Università degli Studi di Ferrara

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Elettronica

Applicazione di una Architettura ad Agenti alle Aste Elettroniche

Tesi di Laurea in Intelligenza Artificiale

Tesi di:
TARIN GAMBERINI

Relatore:
Chiar.ma Prof.sa Ing.
EVELINA LAMMA

Correlatori:
Dott. Ing. MARCO ALBERTI
Dott. Ing. MARCO GAVANELLI

Anno Accademico 2003/2004

taringamberini [at] taringamberini [dot] com
www.taringamberini.com

Alla mia famiglia, ...

Indice

Elenco delle figure

Introduzione

Negli ultimi anni il mercato dell'Information Technology ha visto la rapida evoluzione di settori, come quelli del commercio elettronico e delle aste elettroniche, che hanno dato impulso a nuove ricerche. In questo contesto gli studi sui sistemi multi–agente (MAS[1]) si sono moltiplicati e diversificati, allettati dalle prospettive di applicazioni offerte da un mercato in continua crescita.

Il punto di forza dei MAS è rappresentato dalla capacità di interazione fra gli agenti per il conseguimento di obiettivi. Ogni singola entità[2] può cooperare o competere con altri agenti al fine di raggiungere i propri scopi, oppure può associarsi in strutture organizzate in cui collaborazione e divisione dei compiti permettono di raggiungere goal condivisi con la società.

Il progetto dell'interazione fra agenti diviene una fase cruciale nello sviluppo dei MAS e gli sforzi compiuti in tal senso sono confluiti nella definizione di linguaggi fra agenti e nella definizione di protocolli di comunicazione.

In tale ambito si colloca il progetto europeo SOCS[3] [1] con lo scopo di investigare modelli logici e computazionali, di comportamento individuale e collettivo, di entità – computees – che interagiscono nel contesto di ambienti computazionali globali ed aperti.

Attualmente le tecniche per lo sviluppo dell'interazione in tali ambienti hanno dato esiti sia in implementazioni a basso livello prive di una evidente caratterizzazione logica, sia in specifiche astratte espressive ma in molti casi

[1] Multi–Agent Systems.

[2] Nel seguito adotteremo con medesimo significato i termini: agenti, entità, computees.

[3] Societies Of ComputeeS.

intrattabili. Per colmare questa distanza SOCS mira a fornire un modello per interazioni complesse che non solo permette la specifica e la verifica formale di proprietà, ma che permette realizzazioni concrete la cui correttezza può essere provata "on–the–fly".

In dettaglio gli obiettivi del progetto SOCS possono riassumersi nei seguenti punti:

1. Specificare la conoscenza ed il comportamento dei singoli agenti, astraendo dalla loro struttura, configurazione e progetto interno. Tale specifica include sia l'interfaccia fra le entità sia l'ambiente in cui esse operano.

2. Specificare le interazioni fra gli agenti in modo tale da far emergere una società.

3. Fornire un framework computazionale in cui ottenere i comportamenti attesi degli agenti.

4. Identificare e specificare proprietà desiderabili che dovrebbero essere soddisfatte tanto dai singoli agenti quanto dall'intera società.

5. Verificare quando gli agenti e la società soddisfino o violino tali proprietà.

6. Valutare e validare il framework attraverso una serie di esperimenti mirati.

Scopo di questa tesi è l'approfondimento degli argomenti accennati nei punti 2, 4 e 5 relativamente alle società di agenti costituenti le aste elettroniche. In particolare ci proporremo di definire le specifiche di interazione che caratterizzano tre tipologie di società, operanti rispettivamente negli scenari d'asta Inglese, d'asta Combinatoria e d'asta First Price Sealed Bid. Inoltre verificheremo quando l'agire degli agenti sia conforme, o meno, a tali specifiche. Infine verificheremo se tali specifiche godano, o non godano, di alcune proprietà attraverso una serie di esperimenti mirati per ogni tipologia d'asta.

Nel primo capitolo, dopo aver dato una definizione di agente e di società, descriveremo come gli agenti interagiscano fra loro attraverso *Agent Comunication Languages* ed *Interaction Protocols*, originando strutture sociali di vario tipo e di differente complessità. Accenneremo ai vari modelli implementativi proposti per l'analisi di tali società, soffermandoci in particolare sul modello definito dal progetto SOCS, basato su un *approccio sociale* agli Interaction Protocols, che adotteremo come paradigma di progetto e verifica nel seguito di questa tesi.

Nel secondo capitolo esporremo come il modello sociale SOCS descriva la conoscenza della società in modo dichiarativo. Introdurremo la Programmazione Logica Abduttiva[4] come linguaggio per la definizione di una istanza di società. Sulla base di quest'ultima definiremo una semantica che, partendo dalla *conoscenza sociale* e basandosi su *eventi* sociali ed *aspettative*, deriverà i goal prefissati dalla società. Infine introdurremo $\mathcal{S}$CIFF, una *proof procedure* che implementa il ragionamento abduttivo, che supporta CLP[5] ed è in grado di gestire l'arrivo dinamico di eventi.

Nel terzo capitolo vedremo come l'approccio sociale permetta di progettare protocolli di interazione fra agenti attraverso *vincoli di integrità sociale*. Descriveremo come impiegare $\mathcal{S}$CIFF per verificare la conformità di un insieme di eventi sociali ad un dato protocollo. Inoltre introdurremo g$\mathcal{S}$CIFF, una variante di $\mathcal{S}$CIFF, in grado di generare eventi conformi ad un protocollo. Infine, descriveremo come impiegare g$\mathcal{S}$CIFF per verificare se un dato protocollo goda, o meno, di alcune proprietà.

Nei capitoli quarto, quinto e sesto studieremo tre società di agenti operanti rispettivamente negli scenari d'asta Inglese, d'asta Combinatoria e d'asta First Price Sealed Bid. Per ogni asta svilupperemo un protocollo di interazione, motivando le scelte progettuali adottate, descrivendolo attraverso AUML e vincoli di integrità sociale. Con esempi pratici vedremo sia come $\mathcal{S}$CIFF verifichi la conformità di una history al protocollo, sia come g$\mathcal{S}$CIFF

[4] Abductive Logic Programming [16].
[5] Constraint Logic Programming [15].

verifichi se un protocollo goda, o meno, di determinate proprietà.

Nelle appendici riporteremo alcuni risultati, in versione integrale, come calcolati da $\mathcal{S}$CIFF e g$\mathcal{S}$CIFF che data la loro lunghezza non è stato conveniente inserire all'interno dei relativi paragrafi. Nei vari paragrafi, infatti, riassumeremo tali risultati evidenziandone gli aspetti notevoli.

Capitolo 1

Società di agenti

In questo capitolo, dopo aver dato una definizione di agente e di società, descriveremo come gli agenti interagiscano fra loro attraverso *Agent Comunication Languages* ed *Interaction Protocols*, originando strutture sociali di vario tipo e di differente complessità. Accenneremo ai vari modelli implementativi proposti per l'analisi di tali società, soffermandoci in particolare sul modello definito dal progetto SOCS [1], basato su un *approccio sociale* agli Interaction Protocols, che adotteremo come paradigma di progetto e verifica nel seguito di questa tesi.

1.1 Definizione di agente e di società

Un *agente* è una astrazione per una entità computazionale autonoma, nel senso che la sua attività non è controllata dall'esterno. Infatti ogni agente ha delle proprie conoscenze, capacità, risorse, obiettivi, regole e comportamenti. Inoltre ognuno di essi ha tipicamente una visione dell'ambiente e degli altri agenti incompleta, parziale e non accurata, pertanto potrebbe avere capacità inadeguate al conseguimento del proprio obiettivo.

Più formalmente [27] una entità può essere definita un agente se possiede le seguenti caratteristiche:

Autonomia è la capacità di operare senza alcun intervento esterno, mante-

nendo contemporaneamente un certo controllo sul proprio stato interno e sulle proprie azioni.

Abilità Sociale è la capacità di relazionarsi agli altri agenti interagendo con essi attraverso una qualche forma di linguaggio.

Reattività è la proprietà di reagire ai cambiamenti dell'ambiente esterno acquisiti attraverso opportune forme di percezione.

Inoltre tale entità potrebbe eventualmente possedere ulteriori caratteristiche come:

Pro–Attività è la reattività arricchita dalla capacità di adottare iniziative utili al conseguimento dell proprio obiettivo in modo più efficace.

Mobilità è l'abilità di passare attraverso vari contesti.

Veridicità è l'assunzione secondo cui un agente non comunicherà consapevolmente informazioni false.

Benevolenza è l'ipotesi dell'esistenza di obiettivi perseguibili non conflittuali.

Razionalità è l'assunzione secondo cui un agente perseguirà sempre e solo i propri obiettivi.

Tuttavia tali caratteristiche, pur specificando molti aspetti, rimangono una descrizione ad alto livello dell'agente, e non entrano nel merito di alcun dettaglio architetturale o implementativo dell'entità. Per questo un modello societario che permetta agli agenti di operare ed interagire deve necessariamente essere in grado di gestire entità *non completamente specificate*.

Il modello formale che soddisfa a tale ipotesi è conosciuto come *Open Societies Model*. Hewitt [12] ha dato una definizione, oramai ampiamente accettata, di società aperta costituita dai tre seguenti punti:

- Il comportamento dei membri e le loro interazioni non sono predicibili, perciò l'evoluzione della società non è deterministica.

- L'architettura interna di ogni membro non è né osservabile né nota pubblicamente, quindi i vari membri potrebbero avere architetture eterogenee.

- I membri della società non condividono gli stessi obiettivi, desideri o intenzioni, pertanto ogni membro potrebbe entrare in conflitto con altri mentre è impegnato a perseguire i propri scopi.

Una definizione di società aperta complementare alla precedente, è quella proposta da Davidsson [7]: in una società (artificiale) aperta non esistono restrizioni per gli agenti che desiderano unirsi ad essa o abbandonarla. Per entrare in una società è sufficiente interagire con un membro che già vi appartiene, per lasciarla è sufficiente interrompere ogni tipo di interazione. In particolare quest'ultimo aspetto non permette di asserire con certezza quando e se un agente abbia abbandonato la società. Tale incompletezza nella conoscenza societaria relativamente ai membri che la costituiscono, se da un lato comporta problematiche che dovranno essere opportunamente gestite, dall'altro ha il vantaggio di rendere non strettamente necessario il progetto di infrastrutture per il monitoraggio degli agenti.

1.2 Agent Comunication Languages e Interaction Protocols

Abbiamo osservato come un agente possegga una *abilità sociale* intesa come la capacità di relazionarsi alle altre entità interagendo tramite una qualche forma di linguaggio. Appare quindi evidente l'importanza di definire tale linguaggio in modo da permettere agli agenti di comunicare fra loro all'interno della società.

Agent Comunication Languages (ACL) e *Interaction Protocols* (IP) sono gli approcci tradizionali al supporto delle interazioni fra agenti.

La semantica degli atti comunicativi in ACL è definita solitamente in termini di *attitudini mentali* come convinzioni, desideri ed intenzioni. Tale

approccio è stato criticato come inadeguato all'interno delle società aperte [22], poiché senza vincoli prestabiliti sull'architettura interna degli altri agenti è impossibile verificare se le loro attitudini mentali siano esprimibili nel rispetto della semantica ACL.

Invece gli IP sono *strutture statiche* e definiscono sequenze enunciative che combinate rendono possibili comunicazioni coerenti. Tale approccio è stato criticato per la sua mancanza di flessibilità, soprattutto per la mancanza di regole che governino il modo secondo cui i protocolli possano essere estesi e combinati.

Fortunatamente gli ACL e gli IP sono stati definiti indipendentemente gli uni dagli altri, perciò un opportuno lavoro di sintesi ha permesso di sopperire alle mancanze dell'uno con gli aspetti, e nei casi migliori con i pregi, dell'altro.

1.3 Specifica di interazione fra agenti

Fino a poco tempo fa la ricerca nell'ambito degli Agent Comunication Languages (ACL) è stata rivolta principalmente alla generazione ed interpretazione di messaggi individuali, ed è rimasta fondamentalmente isolata da quella sugli Interaction Protocols (IP). Invero si assumeva, più o meno esplicitamente, che strutture di conversazione dovessero emergere come una conseguenza delle semantiche dei messaggi individuali. Ciò è espresso dalle specifiche della Foundation for Intelligent Physical Agents (FIPA) [9] in cui si afferma che:

> ...il processo di pianificazione degli agenti causa il sorgere spontaneo di un *Interaction Protocol* (IP) come conseguenza delle scelte compiute dagli agenti. Tuttavia ciò comporta un pesante carico di complessità nell'implementazione delle entità. Un'approccio alternativo, molto pragmatico, è quello di specificare a priori gli IP, così un'implementazione degli agenti più semplice può nondimeno tradursi in conversazioni con altri agenti piene

di significato, semplicemente seguendo attentamente le direttive espresse negli IP ...

Oggigiorno la situazione è mutata e gli IP sono considerati strutture teoriche di fondamentale importanza nella modellazione dell'interazione fra agenti.

Andiamo ora ad introdurre brevemente alcuni fra i più significativi formalismi proposti in letteratura per regolare l'interazione e permettere la generazione di conversazioni.

Input–Output Pairs Le coppie di input–output sono il modo più semplice per rappresentare modelli di interazione: la coppia specifica solamente l'appropriata risposta (output) ad un messaggio ricevuto (input). È chiaro che questo modello non permette di riferirsi alla storia del dialogo, perciò l'Input–Output Pairs è adatto per interazioni elementari.

Finite State Machines Gli automi a stati finiti sono il più adeguato, e popolare, formalismo per rappresentare le interazioni *sequenziali*. Lo stato dell'automa definisce lo stato della conversazione.

Dooley Graphs I grafi di Dooley sono stati introdotti nella comunità dei Multi–Agents Systems da Parunak [20], come un modo naturale di rappresentare il "dialogo come accade". Un grafo di Dooley sostanzialmente consiste in un insieme di nodi (i partecipanti), un insieme di indici (gli atti), un insieme di archi che connettono i nodi, ed un insieme di *vertici che connettono gli archi* i quali rappresentano varie forme si relazione fra gli atti.

AUML Protocol Diagrams I diagrammi di protocollo estendono il classico formalismo UML dedicato agli agenti [5] introducendo alcune nuove caratteristiche fra le quali: messaggi concorrenti e cardinalità nei messaggi. Tipicamente rappresentano *linee temporali* di vita dell'agente dalle quali partono, ed alle quali arrivano, gli atti comunicativi. AUML è ancora in fase di sviluppo e rimane un pseudo–formalismo di specifica.

Event Calculus Yolum e Singh [28] propongono una variante del calcolo ad eventi applicato alla specifica di un protocollo basato su *commitments*. La semantica dei messaggi e basata su *operazioni* a loro volta definite da *predicati* su *eventi*. Inoltre i *commitments possono evolvere* in relazione agli eventi come prescritto da un insieme di *postulati*. Un tale approccio rende il protocollo più flessibile rispetto ad un approccio tradizionale in cui si specificano sequenze di azioni. L'unica condizione che deve essere rispettata è che al termine dell'esecuzione del protocollo non deve essere rimasto alcun commitment pendente.

1.4 Modellazione delle società di agenti

Nell'ambito dei *Multi–Agent Systems* sono stati tentati diversi approcci alla *modellazione delle società*, all'interno delle quali gli agenti, cooperando o competendo, possono conseguire i loro obiettivi.

Distribuited Problem Solving La necessità di un modello societario nell'ambito del DPS sorge quando la soluzione di un problema può essere raggiunta coordinando gruppi di entità computazionali. Occorre definire opportunamente il modello societario, i sistemi e le tecnologie che permettano la formazione dinamica di coalizioni, competizioni e cooperazioni, ossia di tutte quelle interazioni che concorrano alla soluzione del problema.

Due tipici esempi di questo tipo di società sono quelli basati sui Market Models e sulle Contract Net Models.

Market Model Il Market Model (MM) [25, 24] si ispira all'idea di mercato, in cui gli agenti comprano e vendono merci e servizi. La società è perciò definita come un mercato, in cui l'equilibrio raggiunto dopo le transazioni economiche è il risultato degli scambi effettuati dalla società. Tuttavia il MM è incentrato particolarmente sull'aspetto computazionale, pertanto rappresenta un modello societario più nelle intenzioni che nella realizzazione pratica.

Contract Net Model Il Contract Net Model (CNM) [14, 23] è strettamente correlato al MM e specifica l'organizzazione dinamica attraverso negoziazioni in uno scenario d'asta. Il contratto è un insieme di compiti che devono essere svolti. La negoziazione mira ad assegnare il contratto ad agenti in grado di portare a termine i compiti in esso richiesti. Una negoziazione è strutturata attraverso tre punti:

1. Annuncio: ogni agente annuncia i compiti che non sa svolgere.
2. Offerta: gli agenti rispondono agli annunci e piazzano offerte relativamente ai compiti che sanno portare a termine.
3. Concessione: le offerte ricevute sono valutate e all'agente vincitore è concesso il contratto.

Il CNM presenta un modello di interazione fra entità più strutturato rispetto al MM, tuttavia presenta lacune nella definizione della società, sostanzialmente poiché si è concentrato lo sforzo maggiore nell'implementare l'articolato meccanismo di distribuzione dei compiti sopra descritto.

BDI Architectures Una pietra miliare nell'ambito dei Multi–Agent Systems è l'architettura *Belif–Desire–Intention* (BDI) [21]. Essa si basa sull'assunzione che gli agenti abbiano una rappresentazione interna del mondo e dello *stato mentale* degli altri agenti. Ognuno di essi basandosi su tale stato mentale è in grado di effettuare *ragionamenti* sugli altri agenti.

I modelli BDI soffrono di un grosso problema difficilmente superabile in scenari che coinvolgono entità computazionali autonome ed eterogenee: come già osservato da Singh [22] è impossibile ricostruire lo stato mentale degli altri agenti poiché, data la loro eterogeneità, non è ammissibile nessuna ipotesi sulla loro architettura interna.

Organizational Models Lo schema organizzativo definito da Dignum et

al. [8] è basato su tre modelli fra loro correlati: *organizzativo*, *sociale* e di *interazione*.

- Il *modello organizzativo* definisce la coordinazione e gli elementi normativi che descrivono il comportamento atteso dalla società. Le sue componenti sono regole, vincoli e strutture comunicative.
- Il *modello sociale* specifica i contratti che rendono espliciti i vincoli di delega tra agenti.
- Il *modello di interazione* descrive le possibili interazioni fra agenti specificando i contratti in termini di accordi, condizioni e sanzioni.

1.5 Approccio sociale

Abbiamo osservato come nell'ambito dei *Multi–Agent Systems* siano necessari l'esistenza di Agent Comunication Languages e Interaction Protocols per permettere l'interazione fra agenti. Tuttavia ACL e IP non sono sufficienti a rendere la comunicazione realmente efficace. Questo problema è da ricercare nel fatto che essi non garantiscono univocità di interpretazione di una comunicazione da parte dei diversi agenti. In altre parole gli agenti comprendono i messaggi che si scambiano, ma ad essi associano un significato che dipende dal proprio *atteggiamento mentale*. D'altro canto in una società aperta, caratterizzata da agenti eterogenei ed internamente non osservabili, è impossibile accedere allo stato mentale delle singole entità per garantire uniformità di significato ai dialoghi scambiati.

Il progetto SOCS [1] ha tentato di conciliare tali aspetti contraddittori proponendo una soluzione caratterizzata da un **approccio sociale** agli IP. Esso consiste nel definire gli IP attraverso **vincoli di integrità sociale su eventi sociali**. Con queste parole intendiamo la possibilità di esprimere sia le regole che devono essere rispettate dagli agenti per poter interagire, sia gli atteggiamenti **attesi** dalla società che sono ritenuti coerenti.

Una delle intuizioni alla base del progetto SOCS è stata quella di garan-

tire l'uniformità di interpretazione dei dialoghi da parte degli agenti facendoli interagire attraverso la società. In risposta agli atti comunicativi degli agenti la **società esige** determinati comportamenti poiché tali atti sono interpretati socialmente con un solo e ben determinato significato: un **unico significato sociale**. Non sono più i singoli agenti che dialogano direttamente fra di loro attraverso un IP privato ma essi dialogano con la società; essi dialogano indirettamente fra di loro utilizzando l'IP pubblico definito dalla società stessa.

Ciò non vieta agli agenti di continuare ad utilizzare propri IP, ma per interagire è necessario rispettare anche quelli societari. Nel caso in cui gli agenti dimostrino atteggiamenti non conformi a quelli attesi dalla società, è possibile:

- individuare atteggiamenti socialmente scorretti;
- rieducare gli agenti ad un corretto comportamento sociale e punire i recidivi;
- verificare il fallimento di atti comunicativi riportando la comunicazione ad uno stato consistente.

Questo approccio richiede la presenza di una *entità autonoma*[1] che osserva le interazioni fra agenti e ne giudica la conformità ai protocolli di interazione.

[1] Vedi Capitolo 3.

Capitolo 2

Abductive Logic Programming e $\mathcal{S}$CIFF

In questo capitolo esporremo come il modello sociale SOCS descriva la conoscenza della società in modo dichiarativo. Introdurremo la Programmazione Logica Abduttiva[1] come linguaggio per la definizione di una istanza di società. Sulla base di quest'ultima definiremo una semantica che, partendo dalla *conoscenza sociale* e basandosi su *eventi* sociali ed *aspettative*, deriverà i goal prefissati dalla società. Infine introdurremo $\mathcal{S}$CIFF, una proof che implementa il ragionamento abduttivo, che supporta CLP[2] ed è in grado di gestire l'arrivo dinamico di eventi.

2.1 Il modello sociale SOCS

Il modello sociale SOCS [4] descrive la conoscenza della società in modo dichiarativo. Tale conoscenza è costituita principalmente di due parti: una parte *statica* che definisce gli elementi organizzativi e normativi della società, ed una parte *dinamica* che descrive gli eventi accaduti considerati rilevanti per la società. Gli eventi che prenderemo in considerazione sono *atti comu-*

[1] Abductive Logic Programming [16].
[2] Constraint Logic Programming [15].

nicativi, in accordo con la maggior parte degli studi compiuti sugli agenti software. In aggiunta a queste due categorie di conoscenza sono inoltre conservate informazioni relativamente agli obiettivi sociali: i goal che la società si prefigge di raggiungere.

Gli elementi normativi sono codificati nei vincoli di integrità sociale che indicheremo con $\mathcal{IC}_S$[3]. La società è consapevole istante per istante degli eventi che accadono dinamicamente nell'ambiente ed il cui succedersi forma la sua *storia* passata. La società basandosi sulla storia, sugli $\mathcal{IC}_S$ e sui goal può definire degli *eventi sociali attesi*, in particolare può definire quali di essi si aspetta che accadano o *non* accadano. Tali eventi rappresentano il comportamento *ideale* degli agenti ed in seguito ci riferiremo ad essi con l'espressione *aspettative sociali*.

2.2 Rappresentazione della conoscenza sociale

La conoscenza di una società S è data dalle seguenti parti:

- La conoscenza organizzativa di base della società $SOKB$[4], che è una conoscenza statica.
- I vincoli di integrità sociale $\mathcal{IC}_S$, che sono normative statiche.
- I goal $\mathcal{G}$ che la società si è prefissata.

Con il continuo accadere di eventi una società può evolvere dando origine a sequenze di istanze di società, ognuna caratterizzata dalle tre parti soprascritte alle quali se ne aggiunge una dinamica che indicheremo con $SEKB$[5]. In dettaglio $SEKB$ è composta da:

- Gli *eventi attesi* indicati da atomi[6] con funtore **H**[7].

[3]Social Integrity Constraints.
[4]Social Organization Knowledge Base.
[5]Social Environment Knowledge Base.
[6]Nel seguito adotteremo l'usuale terminologia della Programmazione Logica [19].
[7]**H**appened.

- Le *aspettative* intese come eventi che dovrebbero (ma potrebbero non) accadere, indicate da atomi con funtore **E**[8], ed eventi che dovrebbero non (ma potrebbero) accadere, indicati da atomi con funtore **EN**[9].

