledge base and cycle theories of computees) which can be combined to guarantee that a solution of a resource reallocation problem is found (if one exists). For example, a society may encode a protocol which produces tasks G that it expects (some of) its members to carry out. One such protocol could be the following:

$$\mathbf{H}(tell(A, B, request(Service)), T) \wedge authorized(A) \rightarrow \\ \mathbf{E}(tell(B, A, accept(request(Service))), T_1) \wedge T_1 < T + 20$$

Society expectations that are generated by such protocols are received by computees through Passive Observations. We assume that these are recorded in their KB_0 as facts of the form $observed(expectation(G), \tau)$ where G is the goal associated with the expectation received.

Computees then respond to the expectations through rules in their Goal Decision Knowledge Base, KB_{GD}, conforming to the schema:

$$r_G^{expect} : G \leftarrow holds\_at(expectation(G), T_{now})$$

which would thus generate an argument for G when an expectation for it is received or perceived by the agent. Then a *fully compliant* agent would have in its policy theory in KB_{GD} additionally the general priority rule

$$R^{comply} : r_G^{expect} > r_{G'}^{Label} \leftarrow Label \neq expect$$

for every such expectation rule and other private rule r_{Label} in the theory. This can be made conditional on specific circumstances for a non fully compliant policy. For example, an agent may receive an expectation from the society to accept unconditionally requests for a special type of *Needs*. Under the above schema this would enable the rule:

$$\begin{array}{l} r^{expect}(Peer, Needs) : \\ \quad msg(to(Peer), yes(Needs)) \leftarrow \\ \quad\quad holds\_at(expectation(msg(from(Peer), req(Needs))). \end{array}$$

Thus with the compliant priority rules this means that after the receipt of this expectation the computee will set itself the goal to reply to the request for these Needs.

Property. It is easy to see that computees with the R^{comply} rule in their goal decision policy will be *expectation conformant* for non-time critical expectations, in the sense that at some stage they will set themselves the goal of the expectation and possible attempt to satisfy this. When time is critical, we would need in addition to add in the behaviour part of the cycle theory an analogous priority rule, that gives to the transition of Goal Introduction higher priority over any other transition when $holds\_at(expectation(G), T_{now})$ is true.

Significance. This property demonstrates an advantage of having a social model (the one based on social integrity constraints) and a computee model (the KGP model) both using a declarative formalism and a similar underlying logic. The smooth combination of social expectations and internal decision making is significant for collaborative agent systems and in open multiagent systems, where no central authority can impose choices on the operations of individuals. The society in this context assumes the role of a driving engine, which suggests decisions to be taken by computees in reply to requests, but which does not directly force them to act in some way rather than in some other way.

Approach. This is still preliminary work. In order to approach the issue of computees adopting social expectations, we have defined a simplified computee architecture [46], and we have defined a cooperative computee as a computee that is willing to give away a resource for nothing in exchange. In addition, we have defined a specific cycle ("philagentic" computee cycle of operation), which implements an exhaustive search in the space of solutions of a resource

reallocation problem. The control of operation of "philagentic" computees allows them to find a combination of choices of plans which allows for negotiation dialogues which solve the resource reallocation problem and to produce the actual dialogues themselves.

Finally, we have defined the behaviour of computees in such a way that they do not consume their resources until a dialogue has successfully terminated, thus preventing computees from following a plan which consumes resources needed by the other computee, while they could follow instead some alternative plan which accommodates both computees' needs.

The decision of a computee to be cooperative or non-cooperative may depend on the existence of relevant social expectations.

Related work. To the best of our knowledge, ours is the first declarative approach that combines agent decision making with social goals while preserving agent autonomy. However, there are of course several logic-based frameworks for agent negotiation related to our work that have been proposed elsewhere [10, 67, 69, 49].

6 Social Infrastructure

In this section we present our results pertaining to properties of the social infrastructure of a society of computees.

6.1 Well-definedness of Societies

Property classification:	`[SI-1]` Well-definedness of societies
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Alberti et al. [5]
Status:	Proved

Introduction. When we define a society in terms of $SOKB$, $\mathcal{IC}_S$ and goal G, we would like to be sure that, under certain assumptions, this society will be *well-defined*, i.e. among its possible (closed) instances there exists at least one for which goal G is achieved (see deliverable D8 [36] for the definition of goal achievement). For instance, a society with goal p and $SOKB$:

$$p \leftarrow \mathbf{E}(q), \mathbf{E}(k)$$

and $\mathcal{IC}_S$ given by:

$$\mathbf{E}(k) \rightarrow \mathbf{EN}(q)$$

is not well-defined with respect to goal p. This means that there is no way to achieve p in such a society, even when its members behave according to the society expectations.

Informal statement of the property. To illustrate the property, consider a simplified case where the goal G is *true*. Whenever the goal is *true*, the notion of well-definedness amounts to stating that there exists at least one course of events (history) fulfilling an (admissible) set of social expectations $\mathbf{EXP}^\star$. If a society is not well-defined with respect to goal *true*, then there is no way (i.e. no course of events) for its members to behave according to the raised set of social expectations $\mathbf{EXP}^\star$.

For instance, a society with goal *true*, empty $SOKB$ and $\mathcal{IC}_S$ given by:

$$\mathbf{H}(p) \rightarrow \mathbf{EN}(p)$$

$$\neg\mathbf{H}(p) \rightarrow \mathbf{E}(p)$$

is not well-defined with respect to *true*, since there does not exist any admissible and fulfilled set of social expectations for each of its (closed) instances. In fact, in any closed instance of such a society, if $\mathbf{H}(p)$ belongs to the history, then $\mathbf{EN}(p)$ has to be abduced, but this leads to violation. Also, if $\mathbf{H}(p)$ does not belong to the history, then $\mathbf{E}(p)$ has to be abduced, but this leads to violation as well.

Significance. It is important to identify the societies for which there cannot exist any compliant history. Such societies are ill defined, since there is no way to achieve the society requirements (goals) even if members interact according to the protocol specified by the society. The well-definedness of a society is a useful guideline for the society designer; the verifiability of the well definedness is thus an important feature of the SOCS social framework.

Formal statement of the property. We define well-definedness of a society with respect to a goal as follows:

Definition 6.1 (Well-definedness wrt goal) *A society $\mathcal{S}$ with ground goal G is well-defined with respect to G iff there exists a closed instance $\mathcal{S}_{\overline{\mathbf{HAP}}}$ of $\mathcal{S}$ such that G is achieved, i.e.:*

$$\mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^\star} G$$

We call $\overline{\mathbf{HAP}}$ a compliant history with respect to society $\mathcal{S}$ and goal G.

Notice that Definition 6.1 implies that there exists a (closed) admissible and fulfilled set of social expectations $\mathbf{EXP}^\star$, such that:

$$SOKB \cup \overline{\mathbf{HAP}} \cup \mathbf{EXP}^\star \models G$$

The following proposition relies on the generative variant of $\mathcal{S}$CIFF, called g$\mathcal{S}$CIFF, which has been defined and implemented in order to generate compliant histories [5]. Whereas in usual $\mathcal{S}$CIFF executions the history is generated by external entities (e.g. by the communicative acts exchanged by computees), in a g$\mathcal{S}$CIFF computation a (compliant) history is generated by the proof procedure itself.

Proposition 6.1 (Well-definedness) *A society $\mathcal{S}$ with ground goal G is well-defined with respect to G if there exists a* g$\mathcal{S}$CIFF *computation such that:*

$$\mathcal{S}_\emptyset \overset{g}{\underset{\mathbf{EXP}}{\vdash}}{}^{\mathbf{HAP}^f} G$$

By Proposition 6.1, we determine the well-definedness of a society $\mathcal{S}$ with respect a to goal G by relying upon g$\mathcal{S}$CIFF.

If there exists a g$\mathcal{S}$CIFF computation for G in $\mathcal{S}$, starting with the empty history set and leading to a final history $\mathbf{HAP}^f$ and (positive and negative) expectations $\mathbf{EXP}$, then society $\mathcal{S}$ is well-defined with respect to G.