Gli eventi accaduti non sono tutti quelli realmente avvenuti, ma solo quelli osservabili esternamente dagli agenti e *rilevanti* per la società. L'insieme di tali eventi costituisce la storia **HAP** di una istanza di società.

Gli eventi sono rappresentati come atomi ground:

$$\mathbf{H}(Event\ [, Time]) \tag{2.2.1}$$

Le aspettative possono essere sia positive **E**, che negative **EN**:

$$\mathbf{E}(Event\ [, Time]) \qquad \mathbf{EN}(Event\ [, Time]) \tag{2.2.2}$$

E indica un evento che la società si aspetta che accada, **EN** indica un evento che la società si aspetta che non accada. L'argomento di un'aspettativa può non essere necessariamente un termine ground, ma può essere anche una variabile. Si considera anche la negazione esplicita delle aspettative indicata rispettivamente da $\neg\mathbf{E}$ e $\neg\mathbf{EN}$.

Per esempio in un contesto d'asta[10] il seguente atomo:

$$\begin{aligned}&\mathbf{E}(tell(Auctioneer, Bidders, openauction(Item, BasePrice,\\ &MinRise, DtRise, DtAnswer), AuctionID), TOpen)\\ &\wedge\ BasePrice > 0\ \wedge\ MinRise > 0\\ &\wedge\ DtRise > 0\ \wedge\ DtAnswer > 0.\end{aligned} \tag{2.2.3}$$

esprime l'aspettativa che l'agente *Auctioneer*[11], banditore dell'asta identificata con $AuctionID$, attraverso l'atto comunicativo *tell* informi dell'evento *openauction* gli agenti *Bidders*[12]. Si osserva che le variabili possono essere

[8] **E**xpectations.
[9] **E**xpectations **N**ot.
[10] Oggetto di studio di questa tesi.
[11] Banditore d'asta.
[12] Offerenti.

soggette a vincoli CLP per restringere e focalizzare i loro domini d'azione. In questo esempio è richiesto che alcune di esse siano quantità positive.

La $SOKB$ definisce strutture e proprietà chiamate goal, inoltre esprime regole, capacità e conoscenze della società. La SOKB è espressa attraverso un programma logico formato da clausole la cui sintassi è di seguito riportata:

$$\begin{aligned} Clause &::= Atom \leftarrow Body \\ Body &::= ExtLiteral\ [\wedge\ ExtLiteral]^* \\ ExtLiteral &::= Literal \mid Expectation \mid Constraint \\ Expectation &::= [\neg]\ \mathbf{E}(Event\ [, T]) \mid [\neg]\ \mathbf{EN}(Event\ [, T]) \end{aligned} \tag{2.2.4}$$

Diremo che un predicato è definito se esiste il suo nome almeno all'interno della testa ($Atom$) di una clausola. Per esempio la clausola seguente:

$$\begin{aligned} &sold(Item) \leftarrow \\ &\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice)), TOpen) \end{aligned} \tag{2.2.5}$$

afferma che, affinché un articolo $Item$ sia venduto, occorre che un agente A nel ruolo di banditore abbia comunicato agli offerenti $Bidders$ l'apertura un'asta in cui è offerto l'articolo $Item$.

Il goal $\mathcal{G}$ di una società descrive gli obiettivi che la società si prefigge di raggiungere. Anch'esso è espresso da un programma logico e rispetta la sintassi:

$$\begin{aligned} Goal &::= ExtLiteral\ [\wedge\ ExtLiteral]^* \\ ExtLiteral &::= Literal \mid Expectation \mid Constraint \\ Expectation &::= [\neg]\ \mathbf{E}(Event\ [, T]) \mid [\neg]\ \mathbf{EN}(Event\ [, T]) \end{aligned} \tag{2.2.6}$$

identica a quella del $Body$ di una clausola della $SOKB$. Per esempio se l'obiettivo di una società è vendere articoli il goal potrebbe essere:

$$\leftarrow sold(Item) \tag{2.2.7}$$

che potrebbe essere soddisfatto nel caso un agente confermasse la vendita di un articolo tramite un atto comunicativo del tipo:

$$\mathbf{E}(tell(Auctioneer, Bidders, sold(collana), TSold)$$

Il protocollo dell'asta stabilisce il modo in cui la società sia aspetta che gli agenti, banditore ed offerenti, interagiscano. Esso è descritto tramite $\mathcal{IC}_S$ ed ogni vincolo di integrità sociale è espresso come implicazione secondo la seguente sintassi:

$$\begin{aligned}
ICs &::= \chi \rightarrow \phi \\
\chi &::= (HEvent \mid Expectation)\ [\wedge\ BodyLiteral]^* \\
BodyLiteral &::= HEvent \mid Expectation \mid Literal \mid Constraint \\
\phi &::= HeadDisjunction\ [\vee\ HeadDisjunction]^* \mid \bot \\
HeadDisjunction &::= Expectation\ [\wedge\ (Expectation \mid Constraint)]^* \\
Expectation &::= [\neg]\ \mathbf{E}(Event\ [, T]) \mid [\neg]\ \mathbf{EN}(Event\ [, T]) \\
Hevent &::= [\neg]\ \mathbf{H}(Event\ [, T])
\end{aligned} \tag{2.2.8}$$

Per esempio il seguente vincolo:

$$\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid) \\
&\longrightarrow \\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice, \\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen).
\end{aligned} \tag{2.2.9}$$

afferma che se è stata piazzata un'offerta presso il banditore A per l'articolo $Item$ nell'asta $AuctionID$, allora ci si aspetta che A abbia aperto un'asta, identificandola con $AuctionID$, ed offrendo $Item$. In altre parole tale $\mathcal{IC}_S$ evita che nella history **HAP** siano presenti messaggi di offerta relativi ad aste mai aperte.

Per ulteriori dettagli in merito alla visibilità e quantificazione delle variabili nella definizione di $SEKB$, $SOKB$, $\mathcal{G}$ ed $\mathcal{IC}_S$ rimandiamo a [4].

2.3 Programmazione logica abduttiva

Il modello sociale SOCS è stato interpretato in termini di Programmazione Logica Abduttiva ALP[13] [16], e per esso è stata proposta una semantica abduttiva [2]. L'abduzione è stata ampiamente riconosciuta come un potente meccanismo per il ragionamento ipotetico in presenza di informazioni incomplete [6, 17]. La conoscenza imperfetta è trattata etichettando alcune informazioni come *abducibili*, ossia possibili ipotesi che è possibile assumere assicurandosi però che siano consistenti con le conoscenze di base già possedute.

Più formalmente, data una teoria T ed una formula $\mathcal{G}$, obiettivo dell'abduzione è trovare un insieme, di atomi Δ, possibilmente il più piccolo possibile, che insieme a T derivino $\mathcal{G}$, nel rispetto di una qualche definizione di *derivazione* di cui è dotato il linguaggio in cui è espressa T.

Un ALP [16] è una tripla $\langle\, KB,\ \mathcal{A},\ \mathcal{IC}_S \,\rangle$ dove KB è un programma logico, $\mathcal{A}$ è un insieme di predicati non definiti in KB chiamati *abducibili*, $\mathcal{IC}_S$ è un insieme di formule chiamati vincoli di integrità. Una spiegazione abduttiva per il goal $\mathcal{G}$ è un insieme $\Delta \subseteq \mathcal{A}$ tale che $KB \cup \Delta \models \mathcal{IC}_S$, per una qualche notazione di derivazione $\models$.

Nel modello sociale SOCS l'idea è stata quella di sfruttare l'abduzione per definire i comportamenti attesi degli agenti, e di utilizzare la proof procedure abduttiva $\mathcal{S}CIFF$ per *generare* dinamicamente le aspettative ed eseguire i *controlli di conformità*. Per controlli di conformità intendiamo le procedure secondo cui verificare che gli $\mathcal{IC}_S$ non siano violati.

Prima di dare la semantica dichiarativa del modello sociale SOCS, diamo alcune definizioni preliminari.

Definizione 2.1 (Chiusura di una storia)**.** Data una società $\mathcal{S}$ diremo che la sua storia **HAP** è *chiusa*, e scriveremo $\overline{\mathbf{HAP}}$, se è stata chiusa sotto l'assunzione del mondo chiuso CWA[14], ossia quando nessun altro evento futuro potrà accadere in $\mathcal{S}$.

[13] Abductive Logic Programming.
[14] Close World Assumption.

Definizione 2.2 (Istanza di una società)**.** Una istanza $\mathcal{S}_{\mathbf{HAP}}$ di una società $\mathcal{S}$ è rappresentata da una tripla ALP come $\langle\, KB,\, \mathcal{A},\, \mathcal{IC}_S \,\rangle$ dove:

- KB è la conoscenza di base $SOKB$ della società $\mathcal{S}$ insieme alla storia degli eventi accaduti **HAP**;
- $\mathcal{A}$ è l'insieme dei predicati abducibili, chiamati **E**, **EN**, ¬**E**, ¬**EN**;
- $\mathcal{IC}_S$ sono i vincoli di integrità sociale di $\mathcal{S}$.

L'insieme **HAP** caratterizza l'istanza di una società, e rappresenta l'insieme di eventi ground *osservabili* e *rilevanti* per la società.

Definizione 2.3 (Istanza di una società chiusa)**.** Una istanza $\mathcal{S}_{\mathbf{HAP}}$ di una società è *chiusa*, e scriveremo $\mathcal{S}_{\overline{\mathbf{HAP}}}$, quando la sua storia **HAP** è chiusa.

2.4 Una semantica dichiarativa per SOCS

Diamo ora una semantica all'istanza di una società definendo quegli insiemi di aspettative, se esistono, che con la conoscenza sociale di base e gli eventi accaduti implicano una istanza del goal, e che *soddisfano* i vincoli di integrità. La nostra definizione di soddisfacimento degli $\mathcal{IC}_S$ è realizzata attraverso una derivazione all'interno di una logica a tre valori, che permette di lavorare indifferentemente con società aperte o chiuse. Vedremo meglio nella sezione 2.5 come collocheremo nel nostro contesto gli studi effettuati sulla logica a tre valori [18]. Pertanto nel seguito il simbolo $\models$ deve essere interpretato come notazione di *derivazione* all'interno di una *logica a tre valori*.

Definizione 2.4 ($\mathcal{IC}_S$–consistenza)**.** Data una istanza $\mathcal{S}_{\mathbf{HAP}}$ di una società *aperta* ed un insieme **EXP** di aspettative, definiamo **EXP** un insieme $\mathcal{IC}_S$–consistente se:

$$SOKB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{IC}_S \tag{2.4.1}$$

Data una istanza $\mathcal{S}_{\overline{\mathbf{HAP}}}$ di una società *chiusa* ed un insieme **EXP** di aspettative, definiamo **EXP** un insieme $\mathcal{IC}_S$–consistente se:

$$SOKB \cup \overline{\mathbf{HAP}} \cup \mathbf{EXP} \models \mathcal{IC}_S \tag{2.4.2}$$

Un insieme di aspettative $\mathcal{IC}_S$–consistente può essere auto–contraddittorio, per esempio potrebbe contenere[15] sia $\mathbf{E}(p)$ che $\neg\mathbf{E}(p)$. Per questo è necessaria la definizione 2.6.

Definizione 2.5 ($\neg$ –consistenza)**.** Un insieme di aspettative sociali **EXP** è definito $\neg$ –consistente se è solo se per ogni termine p ground:

$$\{\mathbf{E}(p), \neg\mathbf{E}(p) \nsubseteq \mathbf{EXP}\} \qquad \text{e} \qquad \{\mathbf{EN}(p), \neg\mathbf{EN}(p) \nsubseteq \mathbf{EXP}\} \tag{2.4.3}$$

Definizione 2.6 (**E**–consistenza)**.** Un insieme di aspettative sociali **EXP** è definito **E**–consistente se è solo se per ogni termine p ground:

$$\{\mathbf{E}(p), \mathbf{EN}(p) \nsubseteq \mathbf{EXP}\} \tag{2.4.4}$$

Definizione 2.7 (Ammissibilità)**.** Data una istanza di una società aperta $\mathcal{S}_{\mathbf{HAP}}$ (chiusa $\mathcal{S}_{\overline{\mathbf{HAP}}}$), un insieme di aspettative sociali **EXP** è definito *ammissibile* se e solo se è $\mathcal{IC}_S$–consistente, **E**–consistente e $\neg$ –consistente.

La definizione di ammissibilità vale anche per una istanza di società chiusa $\mathcal{S}_{\overline{\mathbf{HAP}}}$ a patto di considerare la relativa definizione di $\mathcal{IC}_S$–consistenza. A volte si distingue fra queste due definizioni con le espressioni "ammissibilità aperta" o "ammissibilità chiusa".

Definizione 2.8 (Fulfillment)**.** Data una istanza di una società aperta $\mathcal{S}_{\mathbf{HAP}}$ (chiusa $\mathcal{S}_{\overline{\mathbf{HAP}}}$), un insieme di aspettative sociali **EXP** è definito *fulfilled* se e solo se per ogni termine p ground:

$$\mathbf{HAP} \cup \mathbf{EXP} \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{EN}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models \bot \tag{2.4.5}$$

[15]Per brevità in questa sezione non riporteremo l'argomento temporale negli eventi e nelle aspettative.

Per una istanza di società la definizione di fulfillment richiede che per ogni aspettativa positiva in **EXP** esista un corrispondente evento accaduto in **HAP**, e che per ogni aspettativa negativa non esista un corrispondente evento accaduto. In caso contrario si parla di violazione.

Definizione 2.9 (Violation)**.** Data una istanza di una società aperta $\mathcal{S}_{\mathbf{HAP}}$ (chiusa $\mathcal{S}_{\overline{\mathbf{HAP}}}$), un insieme di aspettative sociali **EXP** è definito *violato* se e solo se per ogni termine p ground:

$$\mathbf{HAP} \cup \mathbf{EXP} \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{EN}(p) \rightarrow \neg\mathbf{H}(p)\} \models \bot \tag{2.4.6}$$

Definizione 2.10 (Raggiungibilità di un goal)**.** Data una istanza di una società *aperta* $\mathcal{S}_{\mathbf{HAP}}$ ed un goal $\mathcal{G}$ diremo che $\mathcal{G}$ è *raggiungibile*, e scriveremo $\mathcal{S}_{\mathbf{HAP}} \tilde{\models}_{\mathbf{EXP}} \mathcal{G}$, se e solo se esiste un insieme di aspettative sociali **EXP** ammissibile e fulfilled tale che:

$$SOKB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{G} \tag{2.4.7}$$

L'ammissibilità è intesa come "ammissibilità aperta".

Definizione 2.11 (Raggiungimento di un goal)**.** Data una istanza di una società *chiusa* $\mathcal{S}_{\overline{\mathbf{HAP}}}$ ed un goal $\mathcal{G}$ diremo che $\mathcal{G}$ è *raggiunto*, e scriveremo $\mathcal{S}_{\mathbf{HAP}} \models_{\mathbf{EXP}} \mathcal{G}$, se e solo se esiste un insieme di aspettative sociali **EXP** ammissibile e fulfilled tale che:

$$SOKB \cup \overline{\mathbf{HAP}} \cup \mathbf{EXP} \models \mathcal{G} \tag{2.4.8}$$

L'ammissibilità è intesa come "ammissibilità chiusa".

2.5 $\mathcal{S}$CIFF

La proof procedure $\mathcal{S}$CIFF si ispira ad IFF di Fung e Kowalski [10]. IFF è basato su un sistema di transizioni che riscrive formule in altre formule finché nessuna ulteriore transizione è ancora applicabile (quiescenza). Ogni formula è rappresentata come nodo di un albero, ed applicando una transizione si generano uno o più nodi figli. $\mathcal{S}$CIFF ha la necessità di gestire:

- variabili negli abducibili quantificate universalmente;
- arrivo dinamico di eventi;
- consistenza, fulfillment e violazioni;
- vincoli tipo CLP.

Cerchiamo ora di spiegare cosa intendiamo con l'espressione "derivazione" nel contesto di $\mathcal{S}$CIFF.

$\mathcal{S}$CIFF rappresenta ogni nodo come una tupla T:

$$T \equiv \langle R, CS, PSIC, \mathbf{PEXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL} \rangle \tag{2.5.1}$$

dove:

R è una disgiunzione di congiunzioni di atomi, inizialmente impostata al goal $\mathcal{G}$ (è l'equivalente del Risolvente in una risoluzione SLD).

CS è la memoria dei vincoli (come nella CLP).

PSIC è un insieme di implicazioni parzialmente risolte: *Partial Solved social Integrity Constrains*.

PEXP è l'insieme di aspettative pendenti.

HAP è la storia degli eventi accaduti.

FULF è l'insieme delle aspettative fulfilled.

VIOL è l'insieme delle aspettative violate (violated).

Definizione 2.12 (Derivazione)**.** Una *derivazione* D è una sequenza di nodi T_i:

$$T_0 \rightarrow T_1 \rightarrow \ldots \rightarrow T_{n-1} \rightarrow T_n \tag{2.5.2}$$

Dato un goal $\mathcal{G}$, un insieme di vincoli $\mathcal{IC}_S$ ed una storia iniziale $\mathbf{HAP}^0$, che tipicamente è vuota, il primo nodo di una derivazione è:

$$\begin{aligned} T &\equiv \langle R, CS, PSIC, \mathbf{PEXP}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL} \rangle \\ T_0 &= \langle \mathcal{G}, \varnothing, \mathcal{IC}_S, \varnothing, \mathbf{HAP}^0, \varnothing, \varnothing \rangle \end{aligned} \tag{2.5.3}$$

in cui si nota che il risolvente R è settato al goal $\mathcal{G}$ mentre i vincoli parzialmente risolti PSIC sono settati ai vincoli di integrità sociale. I nodi successivi T_i con $i > 0$ sono ottenuti applicando una delle transizioni della La proof procedure fino al nodo T_n di quiescenza, cioè un nodo a cui non è più possibile applicare transizioni. Ulteriori dettagli sulle transizioni possono trovarsi in [11].

Definizione 2.13 (Derivazione di successo)**.** Data una istanza iniziale di una società aperta $\mathcal{S}_{\mathbf{HAP}^0}$ ed una storia iniziale $\mathbf{HAP}^0$ che evolve verso una finale $\mathbf{HAP}^n$, con $\mathbf{HAP}^0 \supseteq \mathbf{HAP}^n$, esiste una *derivazione aperta di successo* per un goal $\mathcal{G}$ se e solo se l'albero di dimostrazione di nodo iniziale:

$$T_0 = \langle \mathcal{G}, \varnothing, \mathcal{IC}_S, \varnothing, \mathbf{HAP}^0, \varnothing, \varnothing \rangle \tag{2.5.4}$$

ha almeno un nodo foglia:

$$T_n = \langle \varnothing, CS, PSIC, \mathbf{PEXP}, \mathbf{HAP}^n, \mathbf{FULF}, \varnothing \rangle \tag{2.5.5}$$

dove CS è consistente, ossia esiste un assegnamento con una variabile ground tale da soddisfare i vincoli CS.

Analogamente esiste una *derivazione chiusa di successo* per un goal $\mathcal{G}$ se e solo se l'albero di dimostrazione di nodo iniziale la 2.5.4 ha almeno un nodo foglia:

$$T_n = \langle \varnothing, CS, PSIC, \mathbf{PEXP}, \overline{\mathbf{HAP}^n}, \mathbf{FULF}, \varnothing \rangle \tag{2.5.6}$$

dove CS è consistente.

Da ogni nodo foglia n di non fallimento è possibile estrarre una risposta in modo molto simile a quanto avviene in IFF. Le risposte della proof procedure

$\mathcal{S}$CIFF sono chiamate *risposte con attesa* e sono costituite da una sostituzione e da un insieme di aspettative abducibili. In particolare si calcola una risposta σ' tale che:

- σ' sostituisce tutte le variabili che non sono quantificate universalmente in n con un termine ground;
- σ' soddisfa tutti i vincoli nella memoria dei vincoli CS_n

La non esistenza di σ' è scoperta durante la fase di estrazione della risposta. In tal caso il nodo è etichettato come un nodo di fallimento e la ricerca procede con l'analisi di un nodo successivo, se esiste.

Definizione 2.14 (Risposta)**.** Dato un nodo di non fallimento n siano:

- σ' la sostituzione di risposta estratta dal nodo n;
- $\sigma = \sigma'|_{vars(\mathcal{G})}$ la restrizione di σ' alle variabili del goal $\mathcal{G}$
- $\mathbf{EXP}_n = (\mathbf{FULF}_n \cup \mathbf{PEXP}_n)\sigma'$ il risultato dell'applicazione della sostituzione all'unione degli insiemi delle aspettative completate e pendenti.

La *risposta con attesa* ottenuta dal nodo n è la coppia:

$$(\mathbf{EXP}_n, \sigma) \tag{2.5.7}$$

2.6 Architettura SOCS–SI

Nel contesto di una società aperta è sorta anche l'esigenza di uno strumento per la verifica della conformità dell'interazione degli agenti ad un dato protocollo.

L'implementazione del SOCS–SI tool permette la verifica "on the fly" della conformità dell'interazione grazie alla proof procedure $\mathcal{S}$CIFF, implementata in Prolog, e grazie ad una serie di moduli, implementati in Java, per l'osservazione dell'interazione fra agenti.

Il cuore dell'architettura SOCS–SI, riportata in figura 2.1, è composta da tre moduli principali:

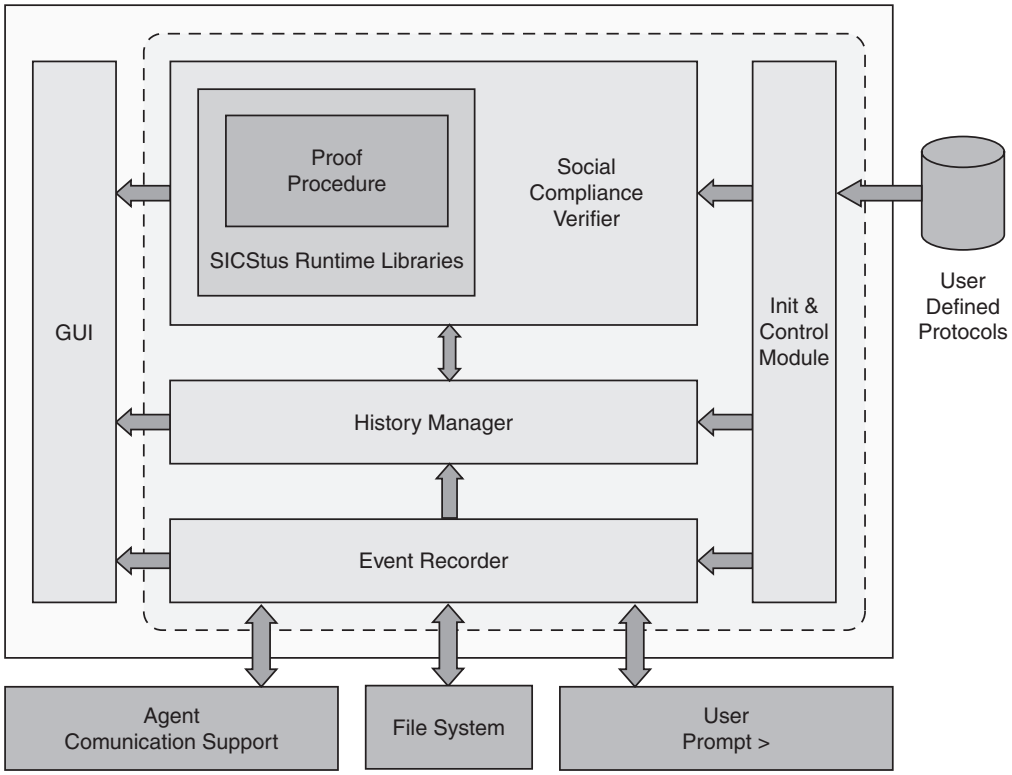

Figura 2.1: Architettura SOCS–SI.