Approach. As with the other properties, the well-definedness property has been dealt with within the project by following a proof-theoretic approach, and by exploiting a variation of the $\mathcal{S}$CIFF proof procedure defined for the society infrastructure component. This has been obtained by replacing the $\mathcal{S}$CIFF Fulfilment transition by a new one (called Fulfiller) which, in practice, turns the pending expectations into events, therefore applying the following meta integrity constraint:

$$\mathbf{E}(p) \rightarrow \mathbf{H}(p)$$

Our proof of Proposition 6.1 above is based on the soundness of g$\mathcal{S}$CIFF [5].

Related work. Since the well-definedness property is a particular instance of protocol properties, we refer to Section 6.2 for a discussion of related work.

6.2 Automatic Verification of Protocol Properties

Property classification:	`[SI-2]` Automatically verifiable properties
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Alberti et al. [5]
Status:	Proved (for restricted cases)

Introduction. Protocols are specifications of social expectations defined by the Society Knowledge Base and the $\mathcal{IC}_S$ parts of the social infrastructure. In this section we describe our general approach to proving protocol properties automatically. In particular we describe how the $\mathcal{S}$CIFF proof procedure can be used and further adapted to this end.

Informal statement of the property. We aim at verifying protocol properties that can be expressed by formulae and, in particular:

- *existential* properties, i.e. formulae that hold for at least one history compliant to the protocol;
- *universal* properties, i.e. formulae that hold for all the histories compliant to a protocol.

Significance. Agent interaction verification is one aspect which has been intensively studied under many perspectives in the area of open computational systems. The capability of the SOCS social model to automatically prove or refute significant protocol properties is a demonstration of the expressiveness of the formal model and of the effectiveness of its operational counterpart. More generally, it shows the applicability of our approach to problems that, so far, have been tackled mainly by model checking techniques.

Formal statement of the property. Let a protocol $\mathcal{P}$ be defined by $SOKB$ and $\mathcal{IC}_S$. By $\mathcal{S}$, we design the society defined by $SOKB$ and $\mathcal{IC}_S$. In the following definitions, we assume that $\mathcal{S}$ is well defined with respect to the goal *true*. A formula f is an existential property of $\mathcal{P}$ iff:

$$\exists_{\mathbf{HAP}} \exists_{\mathbf{EXP}^\star} \mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^\star} f \tag{1}$$

where $\mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^\star} f$ has the meaning explained in Definition 2.8 (goal achievement).

A formula f is a universal property of the protocol $\mathcal{P}$ iff:

$$\forall \mathbf{HAP} \forall \mathbf{EXP}^{\star} (\mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^{\star}} \mathit{true} \rightarrow \mathcal{S}_{\overline{\mathbf{HAP}}} \models_{\mathbf{EXP}^{\star}} f) \tag{2}$$

Approach. Our approach, which is described in detail in [5], aims at proving or refuting properties automatically. For this purpose, we have extended the $\mathcal{S}$CIFF proof procedure, used for on-the-fly conformance testing, with a capability to generate histories that are compliant to a given protocol and to a $\mathcal{S}$CIFF goal (see Sect. 6.1). This extended proof procedure (g$\mathcal{S}$CIFF) can be used for verifying both existential and universal properties, as follows.

- An existential property f can be verified by:
  1. expressing f as a $\mathcal{S}$CIFF goal, and
  2. running g$\mathcal{S}$CIFF. Two cases are possible:
     - g$\mathcal{S}$CIFF returns failure: f is not an existential property of protocol $\mathcal{P}$;
     - g$\mathcal{S}$CIFF returns success, with a history **HAP**: f is an existential property of $\mathcal{P}$, and **HAP** is an example instantiation of a history that satisfies f.
- A universal property f can be verified by:
  1. expressing $\neg f$ as a $\mathcal{S}$CIFF goal, and
  2. running g$\mathcal{S}$CIFF. Two cases are possible:
     - g$\mathcal{S}$CIFF returns failure: f is a universal property of protocol $\mathcal{P}$;
     - g$\mathcal{S}$CIFF returns success, with a history **HAP**: f is not a universal property of $\mathcal{P}$, and **HAP** is an example history for which f does not hold.

At the time of writing, our technique can be applied subject to the following restrictions:

- The only properties that we can verify are
  - existential properties that can be expressed as a $\mathcal{S}$CIFF goal;
  - universal properties whose negation can be expressed as a $\mathcal{S}$CIFF goal.
- Soundness of g$\mathcal{S}$CIFF has been proven, but completeness has not. Thus, our approach is provably effective for proving existential properties and refuting universal properties, but not yet for proving universal properties and refuting existential properties.

Related work. In recent years the provability of properties for communication protocols has received a lot of attention; this holds even more for security protocols. Various techniques have been adopted for the task of automatic verification of properties.

One way to prove/disprove protocol properties in the security domain is the cryptographic approach, used for proofs by hand [37] or, more recently, automatically [14]. Theorem provers, such as Isabelle/HOL [58] have also been applied to this task, together with tools for graphically representing and defining the protocols [76]. Another viewpoint is to embody a possible intruder and plan for an attack [1].

Dixon et al. [21] specify security protocols in $KL_{(n)}$, a language for representing the *Temporal Logic of Knowledge*. Raimondi and Lomuscio [62] also use a temporal logic enriched with epistemic connectives for representing the agents' knowledge, but exploit efficient data

structures (namely, Ordered Binary Decision Diagrams) to improve the efficiency of the model checking algorithm. The g$\mathcal{S}$CIFF procedure is able not only to prove properties expressed in the $\mathcal{S}$CIFF language, but can also provide counterexamples of properties. For instance, g$\mathcal{S}$CIFF is able to generate the Lowe's attack to the Needham-Schroeder protocol [56, 51, 5].

Armando et al. [11] compile a security program into a logic program with choice lp-rules with answer set semantics.[3] Among other approaches to security protocol verification we cite those developed using hereditary Harrop formulas [18], process-algebraic languages [59], model checking with pre-configuration [48], and proof theory [19].

6.3 Negotiating Socially Optimal Allocations of Resources

Property classification:	`[SI-3]` Microeconomic properties
Partners involved:	ICSTM, CITY
Relevant papers:	Endriss et al. [26, 28], Endriss and Maudet [25]
Status:	Proved

Introduction. This strand of work brings together two central aspects of SOCS: distributed resource allocation and the notion of society. In the context of resource allocation problems, the welfare of an individual computee may be measured in terms of the resources it owns and the utility it assigns to these resources. The concept of a *social welfare ordering*, familiar from welfare economics [12, 55], then allows us to compare different states from a societal point of view. We have studied how allocations of resources that are socially optimal in this sense can be guaranteed to emerge eventually, provided that the behaviour of individual computees is restricted by certain rationality constraints when they engage in negotiation and agree on a sequence of deals to exchange resources.

Informal statement of the property. Our results in this area are parametrised by (i) the class of *deals* that computees may negotiate, (ii) the notion of *social optimality* with respect to which we evaluate allocations of resources, and (iii) possible restrictions on the *utility functions* used to measure individual welfare. Our main results may be paraphrased as follows:[4]

- Computees that will agree to any deal strictly improving their personal welfare will eventually reach an allocation with *maximal utilitarian social welfare*, i.e. an allocation where the sum of utilities is maximal (provided that monetary side payments are possible).

- In domains with *additive utility functions*, the above result holds also for deals involving only a single resource at a time.

[3] *Choice lp-rules* (for a formal definition, see [57]) are expressions of the form

$$\{l_0\} \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n,$$

where $l_0, \ldots, l_n$ are literals, which prescribe that if their body is satisfied by an answer set, their head may or may be in the answer set, or not.

[4] Note that a crucial feature of our negotiation framework is that deals may be truly *multilateral:* a single deal may include any number of computees and any number of resources. Unless specified otherwise, monetary side payments between computees (to compensate someone for an otherwise disadvantageous deal) are assumed not to be possible. For precise definitions of the various social welfare orderings and types of utility functions involved, we refer to the textbook by Moulin [55] and papers [25, 26, 28].

- In domains with *0-1 utility functions* (that is, utility functions that are additive and where any single resource has either utility 0 or utility 1), the same result can be achieved without side payments (provided computees do not require a *strict* increase in utility).

- Computees that will agree to any deal improving (or at least maintaining) their personal welfare will eventually reach an allocation that is *Pareto optimal*, i.e. where no computee could enjoy a further gain without reducing the utility of another.

- Computees that will agree to any deal improving the welfare of their trading partner currently worst off will eventually reach an allocation with *maximal egalitarian social welfare*, i.e. an allocation that maximises the utility of society's "poorest" member.

- In domains with *0-1 utility functions*, computees that will agree to any deal involving only a single resource that is either inequality-reducing or results in an improvement for everyone involved will eventually reach an allocation that is *Lorenz optimal* (a notion of social optimality combining ideas from both the utilitarian and the egalitarian programme).

Significance. This line of research provides a theoretical analysis of the resource allocation problem in open societies, which has been identified as a central scenario of interest in SOCS and has also received much attention within the Global Computing initiative in general. Our results can assist system designers in building computees that are, at least in principle, capable of negotiating resource allocations that are optimal with respect to a notion of social welfare appropriate for the application in question. Although, in general, acceptability criteria (i.e. criteria that specify what deals would be acceptable to all parties) are best expressed in terms of utility functions, in many cases they can also be translated into symbolic negotiation strategies expressed as part of a computee's reactivity knowledge base. The simplest examples are 0-1 utility functions, which can be modelled using a predicate `need/1` that would be *true* whenever the resource in question has utility 1 [68].

Formal statement of the property. These properties (as well as several related results, pertaining in particular to the *necessity* of certain classes of deals for achieving optimal outcomes) are formally stated and proved in [26, 28, 25].

Approach. We first prove termination results for each of the instances of the general framework considered. These proofs rely on the fact that the negotiation space is finite (a finite number of discrete resources needs to be distributed amongst a finite number of computees) and the fact that for each of the classes of deals considered any one deal is bound to result in a strict improvement with respect to a suitable social welfare ordering. It is then possible to show that the assumption that negotiation has terminated (in the sense of there being no more feasible deals available to computees) and that the final allocation is not optimal leads to a contradiction. In each case, we show how these assumptions can be used to construct a particular deal that would still be feasible (which contradicts the termination assumption).

In some cases our contribution consists mostly of designing suitable classes of deals for a given social welfare ordering, and the actual proofs are relatively easy. In other cases, the connections between the social welfare orderings and the classes of deals are more complex.

Related work. Our work in this area builds on and extends work by Sandholm on distributed task allocation [70]. In fact, our results for the first instance of the negotiation framework, where strictly self-interested computees negotiate deals with monetary side payments and where we are interested in maximising utilitarian social welfare, directly corresponds to Sandholm's framework (if we interpret tasks as resources with negative utility).

The utilitarian programme, i.e. the idea of trying to maximise the sum of all utilities of the members of a society, is often taken for granted in the multiagent systems literature (see e.g. [53, 64, 71, 77]). This is not the case in welfare economics and social choice theory, where different notions of social welfare are being studied and compared with each other [12, 39, 55, 73]. Our work on negotiation addresses this gap by introducing notions such as egalitarian social welfare into multiagent systems.

While most work on negotiation in multiagent systems has been concerned with either *auctions* or *bilateral* ("one-to-one") negotiation [64, 77], we should stress that our work explicitly addresses *multilateral* exchanges and that it is *not* an auction. Auctions are mechanisms to help agents agree on a price at which an item (or a set of items) is to be sold [47]. In our work, on the other hand, we are not concerned with this aspect of negotiation, but only with the patterns of resource exchanges that computees actually carry out.

6.4 Stability Properties

6.4.1 Stability Properties of Multiagent Systems

Property classification:	`[SI-4]` Stability properties
Partners involved:	ICSTM, DIPISA, CITY
Relevant papers:	Bracciali et al. [15, 16]
Status:	Preliminary work

Introduction. This line of work has been undertaken with the aim of investigating and devising a logic-based, declarative methodology for the study of properties of interacting computees, emerging from their mutual interactions and from their interactions with the environment in which they are situated. The methodology was designed to be abstract, namely focussing on the external, observable behaviour of computees rather than their internal reasoning processes, and possibly supporting automated verification.

We have introduced a novel semantics for generic multiagent systems (MAS), based on the idea of *stable sets* of actions [15, 16]. Agents are considered as input-output transformations from their "private" mental state and the environment they perceive to the observable set of actions they perform in their environment (which are observed by the other agents). A MAS is then understood in terms of the collection of the agent semantics and the environment in which the agents are situated. It is then possible to characterise the "good" evolutions of a MAS as those that lead to a "stability point". The notion of stability has been formalised via a recursive notion of a *stable set* of actions, i.e. the union of all the actions that the agents in the system would perform, if each one of them knew, from an oracle, say, what all the others will do and what will happen in the environment. If such a set exists, then it represents a convergence point where all the agents agree on what the overall behaviour of the system is.

This provides us with a significant set of system evolutions, viz. those corresponding to stable sets, in which it is sensible to study the properties of interest to the agents and the MAS. MAS can then be characterised in terms of further properties, such as their robustness, that can be meaningfully defined in terms of stable sets.

The semantics of stable sets is *abstract*, since it sees agents abstractly as input-output black boxes (and thus, in principle, it can be applied to any agent model), and since it abstracts away from temporal issues and the time of actions and events. It also has an *operational counterpart*, which, in some cases, can support automated verification.

"Well-behaved" MAS can be characterised according to the existence of stable sets for them. Furthermore, stable MAS can be further verified against additional properties of interest, related to either the system as a whole (see below) or to individuals within the system (see section 6.4.2).

As a first step towards using the abstract stable semantics in systems consisting of computees, we have instantiated the abstract stable semantics with simple abductive logic agents, equipped with simple planning and reactive knowledge in the form of abductive logic programs. The simple framework considered ignores temporal issues, namely it ignores the time of actions and events and the reasoning with it. We envisage that, in order to deal with the full KGP model, we will need to extend the abstract stable semantics to deal explicitly with temporal issues, so that the evolution of knowledge by computees (their observation of events in the environment and actions by other computees) is taken into account - when building a stable set - at the appropriate times.

Furthermore, in order to deal with *societies* of computees, rather than merely sets of computees, we will need to extend the abstract stable semantics to take into account social rules and goals, external to the agents.

Informal statement of the property.

- A set of actions (by the different agents) is *stable* if, assuming that an "oracle" could feed each of the agents with all the actions in the set performed by the other agents (and all events happening in the world), then each agent would do exactly what is in the set, namely their observable behaviour would be exactly what the set envisages.

In [15] we have given a bottom-up constructive method for the approximation of stable sets for a restricted class of agents (see above), and we have soundness for non-failing agents has been shown.

The problem of termination in presence of "temporarily" failing agents has been addressed in [16], where the bottom-up constructive method has been adapted to deal with abstract agents which may become inconsistent with respect to a given environment. In that case, the stable set is constructed, if it exists, by coordinating the agents that are able to maintain their consistency while acting together, and by suspending the inconsistent ones.

Relying upon the semantics stable sets, in [15] we have characterised *overall successful*, *robust*, and *world dependent* MAS:

- A system is *overall successful* if a stable set exists, according to which all the agents in the system are *individually successful*. Here, we have assumed that individual success amounts to achievement of individual goals by the agents. Note that this notion of overall success is rather weak, as it only requires *one* successful stable set to exist, and stronger versions could also be of interest.

- A system is *robust* if it is overall successful and there is no agent in it that can make the system not overall successful by not contributing to the evolution of the system itself. Namely, a robust system is one that does not need any of its agents in particular in order to be overall successful, or, alternatively, one in which no agent specifically needs one of

the others in order to be individually successful. Stronger notions of robustness, e.g. up to a given number of potentially "failing" agents, could also be of interest.

- A system is *world-dependent* if it is overall successful only within a non-empty world (i.e. environment) where events that are external to the agents happen. Intuitively speaking, a robust system does not depend on all the agents that build it, while a world-dependent one depends also on the environment, and not only on all its agents, in order to be overall successful.

These properties are verified in equilibrium points of the possible evolutions of a MAS, i.e. those that the agents in the MAS can reach by executing a stable sets of actions.

Significance. The declarative semantics introduced and its use for formal verification address the description and verification of properties of interest for multiagent systems. Such systems (intended as collections of agents) are simpler than societies of agents, the latter being provided with a social kind of infrastructure, like the normative one in societies of computees. However, even in this "simpler" formulation, the approach covers a set of properties of interest for communities of interacting agents and computees. Moreover, the high level of abstraction facilitates the independence of the model from the different agent paradigms. General conditions guaranteeing that verification can be fully automated are still under investigation. Restrictions facilitating the fruitful application of the model to the KGP model of computees are also under investigation.