Event Recorder Carica gli eventi da differenti canali sorgenti e li immagazzina nell'*History Manager*.

History Manager Riceve gli eventi dall'*Event Recorder* e li dispone in una "storia di eventi" chiamata *history*.

Social Compliance Verifier Carica gli eventi dall'*History Manager* e li passa alla proof procedure per esaminare la conformità della history alle specifiche.

Nel modello SOCS gli agenti comunicano scambiandosi messaggi, tradotti in eventi **H**. L'*Event Recorder* carica gli eventi e li registra presso l'*History Manager*, in cui diventano disponibili alla proof procedure. Non appena quest'ultima è pronta a processare un nuovo evento il *Social Compliance Verifier* ne carica uno dall'*History Manager* e glielo sottopone. La proof

procedure $\mathcal{S}$CIFF continua ad esaminare eventi finché essi sono disponibili, diversamente sospende la propria attività in attesa che nuovi eventi accadano.

Un quarto modulo chiamato *INIT & Control Module* provvede all'inizializzazione di tutti i componenti in un ordine opportuno. Esso riceve come ingresso iniziale un insieme di protocolli definiti dall'utente, che saranno utilizzati dalla proof per controllare la conformità dell'interazione degli agenti.

Esiste anche un quinto modulo chiamato *GUI*[16] che fornisce un ambiente visuale all'interno del quale è possibile osservare ed ispezionare le varie fasi di analisi del comportamento degli agenti.

Per approfondimenti sui vari moduli implementati in Java, la cui descrizione non è oggetto di questa tesi, si rimanda a [3].

[16]Graphic User Interface.

Capitolo 3

Verifica della conformità dell'interazione fra agenti

In questo capitolo vedremo come l'approccio sociale permetta di progettare protocolli di interazione fra agenti attraverso *vincoli di integrità sociale*. Descriveremo come impiegare $\mathcal{S}$CIFF per verificare la conformità di un insieme di eventi sociali ad un dato protocollo. Inoltre introdurremo g$\mathcal{S}$CIFF, una variante di $\mathcal{S}$CIFF, in grado di generare eventi conformi ad un protocollo. Infine, descriveremo come impiegare g$\mathcal{S}$CIFF per verificare se un dato protocollo goda, o meno, di alcune proprietà.

3.1 Specifica e verifica basate su logica

Abbiamo osservato come, in letteratura, siano state date varie definizioni di *società aperta*. In tale contesto non possiamo assumere, per esempio, che tutti gli agenti abbiano gli stessi comportamenti, desideri ed intenzioni [21], e comunque, non possiamo supporre di poter accedere alla loro architettura interna senza comprometterne l'autonomia, violando la definizione stessa di agente data nel paragrafo 1.1. Perciò la verifica della conformità degli agenti al protocollo societario può essere eseguita solamente attraverso l'osservazione del comportamento esternamente visibile degli agenti.

L'*approccio sociale* del progetto SOCS prevede l'esistenza di una entità autonoma chiamata *Social Compliance Verifier* (SCV), esterno agli agenti, il cui scopo è quello di controllare la conformità degli agenti alle specifiche di interazione societarie. L'SCV è consapevole dell'andamento dell'interazione sociale degli agenti. Quest'ultima è rappresentata da un insieme di fatti (ground) chiamati *eventi*[1] ed indicata per mezzo del funtore **H**.

Dati un insieme di eventi osservati ed una aspettativa sociale è possibile, tramite i vincoli di integrità sociale $\mathcal{IC}_S$, definire quali nuove aspettative devono essere generate dalla società. Per esempio consideriamo l'interazione fra un agente banditore A ed un'agente offerente B:

$$
\begin{aligned}
&\mathbf{H}(tell(A, B, closeauction(ItemList, DtEnd, DtAnswer),\\
&AuctionID), TClose)\\
&\wedge\\
&\mathbf{H}(tell(B, A, bid(ItemBid, Q), AuctionID), TBid)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, B, answer(win, ItemBid, Q), AuctionID), TAnswer)\\
&\wedge\ TClose < TAnswer\ \wedge\ TAnswer \leq TClose + DtAnswer\\
&\vee\\
&\mathbf{E}(tell(A, B, answer(lose, ItemBid, Q), AuctionID), TAnswer)\\
&\wedge\ TClose < TAnswer\ \wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned}
\tag{3.1.1}
$$

Al di là dei vincoli temporali ci interessa evidenziare che se un'asta è stata chiusa ed è stata piazzata da B un'offerta *bid* allora ci si aspetta che A comunichi una risposta *answer* a B di vincita (*win*) o perdita (*lose*). La

[1] La sintassi degli eventi è stata data nel paragrafo 2.2.

seguente interazione:

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(win, ItemBid, QWin),\\&AuctionID), TAnswerWin)\\&\longrightarrow\\&\mathbf{EN}(tell(A, B, answer(lose, ItemBid, QLose),\\&AuctionID), TAnswerLose).\end{aligned} \tag{3.1.2}$$

afferma che se è stata comunicata a B una vincita allora ci si aspetta non possa **EN** essere comunicata a B una perdita. In questo modo siamo in grado di definire protocolli come insiemi di *regole in avanti* che mettono in relazione gli eventi alle aspettative.

In tale prospettiva gli eventi sociali osservati rappresentano fatti noti sul comportamento degli agenti, mentre le aspettative rappresentano ipotesi sul loro comportamento ideale futuro.

Il meccanismo formale che adotteremo in tale contesto dovrà perciò supportare il ragionamento ipotetico e, come abbiamo introdotto nel capitolo 2, l'abduzione è un paradigma che supporta tale caratteristica.

L'idea dietro il framework proposto dal progetto SOCS [1] è quella di formalizzare le aspettative relative al comportamento degli agenti come abducibili, e di utilizzare vincoli di integrità sociale, come 3.1.1 e 3.1.2, per distinguere quelli conformi al protocollo da quelli che lo violano.

3.2 Uso di $\mathcal{S}$CIFF per la verifica delle history

Le history originate dall'agire autonomo degli agenti devono essere conformi al protocollo di interazione societario.

La verifica della conformità della history alle specifiche è effettuata utilizzando $\mathcal{S}$CIFF, come illustrato in figura 3.1, in cui occorre specificare la history, il protocollo, la $SOKB$ ed il goal.

La history è scritta in un file di testo, costituito da un insieme di eventi

i cui termini sono tutti ground. Ogni evento è rappresentato con la sintassi:

$$tell([s0], dialogID, sender, receiver, message, [parList], time) \qquad (3.2.1)$$

in cui sono specificati[2]: il messaggio (message) ed i suoi attributi (parList), il mittente (sender) ed il destinatario (receiver), l'identificativo del contesto del messaggio (dialogID) e l'istante[3] di invio (time). Per esempio nella history:

```
tell([s0], auctID1, auctioneer, bidders, openauction,
[[art1], 19, 100, 50], 0).
tell([s0], auctID1, b1, auctioneer, bid, [[art1], 19], 100).
```

il banditore *auctioneer* informa gli offerenti *bidders* dell'apertura di un'asta tramite il messaggio *openauction* spedito all'istante 0. Successivamente, all'istante 100, l'offerente $b1$ piazza un'offerta tramite il messaggio *bid* presso l'*auctioneer*. Entrambi i messaggi appartengono allo stesso dialogo d'asta $auctID1$.

Il protocollo è stato realizzato impiegando i vincoli di integrità sociale $\mathcal{IC}_S$ che hanno permesso di definire formalmente gli eventi sociali attesi e non attesi. Anch'esso è scritto in un file di testo, per esempio:

```
H( tell( A, Bidders, openauction( Item, BasePrice, DtRise,
DtAnswer), AuctionID), TOpen)
--->
E( tell( B, A, bid( Item, Q), AuctionID), TBid)
/\ TOpen < TBid
/\ TBid <= TOpen + DtRise
```

La conoscenza di base $SOKB$[4] ed il goal $\mathcal{G}$ sono scritti in un file, con il consueto formalismo della Programmazione Logica [19]. Per esempio nel file:

```
society_goal.
```

[2]Il termine $[s0]$ è dovuto all'accesso della history anche da parte dei moduli Java che costituiscono l'architettura SOCS–SI [3] la cui descrizione esula dagli scopi di questa tesi.

[3]Assumiamo l'ipotesi esemplificativa che l'istante in cui il messaggio è spedito dal mittente coincida con quello in cui esso è ricevuto dal destinatario.

[4]Vedi paragrafo 2.2.

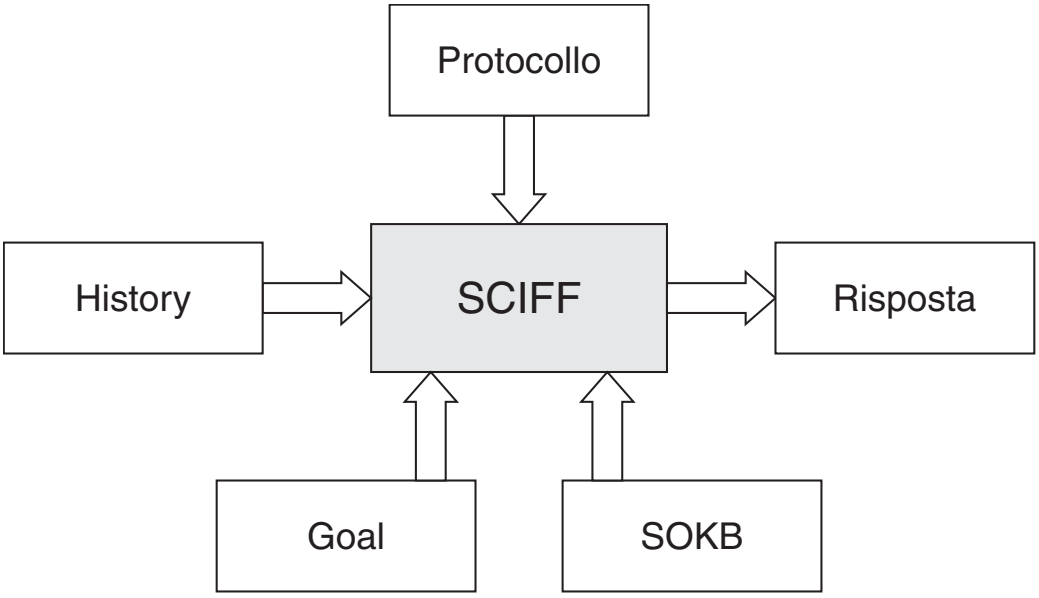

Figura 3.1: Verifica dell'interazione con $\mathcal{S}$CIFF.

```
included([],_).
included([Head|Tail],List):-
member(Head,List),included(Tail,List).

member(Element,[Element|_]).
member(Element,[_|Tail]):-
member(Element,Tail).
```

la conoscenza societaria consiste nelle due clausole `included` e `member` mentre il goal è definito da `society_goal`, in questo caso *true*.

$\mathcal{S}$CIFF elabora la history, il protocollo, la $SOKB$ ed il goal ricevuti in ingresso, e fornisce in uscita una risposta. Se risponde "yes" allora la history è conforme al protocollo altrimenti, se risponde "no", la history non è conforme al protocollo. Diamo allora la seguente:

Definizione 3.1 (compliant)**.** Una history **HAP** è *compliant* ad un protocollo *prot*, e scriveremo:

$$compliant(prot, \mathbf{HAP}) \tag{3.2.2}$$

se $\mathcal{S}$CIFF, operando secondo la metodologia schematizzata in figura 3.1, risponde "yes".

3.3 Uso di $\mathcal{S}$CIFF per la verifica delle proprietà

Affinché un protocollo sia "correttamente" definito esso deve possedere alcune *proprietà* formali. Per esempio, nel progetto delle specifiche di interazione fra agenti che operano in un contesto d'asta, potremmo desiderare che sia garantita la proprietà "se sono state piazzate offerte allora esiste un vincitore". Oppure potremmo richiedere che "se è stata fornita una risposta, in seguito ad un'offerta piazzata per un articolo, allora deve essere stata aperta un'asta in cui si vendeva tale articolo".

La definizione di proprietà può essere data attraverso una metodologia basata sulla logica, ed in alcuni casi, può essere effettuata tramite *vincoli di integrità sociale*. Per esempio la proprietà "se esistono offerte allora deve esistere un vincitore" può essere espressa con il vincolo $\mathcal{IC}_S$:

$$\begin{aligned} &\mathbf{H}(tell(B1, A, bid(Item, Q1), AuctionID), TBid) \\ &\longrightarrow \\ &\mathbf{E}(tell(A, B2, answer(win, Item, Q2), AuctionID), \\ &TAnswer). \end{aligned} \tag{3.3.1}$$

Poiché sia un protocollo che una proprietà sono esprimibili attraverso vincoli di integrità sociale diamo la seguente:

Definizione 3.2 (holds)**.** Una history **HAP** *holds* la proprietà *prop*, e scriveremo:

$$holds(prop, \mathbf{HAP}) \tag{3.3.2}$$

se $\mathcal{S}$CIFF, operando secondo la metodologia schematizzata in figura 3.1, risponde "yes".

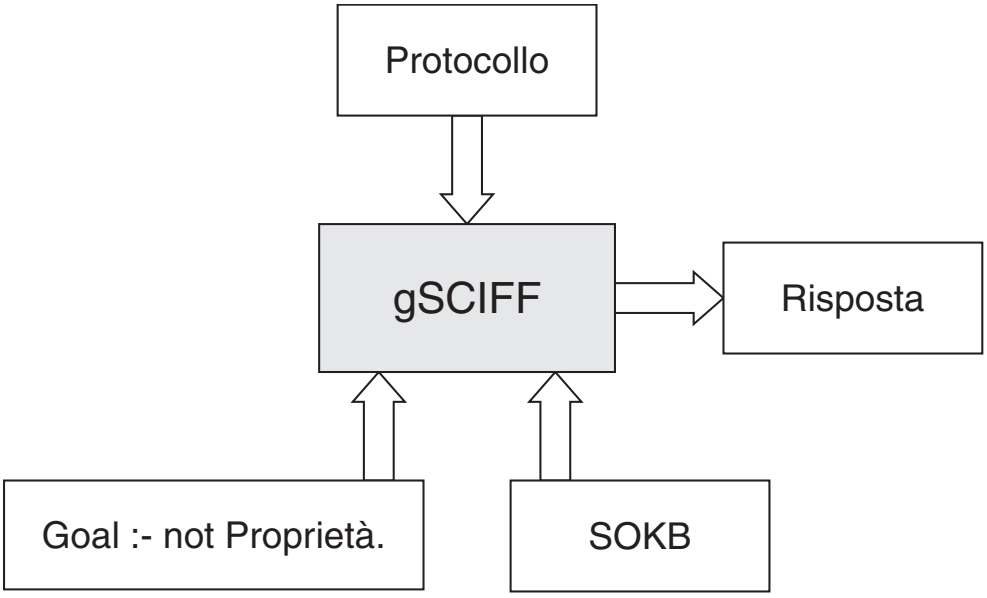

Figura 3.2: Verifica delle proprietà con g$\mathcal{S}$CIFF.

Il fatto che un protocollo goda di una determinata proprietà si può esprimere in questo modo:

$$\forall\, \mathbf{HAP}\ \ prot(\mathbf{HAP}) \rightarrow propr(\mathbf{HAP}) \tag{3.3.3}$$

cioè la proprietà vale per ogni history che soddisfa il protocollo. Applicando una negazione:

$$\begin{aligned} &\neg\, (\, \forall\, \mathbf{HAP}\ \ (\, prot(\mathbf{HAP})\ \rightarrow\ propr(\mathbf{HAP})\,)\,) \\ &\neg\, (\, \forall\, \mathbf{HAP}\ \ (\, \neg\, prot(\mathbf{HAP})\ \vee\ propr(\mathbf{HAP})\,)\,) \\ &\quad \exists\, \mathbf{HAP}\ \ \neg\, (\, \neg\, prot(\mathbf{HAP})\ \vee\ propr(\mathbf{HAP})\,) \\ &\quad \exists\, \mathbf{HAP}\ \ prot(\mathbf{HAP})\ \wedge\ \neg\, propr(\mathbf{HAP}) \end{aligned} \tag{3.3.4}$$

La 3.3.4 può essere provata applicando nuovamente $\mathcal{S}$CIFF. Più in dettaglio utilizzeremo una proof procedure modificata, che chiameremo g$\mathcal{S}$CIFF, dotata di un *generatore di history*, cioè dotata della capacità di generare history compliant al protocollo.

Per confutare la congiunzione fra conformità al protocollo e la negazione della conformità alla proprietà sfrutteremo g$\mathcal{S}$CIFF, avendo cura di speci-

ficare il goal $\mathcal{G}$ come la negazione della proprietà. Per esempio negando la proprietà[5] 3.3.1:

$$\begin{aligned} &\neg\,(\,bid\ \rightarrow\ answer\,).\\ &\neg\,(\,\neg\,bid\ \vee\ answer\,).\\ &bid\ \wedge\ \neg\,answer. \end{aligned} \tag{3.3.5}$$

scriveremo nel file contenente il goal la 3.3.5 con l'usuale formalismo della Programmazione Logica [19]:

```
society_goal:-
e( tell( B1, A, bid( Item, Q1), AuctionID), TBid),
en( tell( A, B2, answer( win, Item, Q2), AuctionID), TAnswer).
```

g$\mathcal{S}$CIFF verifica la proprietà , come riportato in figura 3.2, elaborando il protocollo, il goal e la $SOKB$ ricevuti in ingresso. Per fornire una risposta in uscita la proof procedure tenterà di generare una history conforme al protocollo ed in grado di derivare il goal $\mathcal{G}$.

Poiché g$\mathcal{S}$CIFF è una proof *corretta*, se risponde "yes" significa che esiste un storia conforme al protocollo e contemporaneamente non conforme alla proprietà. Poiché è stato calcolato che la 3.3.4 è vera allora la 3.3.3 è necessariamente falsa, ossia il protocollo *prot* non gode della proprietà *prop*. In tal caso la storia generata rappresenta un controesempio che prova come gli agenti operando in conformità al protocollo producano una history non conforme alla proprietà.

Poiché g$\mathcal{S}$CIFF è una proof non *completa*, il fatto che risponda "no" è una condizione necessaria ma non sufficiente per affermare che il protocollo *prot* gode della proprietà *prop*. In altre parole la proof procedure tenta di generare una history che possegga (holds) la proprietà, ma nel caso in cui non vi riesca tale history potrebbe comunque esistere.

La non completezza di g$\mathcal{S}$CIFF si manifesta anche nel caso in cui la proof non fornisca una risposta in un tempo utile. Con questa espressione

[5]Per brevità riportiamo solo i messaggi tralasciando ogni altra variabile.

vogliamo sottolineare la possibilità che la proof rimanga intrappolata in un loop infinito e pertanto non fornirà né una risposta "no" né una risposta "yes". In tale situazione è impossibile trarre alcuna conclusione.

Capitolo 4

Asta Inglese

In questo capitolo studieremo una società di agenti operanti in uno scenario d'asta: l'asta Inglese. Svilupperemo un protocollo di interazione, motivando le scelte progettuali prese, descrivendolo attraverso AUML e vincoli di integrità sociale. Con esempi pratici vedremo sia come $\mathcal{S}$CIFF verifichi la conformità di una history al protocollo, sia come g$\mathcal{S}$CIFF verifichi se un protocollo goda, o meno, di determinate proprietà.

4.1 Descrizione dell'Asta Inglese

L'asta *Inglese* è molto probabilmente la più conosciuta fra le aste. Essa può essere inquadrata in una tassonomia [26] che la classifica come un'asta *one sided*, *single dimensional* ed *open outcry*. È one sided poiché ci sono molti compratori ed un solo venditore, è single dimensional perché si vende un solo tipo di merce, è open outcry poiché le offerte piazzate dai compratori sono pubbliche.

Nell'asta inglese il banditore fissa un prezzo di partenza per l'articolo in vendita. Gli offerenti possono piazzare in successione varie offerte, alla condizione che ognuna di esse sia superiore alla precedente. Il banditore in genere fissa anche una quota minima di rilancio sull'offerta precedente. La chiusura dell'asta è determinata in base ad una regola che può essere

il raggiungimento di un orario prefissato, può essere l'assenza di offerte in un intervallo di tempo prefissato, altre volte può essere entrambe le regole appena citate oppure un qualunque altro meccanismo. L'ultimo offerente, che per definizione è quello che ha offerto la quota maggiore, vince l'articolo e se lo aggiudica impegnandosi a versare il prezzo massimo offerto.

Un'altra variante dell'asta fpsb è quella che adotta il *prezzo di riserva*. Tale prezzo rappresenta il prezzo minimo a cui il venditore intende vendere l'articolo. Il prezzo di riserva è un *segreto* condiviso fra venditore e banditore e quindi non è comunicato agli offerenti. Il banditore fissa un prezzo di partenza inferiore al prezzo di riserva, successivamente apre l'asta. Al termine dell'asta l'articolo è venduto solo se l'ultima offerta, che per definizione è la maggiore, è uguale o supera il prezzo di riserva. Diversamente se l'ultima offerta è inferiore al prezzo di riserva allora l'articolo non è venduto.

4.2 Scelte di progetto del protocollo

Il progetto del protocollo dell'asta Inglese che abbiamo implementato supporta il meccanismo ad intervallo di tempo per la terminazione dell'asta, e supporta la variante con prezzo di riserva.

Gli atti comunicativi che abbiamo adottato ai fini di una interazione efficace sono i seguenti:

- $openauction(Item, BasePrice, MinRise, DtRise, DtAnswer)$
- $bid(Item, Q)$
- $closeauction(Item, BasePrice, MinRise, DtRise, DtAnswer)$
- $reservedinfo(Item, BasePrice, ReservePrice)$
- $answer(Result, Item, Q)$

dove:

Item è l'articolo messo all'asta.

BasePrice è il prezzo iniziale con cui l'articolo è messo all'asta.

MinRise è la minima quota di rilancio sull'offerta precedente.

DtRise è l'intervallo di tempo entro cui il banditore accetta offerte. Tale intervallo di tempo è misurato a partire dall'istante di apertura dell'asta oppure a partire dall'istante dell'ultima offerta valida. Se in tale intervallo di tempo non è pervenuta alcuna offerta allora il banditore chiude l'asta.

DtAnswer è l'intervallo di tempo entro cui il banditore deve informare gli offerenti dell'esito delle loro offerte.

Q è la quota che un offerente si impegna a pagare nel caso vinca l'articolo messo all'asta.

ReservePrice è il prezzo di riserva dell'articolo messo all'asta. Se il massimo prezzo offerto è inferiore a *ReservePrice* allora l'articolo non è venduto.

Result è l'esito che un offerente riceve in seguito ad un'offerta precedentemente piazzata.

I messaggi *openauction* e *closeauction* sono da intendersi come *broadcast*, ossia rivolti a tutti i membri della società. Ricordiamo che in una società aperta non è possibile sapere a priori quali siano gli agenti partecipanti all'asta. Essi, infatti, sono noti solo nel momento in cui iniziano ad interagire con la società. In questo contesto un messaggio broadcast è da intendere come rivolto a tutti ma non effettivamente inviato ad ogni singolo agente. Tale messaggio è inviato ad un utente *fittizio* chiamato per comodità *bidders* che può essere pensato come un agente "bacheca" consultabile dagli offerenti. Per esempio un evento di apertura d'asta potrebbe essere:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 19, 10, 100, 50], 0).
```

in cui il banditore *auctioneer* informa tutti i *bidders* dell'apertura all'istante 0 di un'asta identificata da *auction*1 in cui si offre l'articolo *art*1 al prezzo base di 19 unità di valuta. L'asta è caratterizzata da una minima quota di rilancio pari a 10, da un intervallo di tempo per il rilancio di 100 e da un intervallo di tempo per la notifica del risultato dell'asta di 50. Un analogo evento di chiusura potrebbe essere:

```
tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 19, 10, 100, 50], 200).
```

Qualora un agente offerente volesse interagire efficacemente con la società dovrebbe assicurarsi di quali aste siano state aperte o chiuse, rilevando in qualche modo i messaggi broadcast inviati dagli agenti banditori alla società.

Il messaggio *bid* è inviato da un offerente al banditore d'asta. Per esempio un evento di offerta potrebbe essere:

```
tell([s0], auction1, b1, auctioneer, bid, [[art1], 39], 100).
```

in cui l'offerente *b*1 piazza un'offerta di 39 unità di valuta presso il banditore *auctioneer* all'istante 100 per l'articolo *art*1 in un'asta identificata da *auction*1.

Il messaggio *answer* è inviato dal banditore ad un offerente. Per esempio un evento di risposta potrebbe essere:

```
tell([s0], auction1, auctioneer, b1, answer,
[ win, [art1], 39], 250).
```

in cui il banditore *auctioneer* informa all'istante 250 che nell'asta identificata da *auction*1 l'offerente *b*1 ha vinto l'articolo *art*1 al prezzo di 39 unità di valuta.

Infine il messaggio *reservedinfo* rappresenta un messaggio *privato* per esplicitare il prezzo di riserva *ReservePrice*. Tale messaggio è un segreto condiviso fra venditore e banditore e non è comunicato ai compratori. Poiché abbiamo modellato la società come costituita di soli agenti banditori e compratori, abbiamo implicitamente assunto che venditore e banditore coincidano. In questo contesto il messaggio privato è inviato dal banditore a se stesso. Per esempio nell'evento:

```
tell([s0], auction1, auctioneer, auctioneer, reservedinfo,
[[art1], 19, 29], 200).
```

il banditore *auctioneer* prende atto, all'istante 200, che l'articolo *art*1 è stato messo all'asta con prezzo base 19 e prezzo di riserva di 29.

4.3 Descrizione del protocollo tramite AUML

Una descrizione del protocollo può essere data attraverso il diagramma temporale, illustrato in figura 4.1, espresso secondo il formalismo AUML (Agent UML) [5]. Tale formalismo rappresenta il tempo con linee tratteggiate, le attività degli agenti con dei rettangoli ed i messaggi con delle frecce. Le frecce possono essere corredate dalla cardinalità degli agenti coinvolti nel messaggio. In tutti i nostri diagrammi temporali utilizzeremo frecce a "mezza punta" che indicano messaggi *asincroni*. Scenari temporali mutuamente esclusivi sono introdotti da un quadrato inclinato.

Le specifiche per l'asta Inglese prevedono che l'auctioneer informi tutti i bidders dell'apertura dell'asta attraverso il messaggio broadcast *openauction*, al quale seguono due scenari alternativi.

Nel primo trascorre l'intervallo di tempo *DtRise* senza che vi sia alcuna offerta. Pertanto al termine di *DtRise* l'auctioneer chiude l'asta attraverso il messaggio broadcast *closeauction*.

Nel secondo un bidder piazza una offerta inviando al banditore d'asta il messaggio *bid*. Ora possono nuovamente verificarsi due scenari alternativi.

Nel primo trascorre l'intervallo di tempo *DtRise* senza che vi sia alcuna offerta. Nel frattempo, poiché è sicuramente già stata piazzata almeno un'offerta, occorre esplicitare il prezzo di riserva attraverso il messaggio *privato reservedinfo* in modo da poter comunicare, ad asta conclusa, l'opportuno esito all'offerente. Al termine di *DtRise* l'auctioneer chiude l'asta attraverso il messaggio broadcast *closeauction*.

Nell'altro caso potrebbe verificarsi una seconda offerta, alla quale succederebbe ricorsivamente una situazione identica a quella precedentemente

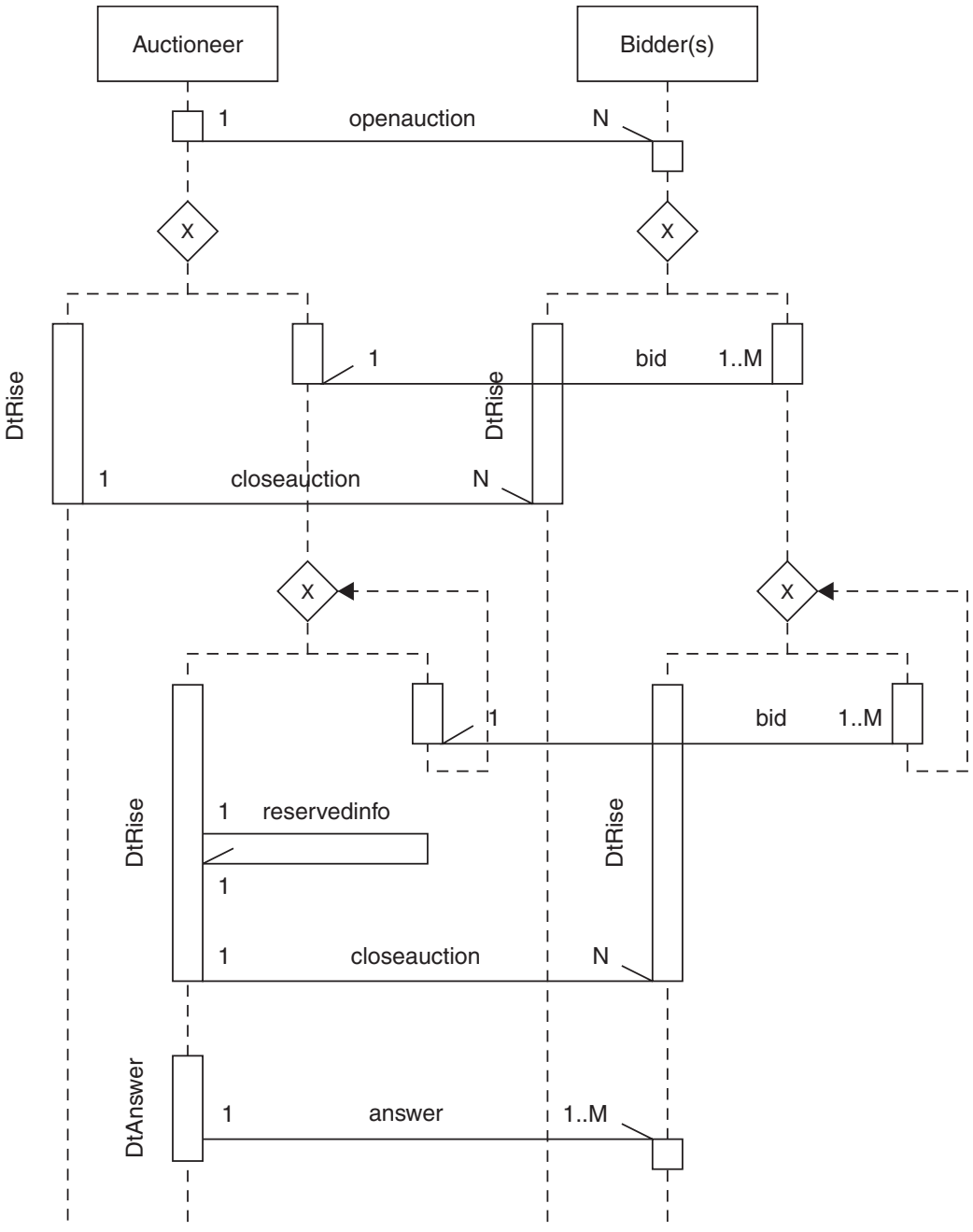

Figura 4.1: Diagramma temporale del protocollo dell'asta Inglese.

descritta. In sintesi potrebbero verificarsi una serie di offerte consecutive alle quali seguirebbe un intervallo *DtRise* privo di offerte che termina con la chiusura dell'asta.

Ad asta terminata l'auctioneer ha a disposizione l'intervallo di tempo *DtAnswer* entro il quale informare, attraverso il messaggio *answer*, ogni bidder dell'esito della relativa offerta.

4.4 Descrizione del protocollo tramite $\mathcal{IC}_S$

Una descrizione più formale del protocollo può essere data attraverso $\mathcal{IC}_S$ che andiamo ora ad analizzare in dettaglio.

Tramite il vincolo di integrità (4.4.1) il protocollo impone consistenza fra apertura e chiusura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata chiusa un'asta, allora ci si aspetta che essa sia stata aperta. I vincoli controllano che i parametri d'asta siano quantità positive. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di chiusura riferiti ad aste mai aperte.

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TClose)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen)\\
&\wedge\ BasePrice > 0\ \wedge\ MinRise > 0\\
&\wedge\ DtRise > 0\ \wedge\ DtAnswer > 0.
\end{aligned}
\tag{4.4.1}
$$

Tramite il vincolo di integrità (4.4.2) il protocollo impone $AuctionID$ diversi per identificare aste diverse. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta, identificata univocamente da $AuctionID$, allora ci si aspetta che non ne sia stata aperta un'altra identificata dallo stesso $AuctionID$. Il vincolo $TOpen1 <> TOpen2$ è necessario per evitare di applicare tale $\mathcal{IC}_S$

ad una stessa *openauction* generando inconsistenza[1]. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi per l'apertura di aste diverse identificate con stesso $AuctionID$.

$$\begin{aligned}
&\mathbf{H}(tell(A1, Bidders1, openauction(Item1, BasePrice1,\\
&MinRise1, DtRise1, DtAnswer1), AuctionID), TOpen1)\\
&\longrightarrow\\
&\mathbf{EN}(tell(A2, Bidders2, openauction(Item2, BasePrice2,\\
&MinRise2, DtRise2, DtAnswer2), AuctionID), TOpen2)\\
&\wedge\ TOpen1 <> TOpen2.
\end{aligned} \tag{4.4.2}$$

Si osserva che è possibile evitare la presenza di messaggi per la chiusura di aste diverse, identificate con stesso $AuctionID$, anche senza inserire il vincolo su *closeauction* analogo al (4.4.2). Infatti un ipotetico *closeauction*, non atteso, implicherebbe attraverso il (4.4.1) il relativo *openauction*, anch'esso non atteso, e già vietato dal (4.4.2).

Tramite il vincolo di integrità (4.4.3) il protocollo impone, dopo l'apertura dell'asta, o un'offerta o la chiusura dell'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se il banditore A ha aperto un'asta $AuctionID$, all'istante $TOpen$, in cui offre l'articolo $Item$, allora ci si aspetta una delle seguenti due alternative. O ci si aspetta che un offerente B piazzi presso A un'offerta Q per l'articolo $Item$ all'istante $TBid$. I vincoli CLP impongono che $TBid$ sia entro $DtRise$ unità di tempo a partire da $TOpen$. La finestra temporale [$TOpen$, $TOpen+DtRise$] individua la *prima* offerta che deve essere caratterizzata da un prezzo Q non inferiore a $BasePrize$, come imposto dal terzo vincolo. Oppure ci si aspetta la chiusura dell'asta. Poiché è trascorso l'intervallo $DtRise$, senza che si sia verificata alcuna offerta, l'istante di chiusura può essere calcolato come $TClose == TOpen + DtRise$.

Tramite il vincolo di integrità (4.4.4) il protocollo impone consistenza

[1] Vedi pag. 18 definizione di E–Consistenza.

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen)\\
&\longrightarrow\\
&\mathbf{E}(tell(B, A, bid(Item, Q), AuctionID), TBid)\\
&\wedge\ TOpen < TBid\\
&\wedge\ TBid \leq TOpen + DtRise\\
&\wedge\ Q \geq BasePrice\\
&\vee\\
&\mathbf{E}(tell(A, Bidders, closeauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TClose)\\
&\wedge\ TClose == TOpen + DtRise.
\end{aligned}
\tag{4.4.3}
$$

d'offerta con l'apertura dell'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata piazzata un'offerta presso il banditore A per l'articolo $Item$ nell'asta $AuctionID$, allora ci si aspetta che A abbia aperto un'asta, identificandola con $AuctionID$, ed offrendo $Item$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di offerta relativi ad aste mai aperte.

$$
\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen).
\end{aligned}
\tag{4.4.4}
$$

Tramite il vincolo di integrità (4.4.5) il protocollo impone, dopo un'offerta, o una successiva offerta o la chiusura dell'asta. In dettaglio la prima parte di tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta e se è stata piazzata un'offerta $Q1$ dall'offerente $B1$ all'istante $TBid1$, allora ci si aspetta che sia

stata piazzata una seconda offerta $Q2$ da un offerente $B2$ all'istante $TBid2$. I vincoli CLP impongono che $TBid2$ sia entro $DtRise$ unità di tempo a partire da $TBid1$. La finestra temporale $[TBid1,\ TBid1 + DtRise]$ individua una *successiva* offerta che deve essere caratterizzata da un prezzo $Q2$ superiore a $Q1$ di almeno $MinRise$, come imposto dal terzo vincolo.

La seconda parte afferma che, in alternativa ad una *successiva* offerta, ci si aspetta la chiusura dell'asta. Poiché è trascorso l'intervallo $DtRise$, senza che si sia verificata alcuna *successiva* offerta, l'istante di chiusura può essere calcolato come $TClose == TBid1 + DtRise$.

Il vincolo (4.4.5) si "attiva" la prima volta solo se è verificato il (4.4.3), ossia in seguito alla prima offerta. Successivamente si "attiva" ricorsivamente su se stesso finché le offerte terminano.

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen)\\
&\wedge\\
&\mathbf{H}(tell(B1, A, bid(Item, Q1), AuctionID), TBid1)\\
&\longrightarrow\\
&\mathbf{E}(tell(B2, A, bid(Item, Q2), AuctionID), TBid2)\\
&\wedge\ TBid1 < TBid2\\
&\wedge\ TBid2 \leq TBid1 + DtRise\\
&\wedge\ Q2 \geq Q1 + MinRise\\
&\vee\\
&\mathbf{E}(tell(A, Bidders, closeauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TClose)\\
&\wedge\ TClose == TBid1 + DtRise.
\end{aligned}
\tag{4.4.5}
$$

Tramite il vincolo di integrità (4.4.6) il protocollo impone, nel caso sia stata aperta e chiusa un'asta e piazzata un'offerta, che sia comunicato tramite

un messaggio *privato* il prezzo di riserva. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'asta è stata aperta all'istante $TOpen$, chiusa all'istante $TClose$, e un offerente ha piazzato un'offerta, allora ci si aspetta che il banditore A comunichi a se stesso (messaggio privato) il prezzo di riserva $ReservePrice$ all'istante $TReservedInfo$. I vincoli CLP impongono che $TReservedInfo$ sia successivo a $TOpen$ e precedente a $TClose$. Inoltre un terzo vincolo impone che $ReservePrice$ non sia inferiore al $BasePrice$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti offerte senza un opportuno messaggio privato relativo al prezzo di riserva.

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TOpen)\\
&\wedge\\
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid)\\
&\wedge\\
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice,\\
&MinRise, DtRise, DtAnswer), AuctionID), TClose) \qquad (4.4.6)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice),\\
&AuctionID), TReservedInfo)\\
&\wedge\ ReservePrice \geq BasePrice\\
&\wedge\ TOpen < TReservedInfo\\
&\wedge\ TReservedInfo \leq TClose.
\end{aligned}
$$

Tramite il vincolo di integrità (4.4.7) il protocollo impone consistenza fra messaggio privato ed apertura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stato comunicato il prezzo di riserva tramite un messaggio privato, allora ci si aspetta che sia stata aperta un'asta e sia stata piazzata almeno un'offerta. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi privati

riferiti ad aste mai aperte. È possibile garantire la validità delle variabili di questa vincolo di integrità senza imporre alcun vincolo CLP. Infatti le aspettative di un *openauction* e di un *bid*, implicitamente, attivano "all'indietro" i precedenti $\mathcal{IC}_S$ che valideranno le variabili.

$$\begin{aligned}
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice),\\
&AuctionID), TReservedInfo)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice,\\
&MinRise, DtEnd, DtAnswer), AuctionID), TOpen)\\
&\wedge\\
&\mathbf{E}(tell(B, A, bid(Item, Q), AuctionID), TBid).
\end{aligned} \tag{4.4.7}$$

Tramite il vincolo di integrità (4.4.8) il protocollo impone di rispondere *win* all'ultimo offerente nel caso sia stato raggiunto, o superato, il prezzo di riserva proposto per l'articolo. Se l'offerente $LastB$ ha piazzato un' offerta $QLastBid$ all'istante $TLastBid$, e se l'asta è stata chiusa all'istante $TClose$, allora il vincolo CLP $TClose == TLastBid + DtRise$ garantisce che $LastB$ sia effettivamente l'ultimo offerente. Inoltre se è stato comunicato tramite *reservedinfo* il prezzo $ReservePrice$ e l'ultima offerta $QLastBid$ non è inferiore a $ReservePrice$ allora ci si aspetta che sia comunicata a $LastB$ la vincita dell'articolo $Item$. Altri vincoli CLP impongono che l'istante $TAnswer$ di notifica del risultato sia dato entro $DtAnswer$ unità di tempo a partire da $TClose$.

Tramite il vincolo di integrità (4.4.9) il protocollo impone di rispondere *lose* all'ultimo offerente nel caso non sia stato raggiunto il prezzo di riserva proposto per l'articolo. In dettaglio tale $\mathcal{IC}_S$ è costruita in modo analogo al (4.4.8) con la sola variante di controllare se $QLastBid < ReservePrice$.

Osserviamo come i vincoli CLP utilizzati in (4.4.8) e (4.4.9) siano un potente mezzo espressivo che non solo valida $TAnswer$ all'interno dell'oppor-

$$
\begin{aligned}
&\mathbf{H}(tell(LastB, A, bid(Item, QLastBid), AuctionID), TLastBid) \\
&\wedge \\
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice), \\
&AuctionID), TReservedInfo) \\
&\wedge \\
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, \\
&MinRise, DtRise, DtAnswer), AuctionID), TClose) \\
&\wedge\ TClose == TLastBid + DtRise \\
&\wedge\ QLastBid \geq ReservePrice \\
&\longrightarrow \\
&\mathbf{E}(tell(A, LastB, answer(win, Item, QLastBid), AuctionID), \\
&TAnswer) \\
&\wedge\ TClose < TAnswer \\
&\wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned}
\tag{4.4.8}
$$

tuna finestra temporale, ma anche focalizza l'attivazione della $\mathcal{IC}_S$ sull'ultimo offerente stabilendo se è, o meno, il vincitore dell'asta. Sottolineiamo questo aspetto in quanto evidenzia come l'approccio sociale sia, in alcuni casi, maturo al punto da definire semantiche di alto livello quali "vincitore o perdente nel contesto di un'Asta Inglese".

$$
\begin{aligned}
&\mathbf{H}(tell(LastB, A, bid(Item, QLastBid), AuctionID), TLastBid) \\
&\wedge \\
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice), \\
&AuctionID), TReservedInfo) \\
&\wedge \\
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, \\
&MinRise, DtRise, DtAnswer), AuctionID), TClose) \\
&\wedge\ TClose == TLastBid + DtRise \\
&\wedge\ QLastBid < ReservePrice \\
&\longrightarrow \\
&\mathbf{E}(tell(A, LastB, answer(lose, Item, QLastBid), AuctionID), \\
&TAnswer) \\
&\wedge\ TClose < TAnswer \\
&\wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned}
\tag{4.4.9}
$$

Tramite il vincolo di integrità (4.4.10) il protocollo impone di rispondere *lose* agli offerenti precedenti l'ultimo. Se l'offerente $LastB$ ha piazzato un' offerta $QLastBid$ all'istante $TLastBid$, e se l'asta è stata chiusa all'istante $TClose$, allora il vincolo CLP $TClose == TLastBid + DtRise$ garantisce che $LastB$ sia effettivamente l'ultimo offerente. Mentre il vincolo $B! = LastB$ garantisce che un offerente B, che ha piazzato l'offerta Q, non sia l'ultimo offerente. Poiché solo l'ultimo offerente è in grado di vincere l'asta, in quanto ha piazzato l'offerta maggiore, ci si aspetta che venga notificata a B la

perdita dell'asta con un messaggio *answer*. Altri vincoli CLP impongono che l'istante $TAnswer$ sia dato entro $DtAnswer$ unità di tempo a partire da $TClose$.

$$
\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid) \\
&\wedge \\
&\mathbf{H}(tell(LastB, A, bid(Item, QLastBid), AuctionID), TLastBid) \\
&\wedge \\
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, \\
&MinRise, DtRise, DtAnswer), AuctionID), TClose) \\
&\wedge\ TClose == TLastBid + DtRise \\
&\wedge\ B! = LastB \\
&\longrightarrow \\
&\mathbf{E}(tell(A, B, answer(lose, Item, Q), AuctionID), TAnswer) \\
&\wedge\ TClose < TAnswer \\
&\wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned}
\tag{4.4.10}
$$

Tramite il vincolo di integrità (4.4.11) il protocollo impone di rispondere in modo non contraddittorio ad un offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una vincita, allora ci si aspetta che non gli sia comunicata una perdita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di vincita che di perdita relativi ad uno stesso offerente.

Tramite il vincolo di integrità (4.4.12) il protocollo impone di rispondere in modo non contraddittorio ad un offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una perdita, allora ci si aspetta che non gli sia comunicata una vincita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di perdita che di vincita relativi ad uno stesso offerente.

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(win, Item, QWin), AuctionID),\\&TAnswerWin)\\&\longrightarrow\\&\mathbf{EN}(tell(A, B, answer(lose, Item, Q2), AuctionID),\\&TAnswer2).\end{aligned} \tag{4.4.11}$$

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(lose, Item, Q), AuctionID),\\&TAnswerLose)\\&\longrightarrow\\&\mathbf{EN}(tell(A, B, answer(win, Item, QWin), AuctionID),\\&TAnswerWin).\end{aligned} \tag{4.4.12}$$

Tramite il vincolo di integrità (4.4.13) il protocollo impone consistenza fra offerta e risposta. In dettaglio tale $\mathcal{IC}_S$ afferma che se un banditore A nell'asta $AuctionID$ ha comunicato una risposta all'offerente B il quale ha offerto Q per l'articolo $Item$, allora ci si aspetta che B abbia piazzato un'offerta presso il banditore A per l'articolo $Item$ offrendo Q all'interno dell'asta $AuctionID$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di risposta riferiti ad offerte mai piazzate.

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(Result, Item, Q), AuctionID), TAnswer)\\&\longrightarrow\\&\mathbf{E}(tell(B, A, bid(Item, Q), AuctionID), TBid).\end{aligned} \tag{4.4.13}$$

4.5 Verifica di conformità con $\mathcal{S}$CIFF

Per verificare se una history è conforme[2] al protocollo abbiamo proceduto secondo quanto descritto nel paragrafo 3.2.

A titolo di esempio riportiamo la seguente history:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 19, 10, 100, 50], 0).

tell([s0], auction1, b1, auctioneer, bid, [[art1], 19], 100).
tell([s0], auction1, b2, auctioneer, bid, [[art1], 29], 200).
tell([s0], auction1, b3, auctioneer, bid, [[art1], 39], 300).
tell([s0], auction1, b4, auctioneer, bid, [[art1], 49], 400).

tell([s0], auction1, auctioneer, auctioneer, reservedinfo,
[[art1], 19, 39], 500).
tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 19, 10, 100, 50], 500).
tell([s0], auction1, auctioneer, b1, answer,
[ lose, [art1], 19], 550).
tell([s0], auction1, auctioneer, b2, answer,
[ lose, [art1], 29], 550).
tell([s0], auction1, auctioneer, b3, answer,
[ lose, [art1], 39], 550).
tell([s0], auction1, auctioneer, b4, answer,
[ win, [art1], 49], 550).
```

In sintesi[3] la simulazione[4] condotta con $\mathcal{S}$CIFF ha prodotto i seguenti risultati:

```
| ?- run.
...
fulf(e(tell(auctioneer,bidders,openauction(
[art1],19,10,100,50),auction1),0)),
fulf(e(tell(b1,auctioneer,bid([art1],19),auction1),100)),
```

[2] Vedi paragrafo 3.2 definizione di history *compliant*.
[3] Le omissioni sono indicate con "...".
[4] Vedi appendice B.1 per il risultato integrale.

```
...
h(tell(auctioneer,bidders,openauction(
[art1],19,10,100,50),auction1),0), ...
h(tell(b1,auctioneer,bid([art1],19),auction1),100), ...
h(tell(b4,auctioneer,bid([art1],49),auction1),400), ...
h(tell(auctioneer,auctioneer,reservedinfo(
[art1],19,39),auction1),500), ...
h(tell(auctioneer,bidders,closeauction(
[art1],19,10,100,50),auction1),500), ...
h(tell(auctioneer,b1,answer(lose,[art1],19),auction1),550), ...
h(tell(auctioneer,b4,answer(win,[art1],49),auction1),550),
... ? ;
no
| ?-
```

In cui notiamo le aspettative sociali fulfilled[5] contrassegnate da `fulf`. Per esempio con il primo `fulf` abbiamo la verifica dell'aspettativa sociale descritta attraverso l'$\mathcal{IC}_S$ (4.4.1), con il secondo `fulf` abbiamo la verifica dell'aspettativa descritta nel vincolo (4.4.3), e così via. Nelle altre righe osserviamo anche le transizioni `h` attraverso le quali $\mathcal{S}$CIFF diviene consapevole degli eventi della history. Infine, tramite il comando ";" abbiamo richiesto alla proof di proseguire la simulazione, ottenendo il termine dell'esecuzione in quanto non vi sono più eventi nella history da analizzare.