Formal statement of the property. The model and the verification of the properties of interest have been introduced in [15], together with an automated verification methodology that has been shown sound for a restricted class of agents, when they are able to remain coherent with their evolving environment. In [16], the methodology has been extended to the case in which agents may become inconsistent, given the current state of the world, but might recover thanks to the activities of the other agents.

The model has been formalised. Properties of the automated verification methodology for restricted classes of agent programming languages have been shown. Conditions for more general soundness results and use of this approach for fully-fledged computees are subject to future work.

Approach. This work has been undertaken with the aim to devise an abstract, declarative semantics for interacting agents, with an operational counterpart allowing for the possibly automated verification of the properties of such systems. The approach has taken inspiration from a multidisciplinary set of fields.

Related work. The model we defined is close to several approaches, based on computational logic, whose aim has been to provide formal models to understand MAS environments, for example [17, 7, 8]. The distinguishing feature of our approach with respect to these proposals is its generality, which, independently of specific agent paradigms, allows us to reason at an abstract and "declarative" level, similarly to [75], where agents are considered in terms of their observable interaction with the environment. Also, we have made novel use of well known logic-based techniques, like bottom-up approximations and the idea of stability itself, as tools aimed at the analysis of MAS.

We also differ from several approaches based on modal logic, like [78], for both the overall approach and the interpretation of the concept of environment.

6.4.2 Stability Properties of Agents in Multiagent Systems

Property classification:	`[SI-4]` Stability properties
Partners involved:	ICSTM, DIPISA, CITY
Relevant papers:	Bracciali et al. [15, 16]
Status:	Preliminary work

Introduction. These properties use the semantics of stable sets of actions given in section 6.4.1, but they refer to individual agents and their role in a (stable) multi-agent system (MAS), rather than the system as a whole. In this sense, they can be seen as regarding the social effects of the agent behaviour. Thus, there are also connections to `IC-3` (success criteria and preferences).

As already discussed in section 6.4.1, the stability semantics for MAS and thus all the properties defined in terms of it, refer to an abstract agent model and neglect the possible presence of social rules and goals in MAS. Thus, future work will be needed to instantiate them for computees and societies of computees.

Informal statement of the property. This class of properties is defined by adopting the notion of *individually successful* agent as an agent that achieved its goals.

The following properties of agent within MAS have been studied:

- an agent within a MAS is *world aware* if it believes, within its mental state, all the events that have happened in the world and that it has observed;
- an agent within a MAS is *j-aware* if it believes all the observations it made upon some other agent j;
- an agent within a MAS it is *environment aware* if it believes everything it observes, including events in the world and actions by all the other agents it can observe.
- an agent within a MAS is *system dependent* if it cannot be successful alone, but it can be successful together with other agents in the system. Thus, this agent has a motivation to look for other agents with which to join forces;
- an agent within a MAS is *dispensable* if it is not needed to guarantee success of the other agents in the system. So, designers of multi-agent systems could exclude any dispensable agent from it (e.g. to reduce communication costs);
- an agent within a MAS is *dangerous* if it can undermine the overall success of a multi-agent system, if added to it. So, designers of a multi-agent system should make sure that no dangerous agent belongs to the system.

The study of the relationships between the properties of single agents within MAS, illustrated here, and those of agents within a MAS or a society, illustrated in section 6.4.1, are scope for interesting future work. For instance, a system dependent agent will probably be world or environment aware, and a robust system will be populated by dispensable agents, while it

obviously will not tolerate dangerous agents. Better understanding this kind of relations should facilitate the choice and the design of agents/computees depending on the desired features of the overall system.

Whenever the construction of the semantics can be supported by automated procedures, the verification of the properties above can also be supported by automated procedures.

Significance. The declarative semantics introduced is aimed at the the formal verification of properties of interest for agents within the systems they constitute. Mainly, the approach is oriented to MAS properties, as illustrated in Section 6.4.1, but suitable to also describe single agent features. Moreover, its high level of abstraction facilitates the instantiation of the model to different agent models. For some specific agent model the verification can be automated. More general conditions on the agent models, including the KGP model as already mentioned, which guarantee automatic verification are under investigation.

Classifying agents and computees according to their behaviour and attitudes towards other agents is an important research issue. Results in this field can help the design of agent-based applications by allowing system designers to determine the most suitable features an agent must exhibit according to its planned use.

Formal statement of the property. The properties have been formally introduced in [15] together with the automated verification methodology for a restricted case of agents. The companion paper [16] extends the approach to agents that may temporarily become inconsistent, i.e. may undergo a failure which might be recovered in the future.

The model has been formalised. Properties of the automated verification methodology for restricted classes of agent programming languages have been shown. Conditions for more general soundness results are subject to future work.

Approach. The model proposed has been devised with the aim of defining an abstract semantics suitable to be used for the possibly automated verification of properties of MAS and, with future work, societies of computees. The work has been inspired by a multidisciplinary set of ideas: the game semantics, for the input/output abstract semantics of agents, and computational logic, for both the idea of stable semantics for abduction that has inspired our notion of stable set, and bottom-up approximations, like the $\mathcal{T}_P$ operator of logic programming, of which our constructive methods can be seen as an instance. Moreover, our idea of stability as an equilibrium point appears strongly reminiscent of the Nash equilibrium, from economics (although this similarity needs to be further studied).

Related work. See Section 6.4.1.

7 Protocol Conformance

A *protocol* specifies the "rules of encounter" governing the social interactions between agents [64, 53]. It specifies which agent is allowed to do what in a given situation. It will usually allow for several alternative actions in every situation and the agent in question has to choose one depending on its private *strategy*. A good protocol will enable fruitful interaction in general. A good strategy will benefit the agent using it. The protocol is *public*, while each agent's strategy is *private*. We specifically consider protocols that regulate communication. Protocols are of

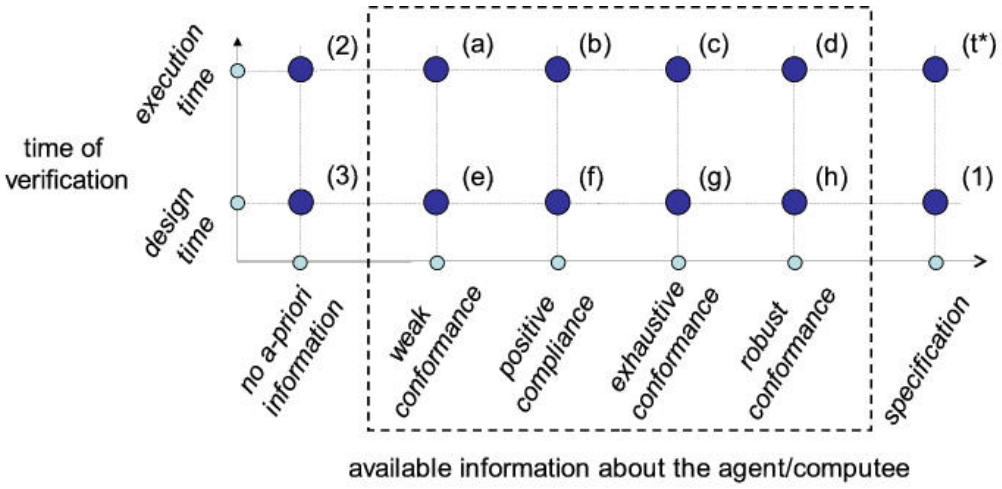

Figure 1: Time/knowledge conformance verification diagram

practical importance because they may help to select the adequate answer to an incoming utterance, thus reducing the complexity of this task for an agent. Moreover, from a social perspective, formal protocol specifications can be used to define what a "good" behaviour of an agent is, in the context of a given society, towards the achievement of a social goal.

In recent years, several formalisms for defining agent interaction *protocols* have been proposed and several notions of *conformance* (or compliance) to such protocols have been defined. Section 7.1 provides an overview of these different concepts to allow us to place the technical contributions reported later in Section 7 within the wider context of this area.

7.1 Aspects of Protocol Conformance

During the first two years of the SOCS project we have introduced a formalism, called *Social Integrity Constraints*, that can be used to define protocols [54, 36]. The notion of compliance to protocols has then been mapped onto that of *fulfilment of social expectations*. In parallel, within SOCS, work has been carried out towards identifying several *levels of protocol conformance*, towards checking a computee's expected conformance *a priori* (on the basis of its internal specification), and towards actively *enforcing conformance* by manipulating a computee's program [27]. Related to this latter line of enquiry, we have also introduced the concept of a computee's *competence* to use a protocol [22].

Figure 1 provides an overview of the different notions of protocol conformance we may consider. It is inspired by the work of Guerin and Pitt [38], who have identified three types of conformance verification, depending on the amount of information that is available about the participating computees, and on whether the verification is done at execution time or at design time. In this diagram, the horizontal axis shows the amount of information available about the computee, and the vertical axis represents the time of verification. If we have "static" information about the computee, such as its specifications, it will be possible to carry out some analysis *a priori* (at design time), and to foresee the behaviour of the computee (marked by (1) in the diagram). Conversely, if we have no information about the computees, we can only

perform *on-the-fly* verification of conformance to a protocol at execution time (2), by observing their behaviour from the outside. Finally, it is possible to study the protocol itself and its properties (3); this is usually done under the assumption that computees will actually follow that protocol.

In Figure 1, we indicate by (t^*) a fourth case where the specifications of computees are available, as well as their runtime behaviour. From the viewpoint of formal property verification, the verification of the computees' compliance and the verification of protocol properties can be done statically (case (1)); the additional information that we could obtain by running the system could be related to the task of testing it (this is discussed in deliverable D14 [2]). For instance, we may test how the properties that a computee exhibits relate to the overall system behaviour in terms of scalability, efficiency, etc.

Having no information about computees and the availability of a full specification are two extremes; we may also consider the availability of different degrees of knowledge about computees in between these two extremes. For instance, it could be the case that we know about some computee that it is going to be either "weakly", "exhaustively" or "robustly" conformant to a protocol [27], or that it is going to be either "positively", "negatively" or "strongly" compliant to it [6]. These notions are related to each other, and in some cases an equivalence between them can be established. Endriss et al. [27] provide the following definitions of conformance:

- A computee is *weakly conformant* to a protocol $\mathcal{P}$ iff it never utters any illegal dialogue moves (wrt. $\mathcal{P}$).
- A computee is *exhaustively conformant* to a protocol $\mathcal{P}$ iff it is weakly conformant to $\mathcal{P}$ and it will utter *some* response for any legal input it receives.

An additional notion of conformance (*robust conformance*) is related to the behaviour of computees in the event of illegal incoming messages. This latter view of conformance is related to the `not-understood` performative of the FIPA communication act library specification [31].

The SOCS social framework developed in D5 and D8 [54, 36] provides a language for defining protocols via a set $\mathcal{IC}_S$ of Social Integrity Constraints, rather than explicitly specifying the valid utterances (and thereby considering all the other possible utterances invalid). A protocol defined through this language does not over-constrained computees in their interaction. Alberti et al. [6] provide the following definitions:

- A group of computees is *negatively compliant* to a set $\mathcal{IC}_S$ of Social Integrity Constraints iff its members never produce a social event which is expected not to happen.
- A group of computees is *positively compliant* to a set $\mathcal{IC}_S$ of Social Integrity Constraints iff its members never fail to produce social events which are expected to happen.
- A group of computees is *strongly compliant* to a set $\mathcal{IC}_S$ of Social Integrity Constraints iff it is both negatively and positively compliant to $\mathcal{IC}_S$.

The protocols of Endriss et al. [27] correspond to *Deterministic Finite Automata* (DFAs). Clearly, it is possible to encode such protocols also by means of social integrity constraints. It turns out that the notions of weak conformance and negative compliance coincide for the class of protocols that can be represented using DFAs; and similarly, exhaustive conformance corresponds to strong compliance.

We have shown in D8 [36] how the $\mathcal{S}$CIFF proof procedure can be used to verify compliance by observation *on the fly* (case (2) in Figure 1). This result is summarised in Section 7.2.

Checking conformance *a priori* (case (1) in Figure 1) is generally a very difficult task, as it involves the (possibly very complex) specification of a computee and the specification of a protocol. In Section 7.3, we show how to check (weak) conformance *a priori* with respect to a simple class of protocols called *shallow* protocols. Our results pertaining to the study of protocol properties themselves (case (3) in Figure 1) have been presented earlier in Section 6.2 (because these properties are not strictly related to the topic of *conformance* but rather properties of the social infrastructure that provides the protocol). Section 7.4 discusses an aspect of *protocol competence*, which is, broadly speaking, a computee's ability to deal appropriately with a given interaction protocol beyond the basic requirement of syntactic conformance to its rules.

7.2 On-the-fly Conformance Checking

Property classification:	`[PC-1]` On-the-fly conformance checking
Partners involved:	UNIBO, DIFERRARA
Relevant papers:	Deliverables D5 [54] and D8 [36]
Status:	Proved

Introduction. On-the-fly conformance checking is an essential feature of the society model and has already been addressed in workpackages WP2 and WP3 throughout the first two years of the project. Given that on-the-fly conformance is also an important property that is of high relevance to the objectives of WP5, we include again a brief summary in this presentation.

Informal statement of the property. On-the-fly conformance checking means verifying that a history of events is conformant to a protocol, by observing the history during its evolution.

Significance. Verifying the conformance of agent interaction to a protocol is of high importance in Multiagent Systems. The SOCS social model has demonstrated that Computational Logic provides a viable approach to achieving this objective.

Formal statement of the property. The formal definitions may be found in deliverables D5 [54] and D8 [36].

Approach. On-the-fly conformance is verified by means of the $\mathcal{S}$CIFF [36] proof procedure. The web site `http://www.lia.deis.unibo.it/Research/Projects/SOCS/partners/societies/protocols.html` contains several examples of protocols and histories that have been checked on-the-fly for conformance.

Related work. Related work on conformance verification has been discussed extensively in deliverables D5 [54] and D8 [36].

7.3 Weak a-priori Conformance Checking for Shallow Protocols

Property classification:	`[PC-2]` A-priori conformance checking
Partners involved:	ICSTM
Relevant papers:	Endriss et al. [27, 29]
Status:	Proved

Introduction. A *shallow protocol* is a protocol where the range of legal follow-up moves at any given stage in a dialogue depends on the previous dialogue move alone. Shallow protocols can be represented using social integrity constraints [54]: the only (positive) expectations required will refer to the next point in time (or, more precisely, the next turn in the conversation regulated by the protocol) and depend only on events that have just happened.

Informal statement of the property. Recall from Section 7.1 that a computee is called *weakly conformant* to a protocol iff it will never utter any illegal dialogue moves (but it may indeed remain silent and thereby violate a social expectation) [27]. We have shown that a computee will be weakly conformant to a given shallow protocol whenever its so-called "response space" (a simple abstraction from the computee's reactivity knowledge base) logically entails (a suitable representation of) that protocol. This result applies to computees whose reactivity knowledge base has the structure of an abductive agent as defined in [68], and whose planning knowledge base does not include any communicative actions.

Significance. Shallow protocols constitute a simple, yet important, class of protocols, which can be used to represent most of the protocols based on finite state machines found in the multiagent systems literature (e.g. [60, 61]).

Weak conformance is certainly a first important requirement for conformance in general. From a technical point of view, it also has the advantage that it can be proved without reference to every single component of the KGP model, and it only relies on soundness (rather than completeness) results for the computational model of individual computees.

Formal statement of the property. The property is formally stated and proved in [27].

Approach. The result applies to computees whose reactivity knowledge base consists of a set of reactive rules of the form $P(\tau) \wedge C \Rightarrow P'(\tau{+}1)$,[5] specifying that whenever the computee receives a communicative action of the form P at time τ and condition C holds, then it should react by performing the communicative action P' at time $\tau+1$. We define the notion of a computee's *response space* as the set of reactive rules we get by first dropping all conditions C and then conjoining implications with identical antecedents by collecting the corresponding consequents into a single disjunction (an example would be $P(\tau) \Rightarrow P'_1(\tau{+}1) \vee P'_2(\tau{+}1) \vee \cdots \vee P'_n(\tau{+}1)$). A response space has the same structure as a shallow protocol (if we remove the operators **H** for happened events and **E** for expectations and consider only the basic structure of protocol constraints). A computee whose response space is a specialisation of the relevant protocol constraints can never utter an illegal dialogue move. Equivalently, a computee whose response space entails the (simplified representation of) the protocol is bound to be weakly conformant to that protocol.