Abbiamo sottoposto a $\mathcal{S}$CIFF anche un history non conforme al protocollo:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 19, 10, 100, 50], 0).

tell([s0], auction1, b1, auctioneer, bid, [[art1], 19], 100).
tell([s0], auction1, b2, auctioneer, bid, [[art1], 29], 200).
tell([s0], auction1, b3, auctioneer, bid, [[art1], 39], 300).
tell([s0], auction1, b4, auctioneer, bid, [[art1], 49], 400).

tell([s0], auction1, auctioneer, auctioneer, reservedinfo,
```

[5] Vedi paragrafo 2.3 definizione di *fulfillment*.

```
[[art1], 19, 39], 500).
tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 19, 10, 100, 50], 500).
tell([s0], auction1, auctioneer, b1, answer,
[ lose, [art1], 19], 550).
tell([s0], auction1, auctioneer, b2, answer,
[ lose, [art1], 29], 550).
tell([s0], auction1, auctioneer, b3, answer,
[ lose, [art1], 39], 550).
tell([s0], auction1, auctioneer, b4, answer,
[ lose, [art1], 49], 550).
```

in cui si nota che all'offerente `b4` è notifica una perdita invece che una vincita. La simulazione condotta dalla proof ha prodotto l'esito:

```
| ?- run.
no
| ?-
```

Ossia $\mathcal{S}$CIFF ha risposto dicendo che la history non è conforme al protocollo.

4.6 Verifica di proprietà

La verifica delle proprietà è stata condotta secondo la metodologia esposta nel paragrafo 3.3. Gli esiti delle risposte sono stati salvati in file di testo, in alcuni casi, di notevole dimensione, perciò ne esporremo una sintesi caratterizzata dagli aspetti di maggior interesse per questa tesi. Per completezza riporteremo in appendice alcune brevi risposte in "versione integrale".

In questo paragrafo analizzeremo alcune risposte fornite da g$\mathcal{S}$CIFF, cercando di interpretare i risultati ottenuti.

4.6.1 History di controesempio

Abbiamo voluto verificare la proprietà "ogni history contiene un messaggio *answer*":

$$tell(A, B, answer(win, Item, Q), AuctionID). \tag{4.6.1}$$

La negazione è:

$$\neg\, tell(A, B, answer(win, Item, Q), AuctionID). \tag{4.6.2}$$

pertanto il goal è stato scritto come:

```
society_goal:-
en( tell( A, B, answer( win, Item, Q), AuctionID), TAnswer).
```

In sintesi[6] il risultato è stato il seguente:

```
| ?- run.
... ;
no
| ?-
```

La prima risposta fornita dalla proof è stata una history vuota. Il calcolo di un'eventuale successiva soluzione ha dato esito negativo.

g$\mathcal{S}$CIFF ha risposto "yes" e la history calcolata rappresenta un controesempio che prova come il protocollo non goda della proprietà (4.6.1). In altre parole il protocollo non gode della proprietà poiché, nel caso gli agenti non facciano alcuna azione, il protocollo non prevede nulla, neppure di rispondere win a qualcuno.

Facciamo notare che il controesempio fornito dalla proof è *uno* fra gli esempi possibili. Con analogo significato si può fornire una history caratterizzata, per esempio, dalla sola apertura e chiusura dell'asta. Si tratta di una history non vuota in cui l'agente banditore esegue due azioni. Poiché nessuna offerta è stata piazzata il protocollo non prevede di fornire risposte a qualcuno.

4.6.2 Probabile loop infinito

Nella ricerca di una proprietà posseduta dal protocollo abbiamo cercato di definire meglio la precedente proprietà verificando la "se è stata fornita

[6] Vedi appendice C.1 per il risultato integrale.

una risposta all'agente B per l'articolo $Item$, allora B deve avere piazzato un'offerta per $Item$":

$$\begin{aligned} & tell(A, B, answer(Result, Item, Q), AuctionID) \\ & \longrightarrow \\ & tell(B, A, bid(Item, Q), AuctionID). \end{aligned} \tag{4.6.3}$$

La negazione[7] è:

$$answer \wedge \neg\, bid \tag{4.6.4}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( A, B, answer( Result, Item, Q), AuctionID), TAnswer),
en( tell( B, A, bid( Item, Q), AuctionID), TBid).
```

In sintesi[8] il risultato è stato il seguente:

```
| ?- run.
==Log== ...
==Log== ...
==Log== ...
...
```

La proof è entrata in un *apparente* loop infinito e teoricamente è lecito aspettarsi, prima o poi, una risposta. Nella pratica g$\mathcal{S}$CIFF non ha fornito alcuna risposta *in tempo utile*, per cui ci si è visti obbligati ad interrompere l'esecuzione.

Poiché la proof non ha fornito alcun tipo di risposta è impossibile stabilire se il protocollo goda della proprietà oggetto di verifica.

Nel conteso dell'Asta Inglese, sempre alla ricerca di una proprietà posseduta dal protocollo, abbiamo constatato la difficoltà di g$\mathcal{S}$CIFF nel fornire una risposta. La proof entra frequentemente in loop infinito. Il motivo di ciò è da ricercare nella *ciclicità* del protocollo unita alla strategia *depth first* di Prolog.

[7] D'ora in poi, per brevità, indicheremo il solo messaggio.

[8] Vedi appendice C.4 per il risultato integrale.

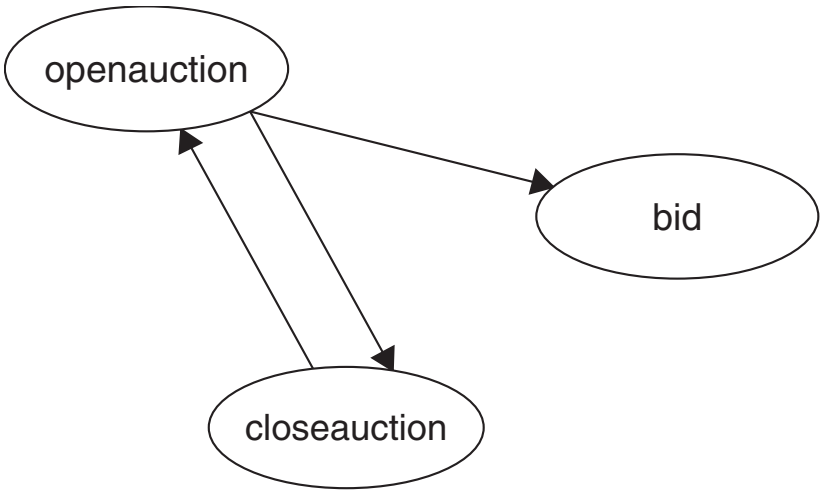

Figura 4.2: $\mathcal{IC}_S$ ciclici.

Con ciclicità del protocollo intendiamo la possibilità di creare percorsi chiusi di eventi ed aspettative. Un esempio banale è rappresentato dai vincoli di integrità sociale (4.4.1) e (4.4.3). Se rappresentiamo i messaggi come nodi di un'albero e l'operatore di implicazione $\longrightarrow$ come un'arco, possiamo osservare in figura 4.2 il ciclo fra i messaggi *openauction* e *closeauction*.

g$\mathcal{S}$CIFF utilizza il protocollo per verificare la proprietà, così può accadere che applicando i due $\mathcal{IC}_S$ in ordine alternato, non riesca a giungere ad una soluzione. L'ordine con cui tali vincoli sono scelti dipende anche da Prolog, il quale adotta una strategia *depth first*, ossia in profondità. Questo significa che nel caso in cui Prolog scelga un ordine che implica richiami ciclici, allora può generare un albero di dimostrazione con un ramo infinito. In realtà la soluzione potrebbe essere in un nodo vicino, come illustrato in figura 4.3, facilmente raggiungibile con un'altra strategia, per esempio *breath first*.

4.6.3 History parziale e probabile loop infinito

Un interessante esempio che evidenzia il modo di operare di g$\mathcal{S}$CIFF è quello emerso dalla verifica della proprietà "se ci sono offerte allora occorrerà

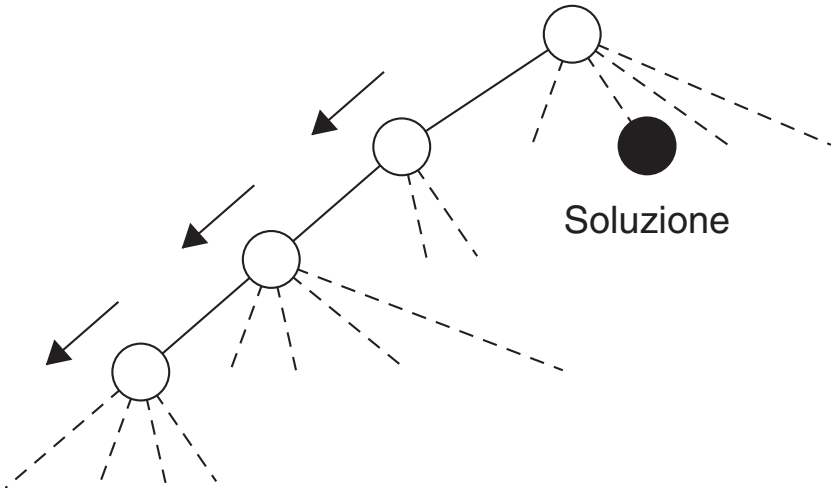

Figura 4.3: Strategia depth first.

rispondere *win* a qualcuno":

$$tell(B, A, bid(Item, Q), AuctionID) \longrightarrow tell(A, LastB, answer(win, Item, QLastBid), AuctionID). \quad (4.6.5)$$

La negazione è:

$$bid \wedge \neg\, answer \quad (4.6.6)$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( B, A, bid( Item, Q), AuctionID), TBid),
en( tell( A, LastB, answer( win, Item, QLastBid), AuctionID),
TAnswer).
```

In sintesi[9] il risultato è stato il seguente:

```
1 | ?- run.
2 ...
3 h(tell(_I,_E,bid(_D,_H),_F),_G), ...
```

[9]Vedi appendice C.5 per il risultato integrale.

```
4  h(tell(_E,_N,openauction(_D,_M,_L,_K,_J),_F),_A), ...
5  _G < _G, ...
6   ? ;
7  ...
8  h(tell(_M,_K,bid(_J,_H),_L),_E), ...
9  h(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A), ...
10 h(tell(_S,_K,bid(_J,_G),_L),_D), ...
11 _D < _D, ...
12  ? ;
13 ...
14 ==Log== ...
15 ==Log== ...
16 ==Log== ...
17 ...
```

Si osserva, nelle righe 3 e 4, che g$\mathcal{S}$CIFF tenta di generare un history di controesempio ma il tentativo fallisce perché un vincolo CLP, in riga 5, non può essere soddisfatto. In riga 6 abbiamo chiesto di calcolare un'altra soluzione, ma il tentativo è nuovamente fallito per via del vincolo in riga 11. Alla richiesta del calcolo di una successiva soluzione g$\mathcal{S}$CIFF entra in loop, apparentemente, infinito, esattamente come descritto in 4.6.2.

4.6.4 Constraint overflow

Un interessante esempio che evidenzia il modo di operare di g$\mathcal{S}$CIFF è quello emerso dalla verifica della proprietà "se è stata chiusa un'asta allora ci si aspetta che essa sia stata aperta":

$$\begin{aligned} &tell(A, Bidders, closeauction(Item, BasePrice, MinRise,\\ &DtRise, DtAnswer), AuctionID)\\ &\longrightarrow \\ &tell(A, Bidders, openauction(Item, BasePrice, MinRise,\\ &DtRise, DtAnswer), AuctionID). \end{aligned} \tag{4.6.7}$$

La negazione è:

$$closeauction \wedge \neg\, openauction \tag{4.6.8}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( A, Bidders, closeauction( Item, BasePrice, MinRise,
DtRise, DtAnswer), AuctionID), TClose),
en( tell( A, Bidders, openauction( Item, BasePrice, MinRise,
DtRise, DtAnswer), AuctionID), TOpen).
```

In sintesi[10] il risultato è stato il seguente:

```
| ?- run.
==Log== ...
==Log== ...
==Log== ...
...
! Representation error in user:'t>=u+c'/3
! CLPFD integer overflow
! goal:  't>=u+c'(_14400,_14051,1)
| ?-
```

La proof interrompe in modo anomalo il calcolo della verifica a causa di un errore di overflow nella rappresentazione degli interi. L'errore non è generato direttamente da g$\mathcal{S}$CIFF, ma bensì dalle librerie Prolog di gestione degli interi. Tuttavia la proof è indirettamente responsabile dell'errore in quanto era impegnata, molto probabilmente, nella costruzione di una successione di vincoli CLP del tipo:

$$T1 > T2 \qquad T2 > T3 \qquad \ldots \qquad T_{n-1} > T_n \tag{4.6.9}$$

in cui per $T1$ era stato fissato un valore, per esempio $T1 = 10$. Ad un certo punto la proof deve aver generato l'n–ma variabile il cui valore, eccedendo la capacità massima fissata per gli interi, ha generato un'eccezione provocando l'interruzione brusca del programma.

Il fatto che g$\mathcal{S}$CIFF stesse generando una successione di vincoli così lunga è indice di una probabile situazione di *apparente* loop infinito. Poiché la proof non ha fornito alcun tipo di risposta è impossibile stabilire se il protocollo goda della proprietà oggetto di verifica.

[10] Vedi appendice C.6 per il risultato integrale.

Capitolo 5

Asta Combinatoria

In questo capitolo studieremo una società di agenti operanti in uno scenario d'asta: l'asta Combinatoria. Svilupperemo un protocollo di interazione, motivando le scelte progettuali prese, descrivendolo attraverso AUML e vincoli di integrità sociale. Con esempi pratici vedremo sia come $\mathcal{S}$CIFF verifichi la conformità di una history al protocollo, sia come g$\mathcal{S}$CIFF verifichi se un protocollo goda, o meno, di determinate proprietà.

5.1 Descrizione dell'Asta Combinatoria

L'asta *Combinatoria* è molto probabilmente l'asta più utilizzata per vendere simultaneamente insiemi eterogenei di articoli. Essa può essere inquadrata in una tassonomia [26] che la classifica come un'asta *one sided*, *multi–dimensional* e *multi–good*. È one sided poiché ci sono molti compratori ed un solo venditore, è multi–dimensional perché si vendono più tipi di merce, è multi–good poiché si possono vendere più pezzi dello stesso articolo.

Nell'asta combinatoria gli offerenti possono piazzare offerte per una combinazione arbitraria di articoli. Il prezzo offerto è quello dell'intera combinazione di articoli che si desidera acquistare. L'asta termina ad un orario prefissato e successivamente il banditore notifica a tutti gli offerenti l'esito della loro offerta.

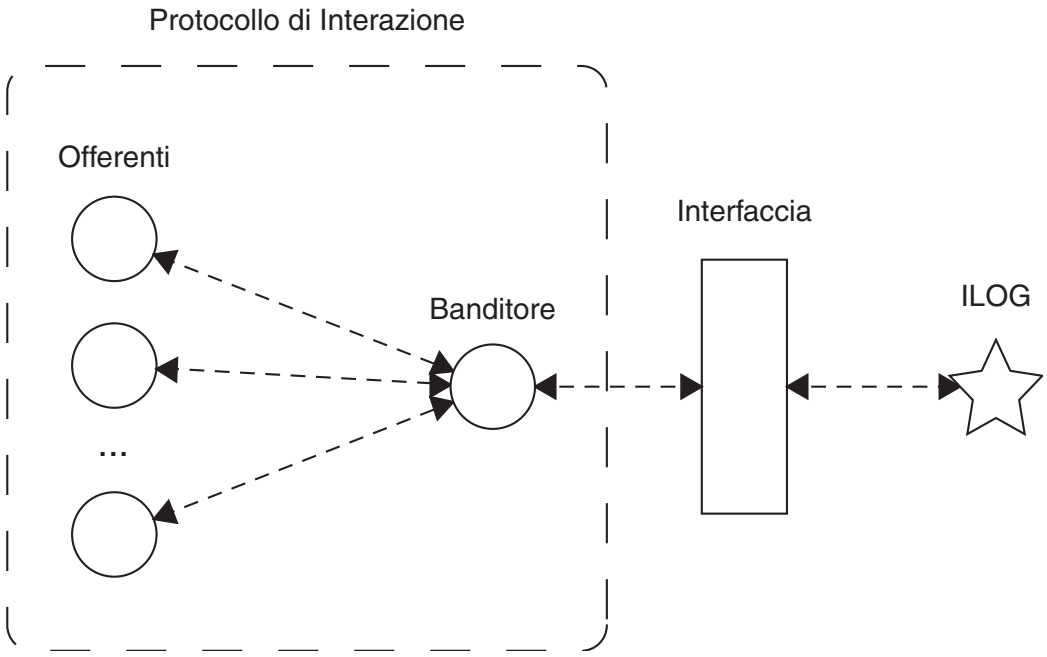

Figura 5.1: Interazione fra agenti e ILOG.

La determinazione del vincitore in un'asta combinatoria costituisce un problema di notevole difficoltà conosciuto come *Winner Determination Problem* (WDP). Nel WDP il banditore deve determinare un sottoinsieme di vincitori, fra gli offerenti, al minimo costo o massimo ricavo. Poiché il WDP ha *complessità NP* la sua soluzione non è direttamente calcolata dal banditore ma è delegata ad un risolutore di vincoli, chiamato ILOG [13], che implementa un algoritmo altamente efficiente sviluppato specificatamente per questo scopo.

Il banditore comunica con ILOG attraverso un'interfaccia come riportato in fig. 5.1. Egli inoltra all'interfaccia le offerte ricevute dagli offerenti e riceve da essa la soluzione del WDP calcolata da ILOG. Oggetto di questa tesi sono il progetto e la verifica del protocollo di interazione fra offerenti e banditore mentre si rimanda a [1] per un approfondimento dell'interazione fra banditore, interfaccia ed ILOG.

5.2 Scelte di progetto del protocollo

Il progetto del protocollo dell'asta Combinatoria che abbiamo implementato supporta il meccanismo di terminazione ad un istante di tempo prefissato.

La *SOKB* di cui è dotata la società è la seguente:

```
notIncluded(SubList,List):-
included(SubList,List),!,fail.
notIncluded(SubList,List).

included([],_).
included([Head|Tail],List):-member(Head,List),
included(Tail,List).

member(Element,[Element|_]).
member(Element,[_|Tail]):-member(Element,Tail).
```

Tramite il predicato *member* la società è in grado di stabilire se un elemento appartiene, o meno, ad una lista. Il predicato *include* permette di stabilire se una lista di articoli è, o non è, inclusa in quella degli articoli offerti. Inverso comportamento, invece, per il predicato *notIncluded*.

Gli atti comunicativi che abbiamo adottato ai fini di una interazione efficace sono i seguenti:

- $openauction(ItemList, DtEnd, DtAnswer)$
- $bid(ItemBid, Q)$
- $closeauction(ItemList, DtEnd, DtAnswer)$
- $answer(Result, ItemBid, Q)$

dove:

ItemList è la lista di articoli messi all'asta.

DtEnd è l'intervallo prefissato entro cui il banditore accetta offerte. Tale intervallo di tempo è misurato a partire dall'istante di apertura dell'asta.

L'istante calcolato come somma fra l'istante di apertura e l'intervallo *DtEnd* è l'istante in cui il banditore decreta la chiusura dell'asta.

DtAnswer è l'intervallo di tempo entro cui il banditore deve informare gli offerenti dell'esito delle loro offerte.

ItemBid è la lista di articoli per i quali l'offerente piazza un'offerta.

Q è la quota che un offerente si impegna a pagare nel caso vinca la lista di articoli per i quali ha piazzato l'offerta.

Result è l'esito dell'offerta piazzata da un offerente.

I messaggi *openauction* e *closeauction* sono indirizzati dal banditore ad ogni singolo offerente. Ricordiamo che in una società aperta non è possibile sapere a priori quali siano gli agenti partecipanti all'asta. Essi, infatti, sono noti solo nel momento in cui iniziano ad interagire con la società. In questo contesto un messaggio di apertura ad un offerente è da intendere come un *invito* a partecipare all'asta, mentre un messaggio di chiusura è da intendere come la comunicazione della fine dell'asta. L'agente invitato all'asta deciderà in piena autonomia se interagire, o meno, con la società. Per esempio un evento di apertura d'asta potrebbe essere:

```
tell([s0], auction1, auctioneer, b1, openauction,
[[i1, i2, i3, i4], 100, 50], 0).
```

in cui il banditore *auctioneer* invita l'offerente $b1$ all'asta identificata da *auction*1 ed aperta all'istante 0, in cui si offrono gli articoli $i1, i2, i3$ ed $i4$. L'asta è caratterizzata da una durata di 100 unità di tempo e da un intervallo per la notifica del risultato dell'asta di 50 unità di tempo. Un analogo evento di chiusura potrebbe essere:

```
tell([s0], auction1, auctioneer, b1, closeauction,
[[i1, i2, i3, i4], 100, 50], 100).
```

Il messaggio *bid* è inviato da un offerente al banditore d'asta. Per esempio un evento di offerta potrebbe essere:

```
tell([s0], auction1, b1, auctioneer, bid, [[i2, i3], 249], 8).
```

in cui l'offerente $b1$ piazza un'offerta di 249 unità di valuta presso il banditore *auctioneer* all'istante 8 per gli articoli $i2$ ed $i3$ in un'asta identificata da *auction*1.

Il messaggio *answer* è inviato dal banditore ad un offerente. Per esempio un evento di risposta potrebbe essere:

```
tell([s0], auction1, auctioneer, b1, answer,
[ lose, [i2, i3], 249], 150).
```

in cui il banditore *auctioneer* informa all'istante 150 che nell'asta identificata da *auction*1 l'offerente $b1$ ha vinto gli articoli $i2$ ed $i3$ al prezzo di 249 unità di valuta.

5.3 Descrizione del protocollo tramite AUML

Una descrizione del protocollo può essere data attraverso il diagramma temporale, illustrato in figura 5.2, espresso secondo il formalismo[1] AUML (Agent UML) [5].

Le specifiche per l'asta Combinatoria prevedono che l'auctioneer informi ogni bidder dell'apertura dell'asta con il messaggio di invito *openauction*, al quale seguono due scenari alternativi.

Nel primo trascorre l'intervallo di tempo *DtEnd* senza che vi sia alcuna offerta. Pertanto al termine di *DtEnd* l'auctioneer chiude l'asta attraverso il messaggio *closeauction*.