Reactivity and Planning are the only transitions that can introduce new communicative actions.[6] Plan Revision may cause an action to be deleted, Action Execution may not execute an action in time, and a badly designed cycle may prevent the transitions generating new actions from being called in the first place, but none of these other components of the KGP model can be responsible for *generating* an illegal communicative action. Therefore, when proving results

[5] We use a simplified syntax to describe reactive rules here.

[6] While Sensing Introduction and Active Observation Introduction can introduce actions as well, these are not communicative actions, and thereby do not affect the computee's conformance to the protocol.

pertaining to *weak* conformance alone, we can restrict our attention to Reactivity and Planning. By assuming that the planning knowledge base does not involve any communicative actions, we have furthermore been able to reduce the problem to a verification problem with respect to the reactivity knowledge base alone.

Related work. We also have preliminary results on *exhaustive conformance* (which additionally requires a computee to utter a legal move whenever it is its turn) for shallow protocols [29], but these results only apply to a simplified model and rely on the availability of complete reasoning mechanisms.

7.4 Reachability for Shallow Protocols

Property classification:	`[PC-3]` Protocol competence
Partners involved:	ICSTM, CITY
Relevant papers:	Endriss et al. [22]
Status:	Preliminary results

Introduction. Arguably, the ability to merely *conform* to a protocol is not sufficient to call a computee a *competent* user of that protocol. For example, in the context of a negotiation protocol, both accepting and rejecting a proposal may constitute legal moves, but a computee rejecting any given proposal (whatever its content) could hardly be called a competent user of such a negotiation protocol, despite behaving in conformance to its rules. The broad term *protocol competence* covers all aspects of a computee's ability to use a given protocol appropriately beyond being able to conform to the rules of interaction of such a protocol. We have introduced this concept and provided a preliminary study of one aspect of competence with respect to shallow protocols (see also Section 7.3) in [22].

Informal statement of the property. We approach the intuitive concept of protocol competence by introducing a notion that considers the joint ability of a pair of computees to reach a particular state of an interaction in a society of computees where interaction is regulated by a protocol that can be represented as a finite state machine. We have provided preliminary results that allow us to automatically check this type of competence with respect to shallow protocols (a large subclass of the protocols representable by finite sate machines) and a simple class of computees.

Significance. Interaction protocols are central to any distributed systems where autonomous entities need to cooperate or negotiate. Being able to use these protocols in a competent manner is equally important for such applications. While the concept of protocol conformance has achieved much attention, both within the SOCS project and within the multiagent systems community in general, there appear to have been no attempts to address the wider issue of protocol competence. Our work in this area provides some first steps in this direction.

The significance of the particular class of shallow protocols considered here has already been argued for in Section 7.3.

We have also introduced a first application of our technique in [22], namely to automatically *customise* interaction protocols to better meet the needs of computees that are not fully competent to use the original protocol. In this approach, we first identify states in a protocol that, once reached, would prevent computees from ever reaching a terminal state (due to the

incompetence of one of the participants). A social authority would be able to use this information to propose a cut-down version of the original protocol that would prevent computees from entering into such deadends.

Formal statement of the property. The property is stated in detail in [22].

Approach. Similarly to the property discussed in Section 7.3, our results apply to computees with a simple reactivity knowledge base for which it is possible to extract the computee's *response space* that can be matched against a given shallow protocol. Furthermore, the result presupposes that no other components of the KGP model (in particular, planning) will interfere with the computee's communicative behaviour in an unexpected manner. Our results are preliminary, in particular, in the sense that the latter is a very strong assumption. We refer to [22] for further details.

Related work. We are not aware of related work that aims specifically at identifying aspects of protocol competence.

8 Conclusion

In this deliverable we have reported the work that was undertaken for workpackage WP5 of the SOCS project. WP5 concerned the verification of properties of the computee and the society models. We chose a broad range of properties, classified into four categories related to the proof procedures, individual computees, societies and protocols, with each category consisting of several results. For each property we have summarised its informal and formal definitions, its significance and the approach taken in its proof and we give references to annex documents that provide detailed discussions.

The summary of most of the properties includes a discussion of work related to that specific property. In the cases of behavioural profiles there is a separate section for the related work (Section 5.1.1), because this is shared by all the properties in this category. To our knowledge our work on all the properties, with the exception of the soundness, completeness and termination of the proof procedures, is novel.

8.1 Evaluation

The results reported in this deliverable go some way towards showing:

- The appropriateness of the models: For example, the coherence properties of Section 5.2 show some advantageous features that result from the design choices made in the computee model, and the soundness, completeness and termination properties of the society proof procedure show the appropriateness of the choices made there.

- The scope and versatility of the models: For example Section 5.1 on the behaviour profiles shows how different behaviours can be accommodated within the computee model. Another example is Section 6.2 which shows how the society proof procedure, itself, can be adapted to prove properties of protocols automatically.

- The suitability of the models for the chosen scenarios, in particular resource allocation: For example, the results reported in the whole of Section 7 on protocol conformance are immediately relevant to the communication requirements for negotiation. In addition, Section 6.3 discusses several concepts of optimality of allocations of resources that are relevant in this context.

- The use of theoretical results towards providing guidelines for the design of computees and the social infrastructure to anyone who wishes to use the PROSOCS platform: For example, the results on well-definedness reported in Section 6.1 help give guidelines to the designers of the social infrastructure, and the investigation of the behaviour profiles of Section 5.1 give guidelines to the designers of computees.

In addition the results show a possible synthesis between the computee and society models: For example, Section 5.4 shows how social expectations can be incorporated in the private policies of computees who negotiate over resources. Another example is the result on a-priori conformance checking reported in Section 7.3, which relates (parts of) the specification of a computee to that of a society protocol.

Work on WP5 has helped provide detailed feedback to us about the models of computees and societies. Work on the profiles (Section 5.1) suggested that it would be better to combine the Goal Revision and Plan Revision transitions into one. Consequently we have defined a new State Revision transition, SR, (see Section 2.3.1) to replace the two.

Work on the Coherence Properties (Section 5.2) revealed two problems with the Action Selection Function. One was an error in its definition that meant that when a set of actions *As* was selected for execution the time constraints of the actions in *As* collectively were not guaranteed to be consistent. This has been corrected as reported in Section 2.3.2. Another problem was that (both the original and the corrected versions of) the Action Selection Function did not guarantee that actions would be selected in the "right" order. So an action with a later time could be selected even if there were earlier unexecuted and non-timed out actions still present in the Plan. After many discussions we decided that it would be better to allow this in the core model and, instead, specify a behaviour profile, namely the *simple punctual*, that would force selection of actions in the right order.

It was valuable to have the work on WP6 on practical experimentation progressing in parallel. This too provided feedback about the models and the properties investigated in WP5.

Despite the successes outlined above, there have been inevitable difficulties. One of the major problems has been that the computee model is complex. It is defined declaratively at all levels and it is modular. So it lends itself well to proving properties about its subparts. However, with more complex, overall properties, one has to take account of the whole model. Not only that, but one also needs to take into account the environment and its possible interventions. These considerations made the work on individual welfare and the behaviour profiles particularly difficult. One consequence of this has been that, for example, the results about the effectiveness of individual behaviour profiles are based on conditions that need more detailed discussion and perhaps more formal characterisation. One example of such a condition is *time-criticality*. It is intuitively clear what we mean by *time-criticality*. However, it is very difficult to give sufficient and necessary conditions for it on the knowledge bases of computees. One possible way to tackle some of the issues and complexities in any related future work may be to specify a model of the environment as well as computees.

8.2 Planning and Division of Work

The work that has been done on WP5 has largely followed the task allocation planned in D12 [3]. Several plenary and subgroup meetings have taken place during the last year to finalise the list of properties to be addressed and to plan the work amongst the partners, and later to discuss progress and cross-fertilisation. This has been done both within WP5 and across WP5 and WP6.

There have been many close collaborations and synergies across the sites and on different topics. This is self-evident from the list of authors in the bibliography. For example the work on *behaviour profiles* has involved the collaboration of ICSTM, DIPISA, CITY and UCY, the work on $\mathcal{S}$CIFF *soundness* and *termination* results has involved the collaboration of UNIBO and DIFERRARA, and the work on *adopting social expectations* has involved a collaboration between UCY and UNIBO and also demonstrates a synergy between the individual computee and the society models and concepts.

8.3 Future Work

The SOCS project has been a complex and very ambitious project at all its stages, from the development of the declarative models, to the development of the computational models and prototype implementations and now at the stage of formalising and proving formal properties. We have tackled a diverse and fairly large collection of properties. Almost in all cases there is scope for refinement and improvement. Some of these we will attempt during the remainder of the project and most are subject of future work beyond the project.

Considering the four categories of the properties: Regarding the proof procedures, there is potential for a more comprehensive proof of completeness of CIFF and $\mathcal{S}$CIFF. Regarding individual computees, there is much scope for building on our existing work, for example to extend the results on the behaviour profiles, or to investigate the consequences, positive and otherwise, of combining different profiles. Another example is to extend the definitions and results of individual welfare to more complex situations, allowing time-critical goals and plans. The work on adopting social expectations is preliminary, but promising. Investigating further and tighter synergies between computees and societies provides scope for much interesting future work. Regarding the society infrastructure there are also possibilities for extending the work. For example extending the automatic verification of protocol properties to proof of universal properties. Another example is investigating policies and protocols that would result in socially optimal allocations of resources through negotiation. Regarding protocol conformance, extending the approach for proving conformance *a priori* discussed in Section 7.3 to both larger classes of protocols and richer agent models presents another important research challenge for the future.

References

[1] L. C. Aiello and F. Massacci. Planning attacks to security protocols: Case studies in logic programming. In A. C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, volume 2407 of *Lecture Notes in Computer Science*, pages 533–560. Springer-Verlag, 2002.

[2] M. Alberti, A. Bracciali, F. Chesani, A. Ciampolini, U. Endriss, M. Gavanelli, A. Guerri, A. Kakas, E. Lamma, W. Lu, P. Mancarella, P. Mello, M. Milano, F. Riguzzi, F. Sadri,

K. Stathis, G. Terreni, F. Toni, P. Torroni, and A. Yip. Experiments with animated societies of computees. Technical report, SOCS Consortium, 2005. Deliverable D14.

[3] M. Alberti, A. Bracciali, F. Chesani, U. Endriss, M. Gavanelli, A. Guerri, A. Kakas, E. Lamma, P. Mancarella, P. Mello, M. Milano, F. Riguzzi, F. Sadri, K. Stathis, G. Terreni, F. Toni, and P. Torroni. Update report on WP1–WP6. Technical report, SOCS Consortium, 2004. Deliverable D12.

[4] M. Alberti, A. Bracciali, F. Chesani, U. Endriss, M. Gavanelli, W. Lu, K. Stathis, and P. Torroni. SOCS prototype. Technical report, SOCS Consortium, 2003. Deliverable D9.

[5] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. On the automatic verification of interaction protocols using g-$\mathcal{S}$CIFF. Technical Report DEIS-LIA-04-004, University of Bologna (Italy), 2005. LIA Series no. 72.

[6] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Specification and verification of agent interactions using social integrity constraints. *Electronic Notes in Theoretical Computer Science*, 85(2), 2003.

[7] J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Computing environment-aware agent behaviours with logic program updates. In A. Pettorossi, editor, *Logic Based Program Synthesis and Transformation, 11th International Workshop, (LOPSTR'01), Selected Papers*, pages 216–232. Springer-Verlag, 2002.

[8] J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, volume 2424 of *Lecture Notes in Artificial Intelligence*, pages 50–61. Springer-Verlag, Sept. 2002.

[9] L. Amgoud and S. Parsons. Agent dialogues with conflicting preferences. In *Intelligent Agents VIII: 8th International Workshop, ATAL 2001, Seattle, WA, USA, Revised Papers*, volume 2333 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[10] L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue and negotiation. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence, Berlin, Germany (ECAI 2000)*. IOS Press, Aug. 2000.

[11] A. Armando, L. Compagna, and Y. Lierler. Automatic compilation of protocol insecurity problems into logic programming. In J. J. Alferes and J. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-2004)*, LNAI, pages 617–627. Springer-Verlag, 2004.

[12] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. North-Holland, 2002.

[13] F. Athienitou, A. Bracciali, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. Profile-related properties. Technical report, SOCS Consortium, 2005. Annex to deliverable D13.

[14] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In P. K. Pandya and J. Radhakrishnan, editors, *FST*

TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings, volume 2914 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 2003.

[15] A. Bracciali, P. Mancarella, K. Stathis, and F. Toni. On modelling declaratively multi-agent systems. In *Proceedings of Declarative Agent Languages and Technologies (DALT 2004)*, LNCS, To appear, 2004. Springer-Verlag.

[16] A. Bracciali, P. Mancarella, K. Stathis, and F. Toni. Stable multi-agent systems. In *Proc. of the 5th International Workshop on Engineering Societies in the Agents World (ESAW 2004)*, Lecture Notes in Artificial Intelligence, Toulouse, France, 2004. Springer-Verlag. To appear.

[17] A. Ciampolini, E. Lamma, P. Mello, F. Toni, and P. Torroni. Co-operation and competition in *ALIAS*: a logic framework for agents that negotiate. *Computational Logic in Multi-Agent Systems. Annals of Mathematics and Artificial Intelligence*, 37(1-2):65–91, 2003.

[18] G. Delzanno. Specifying and debugging security protocols via hereditary Harrop formulas and λProlog - a case-study -. In H. Kuchen and K. Ueda, editors, *Functional and Logic Programming, 5th International Symposium, FLOPS 2001, Tokyo, Japan, March 7-9, 2001, Proceedings*, volume 2024 of *Lecture Notes in Computer Science*, pages 123–137. Springer-Verlag, 2001.

[19] G. Delzanno and S. Etalle. Proof theory, transformations, and logic programming for debugging security protocols. In A. Pettorossi, editor, *Logic Based Program Synthesis and Transformation : 11th International Workshop, (LOPSTR 2001). Selected papers.*, volume 2372 of *Lecture Notes in Computer Science*, pages 76–90, Paphos, Cyprus, November 2001. Springer Verlag.

[20] N. Demetriou and A. C. Kakas. Argumentation with abduction. In *Proceedings of the fourth Panhellenic Symposium on Logic*, 2003.

[21] C. Dixon, M.-C. Fernández Gago, M. Fisher, and W. van der Hoek. Using temporal logics of knowledge in the formal verification of security protocols. In *Procedings of the Eleventh International Workshop on Temporal Representation and Reasoning (TIME'04)*, 2004.

[22] U. Endriss, W. Lu, N. Maudet, and K. Stathis. Competent agents and customising protocols. In A. Omicini, P. Petta, and J. Pitt, editors, *Proceedings of the 4th International Workshop on Engineering Societies in the Agents World (ESAW-2003)*, October 2003.

[23] U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The CIFF proof procedure: Definition and soundness results. Technical Report 2004/2, Department of Computing, Imperial College London, May 2004.

[24] U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The CIFF proof procedure for abductive logic programming with constraints. In J. J. Alferes and J. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-2004)*, LNAI, pages 31–43. Springer-Verlag, 2004.

[25] U. Endriss and N. Maudet. Welfare engineering in multiagent systems. In A. Omicini, P. Petta, and J. Pitt, editors, *Engineering Societies in the Agents World IV*, volume 3071 of *LNAI*, pages 93–106. Springer-Verlag, 2004.

[26] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations over resources. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 177–184, Melbourne, Victoria, July 14–18 2003. ACM Press.

[27] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol conformance for logic-based agents. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico (IJCAI-03)*. Morgan Kaufmann Publishers, Aug. 2003.

[28] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Resource Allocation in Egalitarian Agent Societies. In A. Herzig, B. Chaib-draa, and P. Mathieu, editors, *Secondes Journées Francophones sur les Modèles Formels d'Interaction (MFI-2003)*, pages 101–110. Cépaduès-Éditions, May 2003.

[29] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Logic-based agent communication protocols. In F. Dignum, editor, *Advances in Agent Communication*, volume 2922 of *LNAI*, pages 91–107. Springer-Verlag, 2004.