Nel secondo un bidder può piazzare un'offerta inviando al banditore d'asta il messaggio *bid*. Al termine dell'intervallo *DtEnd* l'auctioneer chiude l'asta attraverso il messaggio *closeauction*. Ad asta terminata l'auctioneer ha a disposizione l'intervallo di tempo *DtAnswer* entro il quale informare ogni bidder, attraverso il messaggio *answer*, dell'esito della relativa offerta.

[1] Descritto brevemente nel paragrafo 4.3.

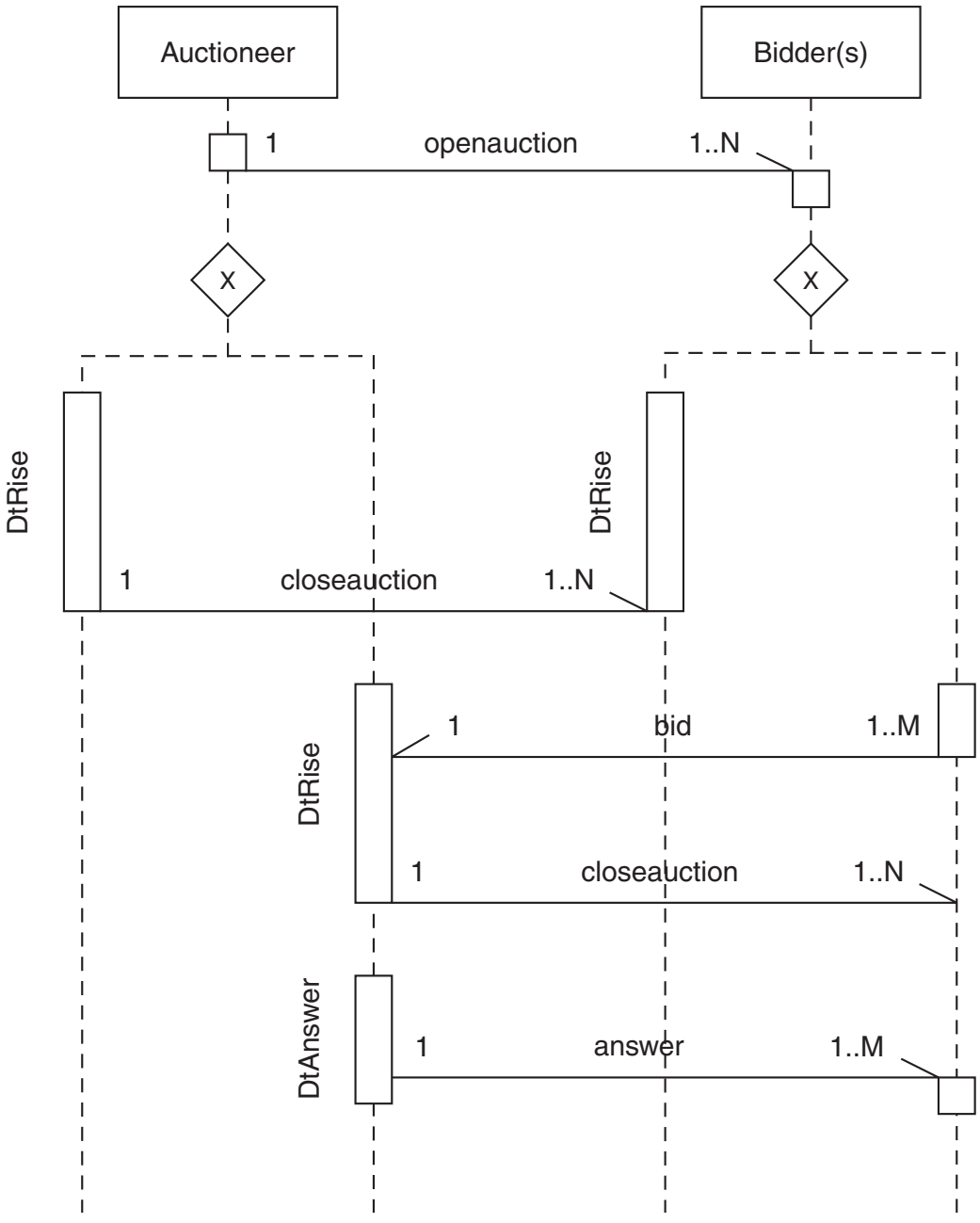

Figura 5.2: Diagramma temporale del protocollo dell'asta Combinatoria.

5.4 Descrizione del protocollo tramite $\mathcal{IC}_S$

Una descrizione più formale del protocollo può essere data attraverso $\mathcal{IC}_S$ che andiamo ora ad analizzare in dettaglio.

Tramite il vincolo di integrità 5.4.1 il protocollo impone consistenza fra apertura e chiusura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta, allora ci si aspetta che essa sia chiusa. I vincoli CLP controllano che i parametri d'asta siano quantità positive. L'asta termina all'istante $TClose$, calcolato come somma fra $TOpen$ e $DtEnd$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di apertura riferiti ad aste mai chiuse.

$$
\begin{aligned}
&\mathbf{H}(tell(A, B, openauction(ItemList, DtEnd, DtAnswer),\\
&AuctionID), TOpen)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, B, closeauction(ItemList, DtEnd, DtAnswer), \qquad (5.4.1)\\
&AuctionID), TClose)\\
&\wedge\ DtEnd > 0\ \wedge\ DtAnswer > 0\\
&\wedge\ TOpen >= 0\ \wedge\ TClose == TOpen + DtEnd.
\end{aligned}
$$

Tramite il vincolo di integrità 5.4.2 il protocollo impone $AuctionID$ diversi per identificare aste diverse. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta, identificata univocamente da $AuctionID$, allora ci si aspetta che non ne sia stata aperta un'altra identificata dallo stesso $AuctionID$. Il vincolo $TOpen1 <> TOpen2$ è necessario per evitare di applicare tale $\mathcal{IC}_S$ ad una stessa *openauction* provocando inconsistenza[2]. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi per l'apertura di aste diverse identificate con stesso $AuctionID$.

Si osserva che è possibile evitare la presenza di messaggi per la chiusura di aste diverse, identificate con stesso $AuctionID$, anche senza inserire il vincolo

[2]Vedi pag. 18 definizione di E–Consistenza.

$$\begin{aligned}
&\mathbf{H}(tell(A1, B1, openauction(ItemList1, DtEnd1, DtAnswer1),\\
&AuctionID), TOpen1)\\
&\longrightarrow\\
&\mathbf{EN}(tell(A2, B2, openauction(ItemList2, DtEnd2, DtAnswer2),\\
&AuctionID), TOpen2)\\
&\wedge\ TOpen1 <> TOpen2.
\end{aligned} \tag{5.4.2}$$

su *closeauction* analogo al (5.4.2). Infatti un ipotetico *closeauction*, non atteso, implicherebbe attraverso il (5.4.3) il relativo *openauction*, anch'esso non atteso, e già vietato dal (5.4.2).

Tramite il vincolo di integrità (5.4.3) il protocollo impone consistenza fra chiusura ed apertura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata chiusa un'asta, allora ci si aspetta che essa sia stata aperta. Non occorre specificare alcun vincolo CLP sulla validità dei parametri in quanto tale $\mathcal{IC}_S$ impone l'esistenza di una *openauction*, che attivando il (5.4.1), vincolerà i parametri. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di chiusura riferiti ad aste mai aperte.

$$\begin{aligned}
&\mathbf{H}(tell(A, B, closeauction(ItemList, DtEnd, DtAnswer),\\
&AuctionID), TClose)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, B, openauction(ItemList, DtEnd, DtAnswer),\\
&AuctionID), TOpen).
\end{aligned} \tag{5.4.3}$$

Tramite il vincolo di integrità (5.4.4) il protocollo impone consistenza d'offerta con l'apertura dell'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'offerente B ha piazzato un'offerta presso il banditore A per il sottoinsieme di articoli $ItemBid$ nell'asta $AuctionID$ all'istante $TBid$, allora ci si aspetta che A

abbia aperto un'asta, identificandola con $AuctionID$, invitando B ed offrendo l'insieme di articoli $ItemList$. Il predicato $included(ItemBid, ItemList)$ verifica che gli articoli specificati nella lista $ItemBid$ siano inclusi nella $ItemList$: in caso affermativo il predicato è vero, diversamente è falso. In tal modo non è possibile piazzare offerte per combinazioni di articoli non inclusi fra quelli messi all'asta. Il vincolo CLP $Q > 0$ impone offerte positive. I vincoli temporali impongono che l'offerta debba essere effettuata all'istante $TBid$ entro $DtEnd$ unità di tempo a partire da $TOpen$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di offerta relativi ad aste mai aperte.

$$
\begin{aligned}
&\mathbf{H}(tell(B, A, bid(ItemBid, Q), AuctionID), TBid) \\
&\longrightarrow \\
&\mathbf{E}(tell(A, B, openauction(ItemList, DtEnd, DtAnswer), \\
&AuctionID), TOpen) \\
&\wedge\ included(ItemBid, ItemList) \\
&\wedge\ Q > 0 \\
&\wedge\ TOpen < TBid\ \wedge\ TBid \leq TOpen + DtEnd.
\end{aligned}
\tag{5.4.4}
$$

Tramite il vincolo di integrità (5.4.5) il protocollo impone l'unicità d'offerta. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'offerente B ha offerto $Q1$, per il sottoinsieme di articoli $ItemBid1$, al banditore A nell'asta identificata da $AuctionID$, allora ci si aspetta che B non piazzi una seconda offerta all'interno della stessa asta. Il vincolo CLP $TBid1 <> TBid2$ è necessario per evitare di applicare tale $\mathcal{IC}_S$ ad una stessa bid, con conseguente inconsistenza[3]. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti più messaggi di offerta piazzati da uno stesso offerente all'interno della stessa asta.

[3]Vedi pag. 18 definizione di E–Consistenza.

$$\begin{aligned}
&\mathbf{H}(tell(B, A, bid(ItemBid1, Q1), AuctionID), TBid1) \\
&\longrightarrow \\
&\mathbf{EN}(tell(B, A, bid(ItemBid2, Q2), AuctionID), TBid2) \\
&\wedge\ TBid1 <> TBid2.
\end{aligned} \tag{5.4.5}$$

Tramite il vincolo di integrità (5.4.6) il protocollo impone di comunicare una risposta agli offerenti. In dettaglio tale $\mathcal{IC}_S$ afferma che se il banditore A ha chiuso l'asta all'istante $TClose$ e l'offerente B ha piazzato un'offerta, allora ci si aspetta che A comunichi a B o una vincita oppure una perdita. La comunicazione del risultato deve avvenire all'istante $TAnswer$ entro $DtAnswer$ unità di tempo a partire da $TClose$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti offerte alle quali non sia stata comunicata una vincita od una perdita.

$$\begin{aligned}
&\mathbf{H}(tell(A, B, closeauction(ItemList, DtEnd, DtAnswer), \\
&AuctionID), TClose) \\
&\wedge \\
&\mathbf{H}(tell(B, A, bid(ItemBid, Q), AuctionID), TBid) \\
&\longrightarrow \\
&\mathbf{E}(tell(A, B, answer(win, ItemBid, Q), AuctionID), TAnswer) \\
&\wedge\ TClose < TAnswer\ \wedge\ TAnswer \leq TClose + DtAnswer \\
&\vee \\
&\mathbf{E}(tell(A, B, answer(lose, ItemBid, Q), AuctionID), TAnswer) \\
&\wedge\ TClose < TAnswer\ \wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned} \tag{5.4.6}$$

Tramite il vincolo di integrità (5.4.7) il protocollo impone consistenza fra risposta ed offerta. In dettaglio tale $\mathcal{IC}_S$ afferma che se un banditore A nell'asta $AuctionID$ ha comunicato una risposta all'offerente B il quale ha

offerto Q per il sottoinsieme di articoli $ItemBid$, allora ci si aspetta che B abbia piazzato un'offerta presso A per il sottoinsieme di articoli $ItemBid$ offrendo Q all'interno dell'asta $AuctionID$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di risposta riferiti ad offerte mai piazzate.

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(Result, ItemBid, Q), AuctionID), TAnswer)\\ &\longrightarrow\\ &\mathbf{E}(tell(B, A, bid(ItemBid, Q), AuctionID), TBid).\end{aligned} \tag{5.4.7}$$

Tramite il vincolo di integrità (5.4.8) il protocollo impone di rispondere in modo non contraddittorio all'offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una vincita, allora ci si aspetta che non gli sia comunicata una perdita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di vincita che di perdita relativi ad uno stesso offerente.

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(win, ItemBid, QWin),\\ &AuctionID), TAnswerWin)\\ &\longrightarrow\\ &\mathbf{EN}(tell(A, B, answer(lose, ItemBid, QLose),\\ &AuctionID), TAnswerLose).\end{aligned} \tag{5.4.8}$$

Tramite il vincolo di integrità (5.4.9) il protocollo impone di rispondere in modo non contraddittorio all'offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una perdita, allora ci si aspetta che non gli sia comunicata una vincita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di perdita che di vincita relativi ad uno stesso offerente.

$$\begin{aligned}
&\mathbf{H}(tell(A, B, answer(lose, ItemBid, QLose),\\
&AuctionID), TAnswerLose)\\
&\longrightarrow \qquad (5.4.9)\\
&\mathbf{EN}(tell(A, B, answer(win, ItemBid, QWin),\\
&AuctionID), TAnswerWin).
\end{aligned}$$

5.5 Verifica di conformità con $\mathcal{S}$CIFF

Per verificare se una history è conforme[4] al protocollo abbiamo proceduto secondo quanto descritto nel paragrafo 3.2.

A titolo di esempio riportiamo la seguente history:

```
tell([s0], auction1, auctioneer, b1, openauction,
[[i1, i2, i3, i4], 100, 50], 0).
tell([s0], auction1, auctioneer, b2, openauction,
[[i1, i2, i3, i4], 100, 50], 0).
tell([s0], auction1, auctioneer, b3, openauction,
[[i1, i2, i3, i4], 100, 50], 0).
tell([s0], auction1, auctioneer, b4, openauction,
[[i1, i2, i3, i4], 100, 50], 0).

tell([s0], auction1, b1, auctioneer, bid,
[[i1            ], 199], 12).
tell([s0], auction1, b2, auctioneer, bid,
[[i1, i2        ], 299], 34).
tell([s0], auction1, b3, auctioneer, bid,
[[    i2, i3    ], 219], 56).
tell([s0], auction1, b4, auctioneer, bid,
[[    i2, i3, i4], 399], 78).

tell([s0], auction1, auctioneer, b1, closeauction,
[[i1, i2, i3, i4], 100, 50], 100).
tell([s0], auction1, auctioneer, b2, closeauction,
```

[4]Vedi paragrafo 3.2 definizione di history *compliant*.

```
[[i1, i2, i3, i4], 100, 50], 100).
tell([s0], auction1, auctioneer, b3, closeauction,
[[i1, i2, i3, i4], 100, 50], 100).
tell([s0], auction1, auctioneer, b4, closeauction,
[[i1, i2, i3, i4], 100, 50], 100).

tell([s0], auction1, auctioneer, b1, answer,
[ win,  [i1            ], 199], 101).
tell([s0], auction1, auctioneer, b2, answer,
[ win,  [i1, i2        ], 299], 112).
tell([s0], auction1, auctioneer, b3, answer,
[ lose, [    i2, i3    ], 219], 123).
tell([s0], auction1, auctioneer, b4, answer,
[ lose, [    i2, i3, i4], 399], 134).
```

In sintesi la simulazione condotta con $\mathcal{S}$CIFF ha prodotto i seguenti risultati:

```
| ?- run.
...
fulf(e(tell(auctioneer,b3,answer(lose,[i2,i3],219),
auction1),123)), ...
fulf(en(tell(auctioneer,b3,answer(win,[i2,i3],_F1),
auction1),_E1)),
...
h(tell(auctioneer,b1,openauction([i1,i2,i3,i4],100,50),
auction1),0), ...
h(tell(auctioneer,b4,answer(lose,[i2,i3,i4],399),
auction1),134), ... ? ;
no
| ?-
```

In cui notiamo le aspettative sociali fulfilled[5] contrassegnate da `fulf`. Per esempio con il primo `fulf` abbiamo la verifica dell'aspettativa sociale descritta attraverso l'$\mathcal{IC}_S$ (5.4.6), con il secondo `fulf` abbiamo la verifica dell'aspettativa descritta nel vincolo (5.4.9), e così via. Nelle altre righe osserviamo anche le transizioni `h` attraverso le quali $\mathcal{S}$CIFF diviene consapevole degli

[5] Vedi paragrafo 2.3 definizione di *fulfillment*.

eventi della history. Infine, tramite il comando ";" abbiamo richiesto alla proof di proseguire la simulazione, ottenendo il termine dell'esecuzione in quanto non vi sono più eventi nella history da analizzare.

Abbiamo sottoposto a $\mathcal{S}$CIFF anche un history non conforme al protocollo, identica alla precedente tranne per l'offerta piazzata da `b1` e per la risposta che egli riceve:

```
...
tell([s0], auction1, b1, auctioneer, bid,
[[i5            ], 199], 12).
...
tell([s0], auction1, auctioneer, b1, answer,
[ lose, [i5            ], 199], 101).
```

in cui si nota che l'offerente `b1` piazza un offerta per un articolo `i5` non incluso fra quelli messi all'asta. Anche se viene comunicata una `answer` di perdita, l'esito della simulazione è:

```
| ?- run.
no
| ?-
```

Ossia $\mathcal{S}$CIFF ha risposto dicendo che la history non è conforme al protocollo.

5.6 Verifica di proprietà

La verifica delle proprietà è stata condotta secondo la metodologia esposta nel paragrafo 3.3. Gli esiti delle risposte sono stati salvati in file di testo, in alcuni casi, di notevole dimensione, perciò ne esporremo una sintesi[6] caratterizzata dagli aspetti di maggior interesse per questa tesi. Per completezza riporteremo in appendice alcune brevi risposte in "versione integrale".

In questo paragrafo analizzeremo alcune risposte fornite da g$\mathcal{S}$CIFF, cercando di interpretare i risultati ottenuti.

[6] Le omissioni sono indicate con "...".

5.6.1 History di controesempio

Abbiamo voluto verificare la proprietà "se è stata chiusa un'asta, allora ci si aspetta che venga comunicato un esito *Result* a qualcuno":

$$tell(A, B, closeauction(ItemList, DtEnd, DtAnswer), AuctionID) \longrightarrow tell(A, B, answer(Result, ItemBid, Q), AuctionID). \quad (5.6.1)$$

La negazione è:

$$closeauction \wedge \neg\, answer. \quad (5.6.2)$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( A, B, closeauction( ItemList, DtEnd, DtAnswer),
AuctionID), TClose),
en( tell( A, B, answer( Result, ItemBid, Q),
AuctionID), TAnswer).
```

In sintesi[7] il risultato è stato il seguente:

```
| ?- run.
...
h(tell(_H,_G,closeauction(_J,_F,_E),_I),_D), ...
h(tell(_H,_G,openauction(_J,_F,_E),_I),_A), ...
_A \= _B,
_A \= _C ? ;
...
h(tell(_G,_F,closeauction(_I,_E,_D),_H),_J), ...
h(tell(_G,_F,openauction(_I,_E,_D),_H),_A), ...
_A \= _B,
_A \= _C ? ;
...
no
| ?-
```

[7] Vedi appendice C.2 per il risultato integrale.

La prima risposta fornita dalla proof è stata una history caratterizzata dagli eventi *openauction* e *closeauction*. Un secondo calcolo ha fornito un analogo risultato. La richiesta di un successivo calcolo ha dato esito negativo facendo terminare l'esecuzione.

g$\mathcal{S}$CIFF ha risposto "yes" e la history calcolata rappresenta un controesempio che prova come il protocollo non gode della proprietà (5.6.1). In altre parole il protocollo non gode della proprietà poiché, nel caso nessun offerente piazzi un'offerta, non è necessario fornire a qualcuno un esito.

Osserviamo infine che g$\mathcal{S}$CIFF avrebbe potuto fornire come controesempio anche una history vuota, come accaduto in 4.6.1. Il risultato calcolato è diverso in quanto, sebbene g$\mathcal{S}$CIFF operi sempre allo stesso modo, il suo risultato dipende, come illustrato in figura 3.2, anche dal tipo di protocollo, dalla $SOKB$ e dal goal.

5.6.2 Probabile successo

Poiché le proprietà si esprimono attraverso vincoli di integrità sociale abbiamo voluto verificare una proprietà simile all'$\mathcal{IC}_S$ (5.4.4), particolarmente interessante per la presenza del predicato *included*:

$$\begin{aligned}
&tell(B, A, bid(ItemBid, Q), AuctionID)\\
&\longrightarrow\\
&tell(A, B, openauction(ItemList, DtEnd, DtAnswer),\\
&AuctionID),\\
&included(ItemBid, ItemList).
\end{aligned} \tag{5.6.3}$$

Per ottenere la negazione abbiamo proceduto come segue:

$$\begin{aligned}
&\neg\,(bid \;\longrightarrow\; (open \;\wedge\; included))\\
&\neg\,(\neg\, bid \;\vee\; (open \;\wedge\; included))\\
&bid \;\wedge\; \neg\,(open \;\wedge\; included)\\
&bid \;\wedge\; (\neg\, open \;\vee\; \neg\, included)\\
&(bid \;\wedge\; \neg\, open) \;\vee\; (bid \;\wedge\; \neg\, included)
\end{aligned} \tag{5.6.4}$$

pertanto il goal è stato scritto come:

```
society_goal:-
(
e( tell( B, A, bid( ItemBid, Q), AuctionID), TBid),
en( tell( A, B, openauction( ItemList, DtEnd, DtAnswer),
AuctionID), TOpen)
);
(
e( tell( B, A, bid( ItemBid, Q), AuctionID), TBid),
notIncluded( ItemBid, ItemList)
).
```

In cui il simbolo "$\vee$" è tradotto col classico formalismo della Programmazione Logica come ";" e la negazione sul predicato *include* è stata ottenuta utilizzando il predicato $notInclude$ definito nella $SOKB$. In sintesi[8] il risultato è stato il seguente:

```
| ?- run.
...
no
| ?-
```

La risposta fornita g$\mathcal{S}$CIFF è "no". Purtroppo, data la non completezza[9] della proof, tale risultato è necessario ma non sufficiente a garantire che il protocollo goda della proprietà (5.6.3).

5.6.3 Probabile loop infinito

Abbiamo voluto verificare la proprietà "se è stata piazzata un'offerta allora ci si aspetta che venga risposto *lose* a qualcuno":

$$\begin{aligned} & tell(B1, A, bid(ItemBid1, Q1), AuctionID) \\ & \longrightarrow \\ & tell(A, B2, answer(lose, ItemBid2, Q2), AuctionID). \end{aligned} \tag{5.6.5}$$

[8] Vedi appendice C.3 per il risultato integrale.

[9] Vedi paragrafo 3.3 *correttezza* e *non completezza* di g$\mathcal{S}$CIFF.

La negazione è:

$$bid \wedge \neg\, answer \tag{5.6.6}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( B1, A, bid( ItemBid1, Q1), AuctionID), TBid),
en( tell( A, B2, answer( lose, ItemBid2, Q2), AuctionID),
TAnswer).
```

In sintesi il risultato è stato il seguente:

```
| ?- run.
==Log== ...
==Log== ...
==Log== ...
...
```

La proof è entrata in un *apparente* loop infinito e teoricamente è lecito aspettarsi, prima o poi, una risposta. Nella pratica g$\mathcal{S}$CIFF non ha fornito alcuna risposta *in tempo utile*, per cui ci si è visti obbligati ad interrompere l'esecuzione. I motivi alla base dell'entrata in loop della proof sono stati analizzati in dettaglio nella sezione 4.6.2.

Poiché la proof non ha fornito alcun tipo di risposta è impossibile stabilire se il protocollo goda della proprietà oggetto di verifica.