[30] U. Endriss, N. Maudet, F. Sadri, F. Toni, and P. Torroni. Preliminary list of verifiable properties of societies of computees. Technical report, SOCS Consortium, June 2003. Internal document.

[31] FIPA Communicative Act Library Specification, Aug. 2001. Published on August 10th, 2001, available for download from the FIPA website, `http://www.fipa.org`.

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

[33] 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.

[34] M. Gavanelli, E. Lamma, and P. Mello. Proof of properties of the $\mathcal{S}$CIFF proof-procedure. Deliverable IST32530/DIFERRARA/410/D/I/b1, SOCS Consortium, Dec 2004.

[35] 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.

[36] 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.

[37] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–207, 1989.

[38] F. Guerin and J. Pitt. Guaranteeing Properties for E-commerce Systems. In J. Padget et al., editors, *Agent-Mediated Electronic Commerce IV: Designing Mechanisms and Systems*, volume 2531 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer-Verlag, 2002.

[39] J. C. Harsanyi. Can the maximin principle serve as a basis for morality? *American Political Science Review*, 69:594–609, 1975.

[40] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J. C. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

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

[42] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. A logic-based approach to model computees. Technical report, SOCS Consortium, 2003. Deliverable D4.

[43] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. In R. Lopez de Mantaras and L. Saitta, editors, *Proceedings of the Sixteenth European Conference on Artificial Intelligence, Valencia, Spain (ECAI 2004)*. IOS Press, Aug. 2004.

[44] A. Kakas, N. Maudet, and P. Moraitis. Layered strategies and protocols for argumentation-based agent interaction. In *Proceedings of the First International Workshop on Argumentation in MultiAgent Systems*, 2004.

[45] A. C. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 883–890, Melbourne, Victoria, July 14–18 2003. ACM Press.

[46] A. C. Kakas, P. Torroni, and N. Demetriou. Agent planning, negotiation, and control of operation. In R. Lopez de Mantaras and L. Saitta, editors, *Proceedings of the Sixteenth European Conference on Artificial Intelligence, Valencia, Spain (ECAI 2004)*, pages 28–32. IOS Press, Aug. 2004.

[47] G. E. Kersten, S. J. Noronha, and J. Teich. Are all e-commerce negotiations auctions? In *Proceedings of the 4th International Conference on the Design of Cooperative Systems*, 2000.

[48] K. Kim, J. A. Abraham, and J. Bhadra. Model checking of security protocols with pre-configuration. In K. Chae and M. Yung, editors, *Information Security Applications, 4th International Workshop, WISA 2003, Jeju Island, Korea, August 25-27, 2003, Revised Papers*, volume 2908 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2004.

[49] P. Küngas and M. Matskin. Linear logic, partial deduction and cooperative problem solving. In J. A. Leite, A. Omicini, L. Sterling, and P. Torroni, editors, *Declarative Agent Languages and Technologies*, volume 2990 of *Lecture Notes in Artificial Intelligence*, pages 263–279. Springer-Verlag, May 2004. First International Workshop, DALT 2003. Melbourne, Australia, July 2003. Revised Selected and Invited Papers.

[50] E. Lamma, P. Mello, P. Mancarella, A. Kakas, K. Stathis, and F. Toni. Self-assessment: parameters and criteria. Technical report, SOCS Consortium, 2003. Deliverable D3.

[51] G. Lowe. Breaking and fixing the Needham-Shroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction*

and Analysis of Systems: Second International Workshop, TACAS'96, volume 1055 of *Lecture Notes in Artificial Intelligence*, pages 147–166. Springer-Verlag, 1996.

[52] P. Mancarella, F. Sadri, K. Stathis, F. Toni, and A. Bracciali. Computees and welfare. Technical report, SOCS Consortium, 2005. Annex to deliverable D13.

[53] P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part I*, pages 402–409, Bologna, Italy, July 15–19 2002. ACM Press.

[54] 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.

[55] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[56] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[57] I. Niemelä and P. Simons. Extending the Smodels system with cardinality and weight constraints. In J. Minker, editor, *Logic-Based Artificial Intelligence*, chapter 21, pages 491–521. Kluwer Academic Publishers, 2000.

[58] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[59] J. Pang. Analysis of a security protocol in μCRL. In C. George and H. Miao, editors, *Formal Methods and Software Engineering, 4th International Conference on Formal Engineering Methods, ICFEM 2002 Shanghai, China, October 21-25, 2002, Proceedings*, volume 2495 of *Lecture Notes in Computer Science*, pages 396–400. Springer-Verlag, 2002.

[60] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[61] J. Pitt and A. Mamdani. Communication protocols in multi-agent systems. In *Workshop on Specifying and Implementing Conversation Policies*, 1999.

[62] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, pages 630–637, Columbia University, New York City, July 2004.

[63] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers, Apr. 1991.

[64] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, MA, 1994.

[65] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[66] F. Sadri and F. Toni. Coherence properties of the KGP model. Technical report, SOCS Consortium, 2005. Annex to deliverable D13.

[67] F. Sadri, F. Toni, and P. Torroni. Logic agents, dialogues and negotiation: an abductive approach. In *Proceedings AISB'01 Convention, York, UK*, Mar. 2001.

[68] F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: agent varieties and dialogue sequences. In *Intelligent Agents VIII: 8th International Workshop, ATAL 2001, Seattle, WA, USA, Revised Papers*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 405–421. Springer-Verlag, 2002.

[69] F. Sadri, F. Toni, and P. Torroni. Minimally intrusive negotiating agents for resource sharing. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico (IJCAI-03)*. Morgan Kaufmann Publishers, Aug. 2003.

[70] T. W. Sandholm. Contract types for satisficing task allocation: I Theoretical results. In *Proceedings of the AAAI Spring Symposium: Satisficing Models*, 1998.

[71] T. W. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press, Cambridge, MA, 1999.

[72] M. Scheutz, A. Sloman, and B. Logan. Emotional states and realistic agent behaviour. In *Proceedings of Game On 2000*. SCS Publishing, 2000.

[73] A. K. Sen. *Collective Choice and Social Welfare*. Holden Day, 1970.

[74] P. Torroni. A study on the termination of negotiation dialogues. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part III*, pages 1223–1230, Bologna, Italy, July 15–19 2002. ACM Press.

[75] M. Viroli, G. Moro, and A. Omicini. On Observation as a coordination paradigm: an ontology and a formal framework. In *ACM Symposium on Applied Computing - Proceedings of the 16th International Conference (SAC01)*, pages 166–175. ACM, March 2001, Las Vegas (NV), USA.

[76] D. von Oheimb and V. Lotz. Formal security analysis with interacting state machines. In D. Gollmann, G. Karjoth, and M. Waidner, editors, *Computer Security - ESORICS 2002, 7th European Symposium on Research in Computer Security, Zurich, Switzerland, October 14-16, 2002, Proceedings*, volume 2502 of *Lecture Notes in Computer Science*, pages 212–229. Springer-Verlag, 2002.

[77] M. Wooldridge. *Introduction to Multi-Agent Systems*. John Wiley & Sons, Ltd., 2002.

[78] M. Wooldridge and A. Lomuscio. A logic of visibility, perception, and knowledge: completeness and correspondence results. *Journal of the IGPL*, 9(2), March 2001.

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

List of Documents in the Annex

The following documents belong to the annex of this deliverable:

[13] F. Athienitou, A. Bracciali, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. *Profile-related properties*. SOCS Consortium, 2005.

[34] M. Gavanelli, E. Lamma, and P. Mello. *Proof of properties of the $\mathcal{S}$CIFF proof-procedure*. SOCS Consortium, 2005.

[52] P. Mancarella, F. Sadri, K. Stathis, F. Toni, and A. Bracciali. *Computees and welfare*. SOCS Consortium, 2005.

[66] F. Sadri and F. Toni. *Coherence properties of the KGP model*. SOCS Consortium, 2005.

These documents describe certain aspects of our work within WP5 in more detail. Many results have already been published or are available in publicly accessible technical reports from one of the partner institutions. In some cases we also refer to previous deliverables [54, 36]. The most relevant papers are cited in the headers of the property templates in Sections 4–7. We have also made these papers available at the following address:

`http://lia.deis.unibo.it/research/socs/guests/publications/D13.html`