Capitolo 6

Asta FPSB

In questo capitolo studieremo una società di agenti operanti in uno scenario d'asta: l'asta First Price Sealed Bid. Svilupperemo un protocollo di interazione, motivando le scelte progettuali prese, descrivendolo attraverso AUML e vincoli di integrità sociale. Con esempi pratici vedremo sia come $\mathcal{S}$CIFF verifichi la conformità di una history al protocollo, sia come g$\mathcal{S}$CIFF verifichi se un protocollo goda, o meno, di determinate proprietà.

6.1 Descrizione dell'Asta FPSB

L'asta *First Price Sealed Bid* (FPSB) è utilizzata nell'ambito degli scambi esteri e per il rifinanziamento di crediti. Essa può essere inquadrata in una tassonomia [26] che la classifica come un'asta *one sided*, *single dimensional* e *sealed bid*. È one sided perché ci sono molti compratori ed un solo venditore, è single dimensional poiché si vende un solo tipo di merce, è sealed bid poiché l'offerta piazzata è segreta, ossia non è dichiarata pubblicamente ed è un segreto condiviso solamente fra offerente e banditore.

Nell'asta FPSB gli offerenti piazzano le loro offerte presso il banditore in forma privata. L'asta termina ad un orario prefissato e successivamente il banditore notifica a tutti gli offerenti l'esito della loro offerta. L'offerente che

ha offerto la quota maggiore vince l'articolo messo all'asta e se lo aggiudica impegnandosi a versare il prezzo offerto.

L'asta FPSB appartiene alla famiglia delle *K–th Price Sealed Bid* (KPSB), in cui l'offerente che ha offerto la quota maggiore vince l'articolo messo all'asta ma se lo aggiudica impegnandosi a versare non quanto egli ha offerto, bensì il k–mo prezzo offerto in ordine decrescente di valore. Per esempio nella *Second Price Sealed Bid* l'offerente che ha offerto la quota maggiore vince l'articolo messo all'asta e se lo aggiudica impegnandosi a versare il prezzo dell'offerta immediatamente inferiore alla sua, cioè pagherà il secondo prezzo offerto in ordine decrescente di valore.

Un'altra variante dell'asta fpsb è quella che adotta il *prezzo di riserva*. Tale prezzo rappresenta il prezzo minimo a cui il venditore intende vendere l'articolo. Il prezzo di riserva è un segreto condiviso fra venditore e banditore e quindi non è comunicato agli offerenti. Il banditore fissa un prezzo di partenza inferiore al prezzo di riserva, successivamente apre l'asta. Al termine dell'asta l'articolo è venduto solo se la prima offerta è uguale o supera il prezzo di riserva. Diversamente se la prima offerta è inferiore al prezzo di riserva allora l'articolo non è venduto.

6.2 Scelte di progetto del protocollo

Il progetto del protocollo dell'asta FPSB che abbiamo implementato supporta il meccanismo ad istante di tempo prefissato per la terminazione dell'asta, e supporta la variante con prezzo di riserva.

Gli atti comunicativi che abbiamo adottato ai fini di una interazione efficace sono i seguenti:

- $openauction(Item, BasePrice, DtEnd, DtAnswer)$
- $bid(Item, Q)$
- $closeauction(Item, BasePrice, DtEnd, DtAnswer)$
- $reservedinfo(Item, BasePrice, ReservePrice)$

- $answer(Result, Item, Q)$

dove:

Item è l'articolo messo all'asta.

BasePrice è il prezzo iniziale con cui l'articolo è messo all'asta.

DtEnd è l'intervallo prefissato entro cui il banditore accetta offerte. Tale intervallo di tempo è misurato a partire dall'istante di apertura dell'asta. L'istante calcolato come somma fra l'istante di apertura e l'intervallo *DtEnd* è l'istante in cui il banditore decreta la chiusura dell'asta.

DtAnswer è l'intervallo di tempo entro cui il banditore deve informare gli offerenti dell'esito delle loro offerte.

Q è la quota che un offerente si impegna a pagare nel caso vinca l'articolo messo all'asta.

ReservePrice è il prezzo di riserva dell'articolo messo all'asta. Se il primo prezzo offerto è inferiore a *ReservePrice* allora l'articolo non è venduto.

Result è l'esito che un offerente riceve in seguito ad un'offerta precedentemente piazzata.

I messaggi *openauction* e *closeauction* sono da intendersi come *broadcast*, ossia rivolti a tutti i membri della società. Ricordiamo che in una società aperta non è possibile sapere a priori quali siano gli agenti partecipanti all'asta. Essi, infatti, sono noti solo nel momento in cui iniziano ad interagire con la società. In questo contesto un messaggio broadcast è da intendere come rivolto a tutti ma non effettivamente inviato ad ogni singolo agente. Tale messaggio è inviato ad un utente *fittizio* chiamato per comodità *bidders* che può essere pensato come un agente "bacheca" consultabile dagli offerenti. Per esempio un evento di apertura d'asta potrebbe essere:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 19, 100, 50], 0).
```

in cui il banditore *auctioneer* informa tutti i *bidders* dell'apertura all'istante 0 di un'asta identificata da *auction*1 in cui si offre l'articolo *art*1 al prezzo base di 19 unità di valuta. L'asta è caratterizzata da una durata di 100 unità di tempo e da un intervallo per la notifica del risultato dell'asta di 50 unità di tempo. Un analogo evento di chiusura potrebbe essere:

```
tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 19, 100, 50], 100).
```

Qualora un agente offerente volesse interagire efficacemente con la società dovrebbe assicurarsi di quali aste siano state aperte o chiuse, rilevando in qualche modo i messaggi broadcast inviati dagli agenti banditori alla società.

Il messaggio *bid* è inviato da un offerente al banditore d'asta. Per esempio un evento di offerta potrebbe essere:

```
tell([s0], auction1, b1, auctioneer, bid, [[art1], 39], 50).
```

in cui l'offerente *b*1 piazza un'offerta di 39 unità di valuta presso il banditore *auctioneer* all'istante 50 per l'articolo *art*1 in un'asta identificata da *auction*1.

Il messaggio *answer* è inviato dal banditore ad un offerente. Per esempio un evento di risposta potrebbe essere:

```
tell([s0], auction1, auctioneer, b1, answer,
[ win, [art1], 39], 150).
```

in cui il banditore *auctioneer* informa all'istante 150 che nell'asta identificata da *auction*1 l'offerente *b*1 ha vinto l'articolo *art*1 al prezzo di 39 unità di valuta.

Infine il messaggio *reservedinfo* rappresenta un messaggio *privato* per esplicitare il prezzo di riserva *ReservePrice*. Tale messaggio è un segreto condiviso fra venditore e banditore e non è comunicato ai compratori. Poiché abbiamo modellato la società come costituita di soli agenti banditori e compratori, abbiamo implicitamente assunto che venditore e banditore coincidano. In questo contesto il messaggio privato è inviato dal banditore a se stesso. Per esempio nell'evento:

```
tell([s0], auction1, auctioneer, auctioneer, reservedinfo,
[[art1], 19, 29], 100).
```

il banditore *auctioneer* prende atto, all'istante 100, che l'articolo *art*1 è stato messo all'asta con prezzo base 19 e prezzo di riserva 29.

6.3 Descrizione del protocollo tramite AUML

Una descrizione del protocollo può essere data attraverso il diagramma temporale, illustrato in figura 6.1, espresso secondo il formalismo[1] AUML (Agent UML) [5].

Le specifiche per l'asta Combinatoria prevedono che l'auctioneer informi tutti i bidders dell'apertura dell'asta attraverso il messaggio broadcast *openauction*, al quale seguono due scenari alternativi.

Nel primo trascorre l'intervallo di tempo *DtEnd* senza che vi sia alcuna offerta. Pertanto al termine di *DtEnd* l'auctioneer chiude l'asta attraverso il messaggio broadcast *closeauction*.

Nel secondo un bidder può piazzare una offerta inviando al banditore d'asta il messaggio *bid*. Nel frattempo, poiché è stata piazzata almeno un'offerta, occorre esplicitare il prezzo di riserva attraverso il messaggio *privato reservedinfo* in modo da poter comunicare, ad asta conclusa, l'opportuno esito all'offerente. Al termine dell'intervallo *DtEnd* l'auctioneer chiude l'asta attraverso il messaggio *closeauction*.

Ad asta terminata l'auctioneer ha a disposizione l'intervallo di tempo *DtAnswer* entro il quale informare, attraverso il messaggio *answer*, ogni bidder dell'esito della relativa offerta.

6.4 Descrizione del protocollo tramite $\mathcal{IC}_S$

Una descrizione più formale del protocollo può essere data attraverso $\mathcal{IC}_S$ che andiamo ora ad analizzare in dettaglio.

[1] Descritto brevemente nel paragrafo 4.3.

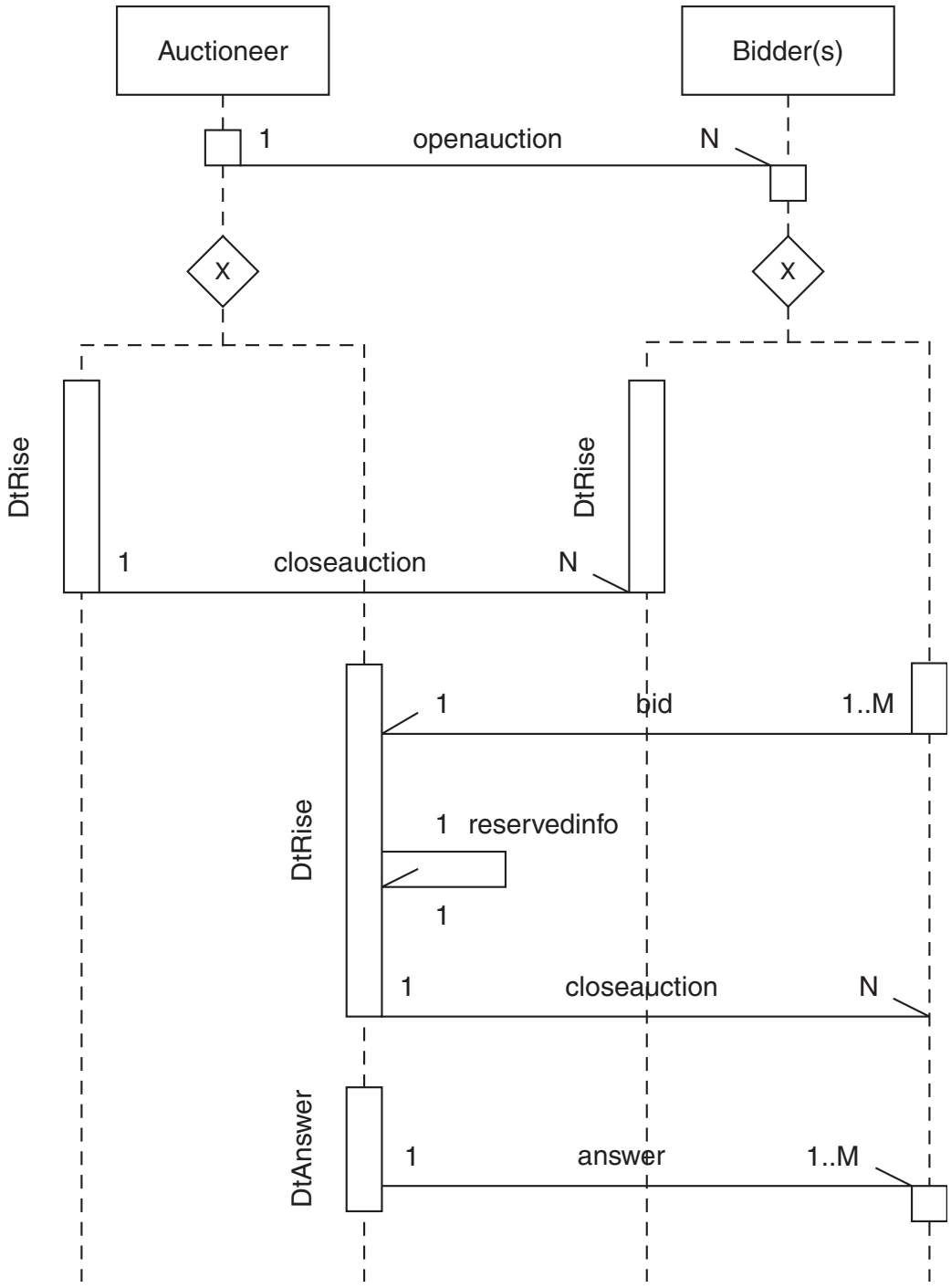

Figura 6.1: Diagramma temporale del protocollo dell'asta FPSB.

Tramite il vincolo di integrità (6.4.1) il protocollo impone consistenza fra apertura e chiusura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta, allora ci si aspetta che essa sia stata chiusa. I vincoli CLP controllano che i parametri d'asta siano quantità positive. L'asta termina all'istante $TClose$, calcolato come somma fra $TOpen$ e $DtEnd$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di apertura riferiti ad aste mai chiuse.

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, openauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TOpen)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, closeauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TClose)\\
&\wedge\ BasePrice > 0\\
&\wedge\ DtEnd > 0\\
&\wedge\ DtAnswer > 0\\
&\wedge\ TOpen >= 0\\
&\wedge\ TClose == TOpen + DtEnd.
\end{aligned}
\tag{6.4.1}
$$

Tramite il vincolo di integrità (6.4.2) il protocollo impone $AuctionID$ diversi per identificare aste diverse. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata aperta un'asta, identificata univocamente da $AuctionID$, allora ci si aspetta che non ne sia stata aperta un'altra identificata dallo stesso $AuctionID$. Il vincolo $TOpen1 <> TOpen2$ è necessario per evitare di applicare tale $\mathcal{IC}_S$ ad una stessa *openauction* provocando inconsistenza[2]. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi per l'apertura di aste diverse identificate con stesso $AuctionID$.

Si osserva che è possibile evitare la presenza di messaggi per la chiusura di

[2]Vedi pag. 18 definizione di E–Consistenza.

$$\begin{aligned}
&\mathbf{H}(tell(A1, Bidders1, openauction(Item1, BasePrice1, DtEnd1,\\
&DtAnswer1), AuctionID), TOpen1)\\
&\longrightarrow\\
&\mathbf{EN}(tell(A2, Bidders2, openauction(Item2, BasePrice2, DtEnd2,\\
&DtAnswer2), AuctionID), TOpen2)\\
&\wedge\ TOpen1 <> TOpen2.
\end{aligned} \tag{6.4.2}$$

aste diverse, identificate con stesso $AuctionID$, anche senza inserire il vincolo su *closeauction* analogo al (6.4.2). Infatti un ipotetico *closeauction*, non atteso, implicherebbe attraverso il (6.4.3) il relativo *openauction*, anch'esso non atteso, e già vietato dal (6.4.2).

Tramite il vincolo di integrità (6.4.3) il protocollo impone consistenza fra chiusura ed apertura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata chiusa un'asta, allora ci si aspetta che essa sia stata aperta. Non occorre specificare alcun vincolo CLP sulla validità dei parametri in quanto tale $\mathcal{IC}_S$ impone l'esistenza di una *openauction*, che attivando il (6.4.1), vincolerà i parametri. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di chiusura riferiti ad aste mai aperte.

$$\begin{aligned}
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TClose)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TOpen).
\end{aligned} \tag{6.4.3}$$

Tramite il vincolo di integrità (6.4.4) il protocollo impone consistenza d'offerta con l'apertura dell'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stata piazzata un'offerta presso il banditore A per l'articolo $Item$ nell'asta

$AuctionID$, allora ci si aspetta che A abbia aperto un'asta, identificandola con $AuctionID$, ed offrendo l'articolo $Item$. Il vincolo CLP $Q > BasePrice$ impone offerte superiori al prezzo di base fissato per l'articolo. I vincoli temporali impongono che l'offerta debba essere effettuata all'istante $TBid$ entro $DtEnd$ unità di tempo a partire da $TOpen$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di offerta relativi ad aste mai aperte.

$$\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid) \\
&\longrightarrow \\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice, DtEnd, \\
&DtAnswer), AuctionID), TOpen) \\
&\wedge\ Q > BasePrice \\
&\wedge\ TOpen < TBid \\
&\wedge\ TBid \leq TOpen + DtEnd.
\end{aligned} \tag{6.4.4}$$

Tramite il vincolo di integrità (6.4.5) il protocollo impone l'unicità d'offerta. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'offerente B ha offerto $Q1$ per l'articolo $Item$ al banditore A nell'asta identificata da $AuctionID$, allora ci si aspetta che B non piazzi una seconda offerta all'interno della stessa asta. Il vincolo $TBid1 <> TBid2$ è necessario per evitare di applicare tale $\mathcal{IC}_S$ ad una stessa bid provocando inconsistenza[3]. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti più messaggi di offerta piazzati da uno stesso offerente all'interno della stessa asta.

Tramite il vincolo di integrità (6.4.6) il protocollo impone, nel caso vi sia stata un'offerta, un messaggio privato per la notifica del prezzo di riserva per l'articolo. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'asta è stata aperta all'istante $TOpen$, chiusa all'istante $TClose$, e un offerente ha piazzato un'offerta, allora ci si aspetta che il banditore A comunichi a se stesso (messaggio privato)

[3]Vedi pag. 18 definizione di E–Consistenza.

$$\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q1), AuctionID), TBid1) \\
&\longrightarrow \\
&\mathbf{EN}(tell(B, A, bid(Item, Q2), AuctionID), TBid2) \\
&\wedge\ TBid1 <> TBid2.
\end{aligned} \tag{6.4.5}$$

il prezzo di riserva $ReservePrice$ all'istante $TReservedInfo$, successivo a $TOpen$ e precedente a $TClose$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti offerte senza un opportuno messaggio privato relativo al prezzo di riserva.

$$\begin{aligned}
&\mathbf{H}(tell(A, Bidders, openauction(Item, BasePrice, DtEnd, \\
&DtAnswer), AuctionID), TOpen) \\
&\wedge \\
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid) \\
&\wedge \\
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, DtEnd, \\
&DtAnswer), AuctionID), TClose) \\
&\longrightarrow \\
&\mathbf{E}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice), \\
&AuctionID), TReservedInfo) \\
&\wedge\ ReservePrice \geq BasePrice \\
&\wedge\ TOpen < TReservedInfo \\
&\wedge\ TReservedInfo \leq TClose.
\end{aligned} \tag{6.4.6}$$

Tramite il vincolo di integrità (6.4.7) il protocollo impone consistenza fra messaggio privato ed apertura d'asta. In dettaglio tale $\mathcal{IC}_S$ afferma che se è stato comunicato il prezzo di riserva tramite un messaggio privato, allora ci

si aspetta che sia stata aperta un'asta e sia stata piazzata almeno un'offerta. È possibile garantire la validità delle variabili di questa vincolo di integrità senza imporre alcun vincolo CLP. Infatti le aspettative di un *openauction* e di un *bid*, implicitamente, attivano "all'indietro" i precedenti $\mathcal{IC}_S$ che valideranno le variabili. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi privati riferiti ad aste mai aperte.

$$\begin{aligned}
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice),\\
&AuctionID), TReservedInfo)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, Bidders, openauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TOpen)\\
&\wedge\\
&\mathbf{E}(tell(B, A, bid(Item, Q), AuctionID), TBid).
\end{aligned} \tag{6.4.7}$$

Tramite il vincolo di integrità (6.4.8) il protocollo impone di comunicare una risposta agli offerenti. In dettaglio tale $\mathcal{IC}_S$ afferma che se l'asta è stata chiusa e l'offerente B ha piazzato un'offerta, allora ci si aspetta che il banditore A comunichi a B o una vincita oppure una perdita. La comunicazione del risultato deve avvenire all'istante $TAnswer$ successivo a TClose ed entro DtAnswer unità di tempo. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti offerte alle quali non sia stata comunicato un risultato.

Tramite il vincolo di integrità (6.4.9) il protocollo impone consistenza fra offerta e risposta. In dettaglio tale $\mathcal{IC}_S$ afferma che se un offerente B ha offerto Q per l'articolo $Item$, se era stato comunicato il prezzo di riserva $ReservePrice$ e se $Q < ReservePrice$, allora ci si aspetta che a B sia comunicata una perdita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti comunicazioni diverse dalla perdita nel caso un'offerente offra meno del prezzo di riserva.

Tramite il vincolo di integrità (6.4.10) il protocollo impone consistenza

$$
\begin{aligned}
&\mathbf{H}(tell(A, Bidders, closeauction(Item, BasePrice, DtEnd,\\
&DtAnswer), AuctionID), TClose)\\
&\wedge\\
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid)\\
&\longrightarrow\\
&\mathbf{E}(tell(A, B, answer(win, Item, Q), AuctionID), TAnswer)\\
&\wedge\ TClose < TAnswer\\
&\wedge\ TAnswer \leq TClose + DtAnswer\\
&\vee\\
&\mathbf{E}(tell(A, B, answer(lose, Item, Q), AuctionID), TAnswer)\\
&\wedge\ TClose < TAnswer\\
&\wedge\ TAnswer \leq TClose + DtAnswer.
\end{aligned}
\tag{6.4.8}
$$

$$
\begin{aligned}
&\mathbf{H}(tell(B, A, bid(Item, Q), AuctionID), TBid)\\
&\wedge\\
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice),\\
&AuctionID), TReservedInfo)\\
&\wedge\ Q < ReservePrice\\
&\longrightarrow\\
&\mathbf{E}(tell(A, B, answer(lose, Item, Q), AuctionID), TAnswer).
\end{aligned}
\tag{6.4.9}
$$

fra offerta e risposta. In dettaglio tale $\mathcal{IC}_S$ afferma che se un offerente $B1$ ha offerto $Q1$ per l'articolo $Item$, se era stato comunicato il prezzo di riserva $ReservePrice$ e se $Q1 \geq ReservePrice$, allora ci si aspetta che a un $B2$ sia comunicata una vincita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti comunicazioni di sola perdita nel caso almeno un'offerente abbia offerto più del prezzo di riserva.

$$\begin{aligned}
&\mathbf{H}(tell(B1, A, bid(Item, Q1), AuctionID), TBid) \\
&\wedge \\
&\mathbf{H}(tell(A, A, reservedinfo(Item, BasePrice, ReservePrice), \\
&AuctionID), TReservedInfo) \\
&\wedge\ Q1 \geq ReservePrice \\
&\longrightarrow \\
&\mathbf{E}(tell(A, B2, answer(win, Item, Q2), AuctionID), TAnswer).
\end{aligned} \tag{6.4.10}$$

Ricordiamo che nell'asta Inglese è stato possibile realizzare un meccanismo basato su vincoli CLP, applicati a variabili temporali, che permetteva di identificare se un'offerta fosse la prima, l'ultima oppure una intermedia. In tal modo si riusciva ad identificare l'ultimo offerente che per definizione era il vincitore dell'asta.

Nell'asta FPSB occorre identificare il vincitore attraverso un meccanismo differente da quello basato su *finestre temporali*. Infatti il vincitore è l'offerente che offre la quota più alta, piazzata in un *qualsiasi* istante valido. D'altro canto sebbene il vincolo (6.4.10) sia corretto identifica i potenziali vincitori e non quello che ha offerto il massimo. Per determinare fra questi *il* vincitore utilizziamo il vincolo (6.4.11) che sfrutta un meccanismo basato sull'*esclusione*.

Tramite il vincolo di integrità (6.4.11) il protocollo impone la comunicazione di vincita all'offerente che ha versato la quota maggiore. In dettaglio tale $\mathcal{IC}_S$ afferma che se all'offerente $B1$, che ha offerto $Q1$ per l'articolo $Item$,

è stata comunicata una vincita, allora ci si aspetta che non sia stata piazzata alcuna offerta $Q2$ superiore a $Q1$ da un altro offerente $B2$. In altre parole tale $\mathcal{IC}_S$ evita che sia comunicata una vincita a chi ha non ha offerto il massimo.

$$\begin{aligned}
&\mathbf{H}(tell(A, B1, answer(win, Item, Q1), AuctionID), TAnswer) \\
&\longrightarrow \\
&\mathbf{EN}(tell(B2, A, bid(Item, Q2), AuctionID), TBid) \\
&\wedge\ Q2 \geq Q1 \\
&\wedge\ B2! = B1.
\end{aligned} \tag{6.4.11}$$

Tramite il vincolo di integrità (6.4.12) il protocollo impone consistenza fra risposta ed offerta. In dettaglio tale $\mathcal{IC}_S$ afferma che se un banditore A nell'asta $AuctionID$ ha comunicato una risposta all'offerente B, il quale ha offerto Q per l'articolo $Item$, allora ci si aspetta che B abbia piazzato un'offerta presso il banditore A per l'articolo $Item$ offrendo Q all'interno dell'asta $AuctionID$. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti messaggi di risposta riferiti ad offerte mai piazzate.

$$\begin{aligned}
&\mathbf{H}(tell(A, B, answer(Result, Item, Q), AuctionID), TAnswer) \\
&\longrightarrow \\
&\mathbf{E}(tell(B, A, bid(Item, Q), AuctionID), TBid).
\end{aligned} \tag{6.4.12}$$

Tramite il vincolo di integrità (6.4.13) il protocollo impone l'unicità del messaggio di vincita. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente $B1$ è stata comunicata una vincita all'interno dell'asta $AuctionID$, allora ci si aspetta che non sia comunicata una vincita anche all'offerente $B2$ che ha partecipato alla stessa asta. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti più messaggi di vincita per l'articolo $Item$ all'interno della stessa asta.

$$\begin{aligned}
&\mathbf{H}(tell(A, B1, answer(win, Item, QWin1), AuctionID),\\
&TAnswerWin1)\\
&\longrightarrow\\
&\mathbf{EN}(tell(A, B2, answer(win, Item, QWin2), AuctionID),\\
&TAnswerWin2)\\
&\wedge\ B1! = B2.
\end{aligned} \tag{6.4.13}$$

Tramite il vincolo di integrità (6.4.14) il protocollo impone di rispondere in modo non contraddittorio all'offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una vincita, allora ci si aspetta che non gli sia comunicata una perdita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di vincita che di perdita relativi ad uno stesso offerente.

$$\begin{aligned}
&\mathbf{H}(tell(A, B, answer(win, Item, QWin), AuctionID),\\
&TAnswerWin)\\
&\longrightarrow\\
&\mathbf{EN}(tell(A, B, answer(lose, Item, QLose), AuctionID),\\
&TAnswerLose).
\end{aligned} \tag{6.4.14}$$

Tramite il vincolo di integrità (6.4.15) il protocollo impone di rispondere in modo non contraddittorio all'offerente. In dettaglio tale $\mathcal{IC}_S$ afferma che se ad un offerente è stata comunicata una perdita, allora ci si aspetta che non gli sia comunicata una vincita. In altre parole tale $\mathcal{IC}_S$ evita che nella history siano presenti contemporaneamente messaggi sia di perdita che di vincita relativi ad uno stesso offerente.

$$\begin{aligned}&\mathbf{H}(tell(A, B, answer(lose, Item, QLose), AuctionID),\\&TAnswerLose)\\&\longrightarrow\\&\mathbf{EN}(tell(A, B, answer(win, Item, QWin), AuctionID),\\&TAnswerWin).\end{aligned} \tag{6.4.15}$$

6.5 Verifica di conformità con $\mathcal{S}$CIFF

Per verificare se una history è conforme[4] al protocollo abbiamo proceduto secondo quanto descritto nel paragrafo 3.2.

A titolo di esempio riportiamo la seguente history:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 9, 100, 50], 0).

tell([s0], auction1, b1, auctioneer, bid, [[art1], 19], 12).
tell([s0], auction1, b2, auctioneer, bid, [[art1], 99], 34).
tell([s0], auction1, b3, auctioneer, bid, [[art1], 39], 56).

tell([s0], auction1, auctioneer, auctioneer, reservedinfo,
[[art1], 9, 39], 100).
tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 9, 100, 50], 100).

tell([s0], auction1, auctioneer, b1, answer,
[ lose, [art1], 19], 130).
tell([s0], auction1, auctioneer, b2, answer,
[ win,  [art1], 99], 140).
tell([s0], auction1, auctioneer, b3, answer,
[ lose, [art1], 39], 150).
```

In sintesi la simulazione condotta con $\mathcal{S}$CIFF ha prodotto i seguenti risultati:

[4] Vedi paragrafo 3.2 definizione di history *compliant*.

```
| ?- run.
...
fulf(e(tell(auctioneer,bidders,openauction([art1],9,100,50),
auction1),0)), ...
fulf(en(tell(_C1,_D1,openauction(_A1,_B1,_Z,_Y),auction1),_A)),
...
h(tell(auctioneer,bidders,openauction([art1],9,100,50),
auction1),0), ...
h(tell(auctioneer,b3,answer(lose,[art1],39),auction1),150),
... ? ;
no
| ?-
```

In cui notiamo le aspettative sociali fulfilled[5] contrassegnate da `fulf`. Per esempio con il primo `fulf` abbiamo la verifica dell'aspettativa sociale descritta attraverso l'$\mathcal{IC}_S$ (6.4.3), con il secondo `fulf` abbiamo la verifica dell'aspettativa descritta nel vincolo (6.4.2), e così via. Nelle altre righe osserviamo anche le transizioni `h` attraverso le quali $\mathcal{S}$CIFF diviene consapevole degli eventi della history. Infine, tramite il comando ";" abbiamo richiesto alla proof di proseguire la simulazione, ottenendo il termine dell'esecuzione in quanto non vi sono più eventi nella history da analizzare.

Abbiamo sottoposto a $\mathcal{S}$CIFF anche un history non conforme al protocollo, identica alla precedente tranne per l'assenza della comunicazione `reservedinfo` il cui scopo è quello di esplicitare il prezzo di riserva:

```
tell([s0], auction1, auctioneer, bidders, openauction,
[[art1], 9, 100, 50], 0).

tell([s0], auction1, b1, auctioneer, bid, [[art1], 19], 12).
tell([s0], auction1, b2, auctioneer, bid, [[art1], 99], 34).
tell([s0], auction1, b3, auctioneer, bid, [[art1], 39], 56).

tell([s0], auction1, auctioneer, bidders, closeauction,
[[art1], 9, 100, 50], 100).
```

[5] Vedi paragrafo 2.3 definizione di *fulfillment*.

```
tell([s0], auction1, auctioneer, b1, answer,
[ lose, [art1], 19], 130).
tell([s0], auction1, auctioneer, b2, answer,
[ win,  [art1], 99], 140).
tell([s0], auction1, auctioneer, b3, answer,
```

l'esito della simulazione è:

```
| ?- run.
no
| ?-
```

Ossia $\mathcal{S}$CIFF ha risposto dicendo che la history non è conforme al protocollo.

6.6 Verifica di proprietà

La verifica delle proprietà è stata condotta secondo la metodologia esposta nel paragrafo 3.3. Gli esiti delle risposte sono stati salvati in file di testo, in alcuni casi, di notevole dimensione, perciò ne esporremo una sintesi[6] caratterizzata dagli aspetti di maggior interesse per questa tesi. Per completezza riporteremo in appendice alcune brevi risposte in "versione integrale".

In questo paragrafo analizzeremo alcune risposte fornite da g$\mathcal{S}$CIFF, cercando di interpretare i risultati ottenuti.

6.6.1 History di controesempio

Abbiamo voluto verificare la proprietà "se è stata aperta un'asta, allora ci si aspettano eventuali offerte":

$$\begin{array}{l} tell(A, Bidders, openauction(Item, BasePrice, DtEnd, \\ DtAnswer), AuctionID) \\ \longrightarrow \\ tell(B, A, bid(Item, Q), AuctionID). \end{array} \tag{6.6.1}$$

[6] Le omissioni sono indicate con "...".

La negazione è:

$$openauction \wedge \neg\, bid \tag{6.6.2}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( A, Bidders, openauction( Item, BasePrice, DtEnd, DtAnswer),
AuctionID), TOpen),
en( tell( B, A, bid( Item, Q), AuctionID), TBid).
```

In sintesi il risultato è stato il seguente:

```
| ?- run.
...
h(tell(_E,_J,openauction(_D,_I,_H,_G),_F),_A), ...
h(tell(_E,_J,closeauction(_D,_I,_H,_G),_F),_K), ...
_A + _H = _K,
_A >= 1,
_G >= 1,
_H >= 1,
_I >= 1,
_K >= 2,
_A \= _B,
_A \= _B,
_A \= _C,
_A \= _C ? ;
...
no
| ?-
```

La risposta fornita dalla proof è stata una history caratterizzata dagli eventi *openauction* e *closeauction*. La richiesta di un successivo calcolo ha dato esito negativo facendo terminare l'esecuzione.

g$\mathcal{S}$CIFF ha risposto "yes" e la history calcolata rappresenta un controesempio che prova come il protocollo non gode della proprietà (6.6.1). In altre parole il protocollo non gode della proprietà poiché, nel caso un offerente apra e chiuda l'asta non è necessario che un offerente piazzi un'offerta.

Di interesse sono i vincoli CLP relativi alle variabili `_H`, `_A` e `_K` rappresentanti rispettivamente la durata dell'asta, l'istante di apertura e l'istante di chiusura dell'asta, ossia le variabili *DtEnd*, *TOpen* e *TClose* dell'g$\mathcal{IC}_S$ (6.4.1). Si osserva che il vincolo CLP $TClose == TOpen + DtEnd$ è correttamente tradotto come `_A + _H = _K` e poiché `_H` e `_A` devono essere maggiori di 1 allora `_K` sarà necessariamente maggiore di 2.

Osserviamo infine che g$\mathcal{S}$CIFF avrebbe potuto fornire come controesempio anche una history vuota, come accaduto in 4.6.1. Il risultato calcolato è diverso in quanto, sebbene g$\mathcal{S}$CIFF operi sempre allo stesso modo, il suo risultato dipende, come illustrato in figura 3.2, anche dal tipo di protocollo, dalla $SOKB$ e dal goal.

6.6.2 Probabile successo

Alla ricerca di una proprietà posseduta dal protocollo abbiamo verificato la proprietà "se un'asta è stata aperta allora deve essere stata chiusa":

$$\begin{aligned} &tell(A, Bidders, openauction(Item, BasePrice, DtEnd,\\ &DtAnswer), AuctionID),\\ &\longrightarrow \\ &tell(A, Bidders, closeauction(Item, BasePrice, DtEnd,\\ &DtAnswer), AuctionID). \end{aligned} \tag{6.6.3}$$

La negazione è:

$$openauction \wedge \neg\, closeauction. \tag{6.6.4}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( A, Bidders, openauction( Item, BasePrice, DtEnd,
DtAnswer), AuctionID), TOpen),
en( tell( A, Bidders, closeauction( Item, BasePrice, DtEnd,
DtAnswer), AuctionID), TClose).
```

In sintesi il risultato è stato il seguente:

```
| ?- run.
...
no
| ?-
```

La risposta fornita g$\mathcal{S}$CIFF è "no". Purtroppo, data la non completezza[7] della proof, tale risultato è necessario ma non sufficiente a garantire che il protocollo goda della proprietà (6.6.3).

6.6.3 Constraint overflow

Abbiamo voluto verificare la proprietà "se è stata piazzata un'offerta allora ci si aspetta che qualcuno vinca":

$$\begin{aligned} & tell(B1, A, bid(Item, Q1), AuctionID) \\ & \longrightarrow \\ & tell(A, B2, answer(win, Item, Q2), AuctionID). \end{aligned} \tag{6.6.5}$$

La negazione è:

$$bid \wedge \neg\, answer \tag{6.6.6}$$

pertanto il goal è stato scritto come:

```
society_goal:-
e( tell( B1, A, bid( Item, Q1), AuctionID), TBid),
en( tell( A, B2, answer( win, Item, Q2), AuctionID),
TAnswer).
```

In sintesi il risultato è stato il seguente:

```
| ?- run.
==Log== ...
==Log== ...
==Log== ...
```

[7] Vedi paragrafo 3.3 *correttezza* e *non completezza* di g$\mathcal{S}$CIFF.

```
...
! Representation error in user:'t>=u+c'/3
! CLPFD integer overflow
! goal:  't>=u+c'(_9752,_9221,1)
| ?-
```

La proof interrompe in modo anomalo il calcolo della verifica a causa di un errore di overflow nella rappresentazione degli interi. La causa di tale errore è già stata analizzata in dettaglio nella sezione 4.6.4.

Poiché la proof non ha fornito alcun tipo di risposta è impossibile stabilire se il protocollo goda della proprietà oggetto di verifica.

Conclusioni

In questa tesi i risultati conseguiti sono da interpretarsi nel contesto del progetto SOCS [1] a cui ci siamo costantemente riferiti. I risultati ottenuti sono stati:

- Un progetto corretto ed efficace di tre protocolli di interazione fra agenti operanti rispettivamente negli scenari d'asta Inglese, d'asta Combinatoria e d'asta First Price Sealed Bid.
- La corretta verifica sull'interagire degli agenti in conformità, o meno, ai protocolli precedentemente definiti.
- La parziale verifica di proprietà godute, o meno, dai protocolli precedentemente definiti.

Progetto di protocolli

Il progetto dei protocolli è stato effettuato adottando un *approccio sociale* agli Interaction Protocols (IP). Con questa espressione intendiamo che la definizione delle specifiche di interazione è stata realizzata attraverso vincoli di integrità sociale su eventi sociali.

Con un'opportuna rappresentazione della conoscenza sociale abbiamo definito ogni vincolo di integrità come una relazione fra *eventi* accaduti nella società ed *aspettative* sul comportamento esternamente osservabile degli agenti. L'insieme di più vincoli ha permesso di definire protocolli come *regole in avanti* che mettono in relazione gli eventi alle aspettative.

L'approccio sociale agli IP si è dimostrato una tecnica per la progettazione di protocolli molto potente ed espressiva. Ha permesso non solo di definire un'interazione efficiente fra gli agenti ma, in alcuni casi come l'asta Inglese e First Price Sealed Bid, si è dimostrato uno strumento maturo per stabilire anche il vincitore dell'asta. Solitamente il calcolo del vincitore è delegato a moduli esterni, altamente efficienti, implementati in linguaggi imperativi.

Verifiche di conformità

Le verifiche di conformità del comportamento degli agenti agli IP, secondo un'opportuna semantica dichiarativa, sono state ottenute con un metodo di *ragionamento abduttivo* implementato dalla *proof procedure* $\mathcal{S}CIFF$. Le potenzialità di questo approccio risiedono nella possibilità di eseguire i controlli di conformità "on–the–fly", ossia durante la attività degli agenti nella società, e non in una separata sede di simulazione.

I test che abbiamo effettuato sono stati condotti sottoponendo a $\mathcal{S}CIFF$ un insieme di eventi, chiamato history, rappresentativo di una tipica situazione d'asta caratterizzata da messaggi fra agenti che vendono e comprano beni.

Per ogni protocollo abbiamo dimostrato con successo come $\mathcal{S}CIFF$ giudichi conformi al protocollo le history corrette, e non conformi quelle non corrette. Per history corrette intendiamo quelle in cui *tutti* gli agenti operano nel rispetto di *tutte* le specifiche descritte nell'IP, mentre per history non corrette intendiamo quelle in cui *almeno* un agente violi *almeno* una di tali specifiche.

Verifiche di proprietà

Le proprietà, che si desidera che un protocollo possegga, sono state definite attraverso vincoli di integrità sociale. Ciò ha permesso di condurre le verifiche di proprietà riutilizzando $\mathcal{S}$CIFF. Più in dettaglio abbiamo utilizzato

g$\mathcal{S}$CIFF, una variante della proof dotata di un *generatore di history*: dotata cioè della capacità di generare history conformi al protocollo.

I test condotti su una vasta gamma di proprietà hanno dato risultati parzialmente soddisfacenti: in primo luogo per un limite intrinseco di g$\mathcal{S}$CIFF, che è una proof *corretta* ma non *completa*, in secondo luogo per la natura *ciclica* dei protocolli.

In virtù della *correttezza* di g$\mathcal{S}$CIFF la verifica di una proprietà, con esito negativo, ha permesso di affermare con certezza che il protocollo non gode della proprietà. Purtroppo a causa della *non completezza* di g$\mathcal{S}$CIFF la verifica di una proprietà, con esito positivo, è una condizione necessaria ma non sufficiente per affermare che il protocollo goda della proprietà.

Un secondo motivo di insoddisfazione è rappresentato dalla natura *ciclica* dei protocolli. Con questa espressione indichiamo situazioni in cui gli eventi accaduti nella società possano coincidere, attraverso propagazione in avanti dei vincoli di integrità, con le aspettative della società stessa. Se da un lato tali coincidenze siano state deliberatamente introdotte per conferire stabilità e robustezza al protocollo, d'altro canto hanno avuto l'effetto collaterale di obbligare g$\mathcal{S}$CIFF in loop, apparentemente, infiniti o di causarne una interruzione anomala. In questi casi, poiché la proof non ha fornito alcuna risposta, non è possibile affermare se il protocollo goda, o meno, della proprietà oggetto di verifica.

Possibili sviluppi futuri

Possibili sviluppi futuri potrebbero riguardare i protocolli progettati e la proof procedure g$\mathcal{S}$CIFF.

In particolare ogni protocollo potrebbe essere ampliato in modo da supportare la *rieducazione* degli agenti ad un corretto comportamento sociale e la *sanzione* di ripetuti atteggiamenti scorretti. Inoltre sarebbe desiderabile che il protocollo, appena verificato il fallimento di atti comunicativi, riportasse la comunicazione ad uno stato consistente. Ciò potrebbe realizzarsi at-

traverso opportuni messaggi standardizzati dalla società, oppure delegando il "recovery" a moduli esterni.

Relativamente a g$\mathcal{S}$CIFF potrebbe essere migliorato il generatore di history, in modo da evitare il verificarsi di loop apparentemente infiniti, ed interruzioni anomale per overflow. Allo stato attuale, infatti, non esiste una strategia per evitare tali situazioni ed in aiuto possiamo dare solo alcune linee guida. Per esempio progettare un protocollo pensando già in termini di quali proprietà si desidererebbe che esso possegga, oppure progettare le specifiche di interazione nel modo più semplice possibile, cercando un compromesso con esigenze che tenderebbero ad aumentarne la complessità. In tale contesto un motore di generatore di history migliore fornisce molti aspetti da approfondire in futuro.

Ringraziamenti

Ringrazio la Prof. Ing. E. Lamma per avermi offerto questa possibilità e per la disponibilità che mi ha sempre dimostrato. Ringrazio il Dott. Ing. M. Alberti per la disponibilità e l'attenzione dedicatemi in ogni passo di questo lavoro.

Se da un lato mi è sembrato doveroso ringraziare pubblicamente chi ha contribuito col proprio lavoro a questa tesi, dall'altro ho sentito l'esigenza di proteggere la mia privacy omettendo i ringraziamenti a familiari, amici e colleghi da questa *versione on-line* della tesi. I sentiti ringraziamenti a familiari, amici e colleghi sono invece presenti nella *versione integrale* della tesi.

Bibliografia

[1] Societies Of ComputeeS (SOCS): a computational logic model for the description, analysis and verification of global and open societies of heterogeneous computees. http://lia.deis.unibo.it/Research/SOCS/.

[2] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. An Abductive Interpretation for Open Societies. In A. Cappelli and F. Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence, Proceedings of the 8th Congress of the Italian Association for Artificial Intelligence, Pisa*, volume 2829 of *Lecture Notes in Artificial Intelligence*, pages 287–299. Springer-Verlag, September 23–26 2003. http://www-aiia2003.di.unipi.it.

[3] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Compliance verification of agent interaction: a logic-based tool. pages 570–575, Vienna, Austria, April 13–16 2004. Austrian Society for Cybernetic Studies.

[4] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. The SOCS computational logic approach for the specification and verification of agent societies. In *Global Computing Workshop, Rovereto, Italy, March 2004*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004. to appear.

[5] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent interaction. In P. Ciancarini and M. Wooldridge,

editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer-Verlag, 2001.

[6] P. T. Cox and T. Pietrzykowski. Causes for events: Their computation and applications. In *Proceedings CADE–86*, pages 608–621, 1986.

[7] P. Davidsson. Categories of artificial societies. In A. Omicini, P. Petta, and R. Tolksdorf, editors, *Engineering Societies in the Agents World II*, volume 2203 of *Lecture Notes in Artificial Intelligence*, pages 1–9. Springer-Verlag, December 2001. http://link.springer.de/link/service/-series/0558/papers/2203/% -22030001.pdf.

[8] V. Dignum, J. J. Meyer, H. Weigand, and F. Dignum. An organizational-oriented model for agent societies. In *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications. AAMAS'02, Bologna*, 2002.

[9] *FIPA Interaction Protocol Library Specification (XC00025E)*, 2001. Available for download from the FIPA website, http://www.fipa.org.

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

[11] 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.

[12] C. Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47(1–3):79–106, 1991.

[13] ILOG S.A., France. *ILOG Solver*, 5.0 edition, 2003.

[14] M. Ito and J.S. Sichman. Dependence based coalitions and contract net: A comparative analysis. In *Pacific Rim International Conference on Artificial Intelligence*, page 812, 2000. http://citeseer.nj.nec.com/-442119.html.

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

[16] A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. Oxford University Press, 1998.

[17] A. C. Kakas and P. Mancarella. On the relation between Truth Maintenance and Abduction. In T. Fukumura, editor, *Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence, PRICAI–90, Nagoya, Japan*, pages 438–443. Ohmsha Ltd., 1990.

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

[19] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd extended edition, 1987.

[20] H. V. Parunak. Visualizing agent conversations: using enhanced dooley graphs for agent design and analysis. In *Proceedings of the 2nd International Conference on Multiagent Systems, San Francisco, California*, pages 275–282. AAAI Press, 1996.

[21] A. Rao and M. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of the International Workshop on Knowledge Representation, KR'92*, pages 439–449, 1992.

[22] M. Singh. Agent communication language: rethinking the principles. *IEEE Computer*, pages 40–47, December 1998.

[23] R.G. Smith. The contract net protocol: High level communication and control in a distributed porblem solver. *IEEE Transaction on Computers*, C–29(12):1104–1113, 1980.

[24] M. P. Wellman. A market–oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993. http:// citeseer.nj.nec.com /wellman93marketoriented.html.

[25] M. P. Wellman and P. R. Wurman. Market–aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998. http://citeseer.nj.nec.com/wellman97marketaware.html.

[26] Yoav Shoham (Written with Trond Grenager). *Introduction to Multi--Agent Systems*, chapter 7, pages 161–167, 171–172. April, 30 2002. http://www.stanford.edu/class/cs206/ch07.pdf.

[27] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[28] P. Yolum and M.P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS–2002), Part II*, pages 527–534, Bologna, Italy, July 15–19 2002. ACM Press. http://portal.acm.org/ ft_gateway-.cfm ?id=544867 &type=pdf &dl=GUIDE &dl=ACM &CFID=4415868 &CFTOKEN=57395936.

Appendice A

Tabella dei simboli

Simbolo	Significato
H	Evento accaduto.
E	Evento che ci si aspetta debba accadere.
EN	Evento che ci si aspetta debba non accadere.
$\longrightarrow$	Operatore di implicazione logica.
$\wedge$	Operatore di and logico.
$\vee$	Operatore di or logico.
$\neg$	Negazione Logica.
$\perp$	Insieme vuoto.
$\models$	Derivazione.
$\tilde{\models}_{\mathbf{EXP}}$	Raggiungibilità di un goal.
$\models_{\mathbf{EXP}}$	Raggiungimento di un goal.
==	Uguaglianza letterale fra termini.
<>	Disuguaglianza letterale fra termini.
!=	Non unificabilità fra termini.

Tabella A.1: Tabella dei simboli.

Appendice B

Verifiche di conformità

B.1 History compliant

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 4.5 relativa alla conformità di una history al protocollo dell'asta Inglese.

```
| ?- run.
forallf(_A),
forallf(_B),
close_history,
current_time(0),



fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
fulf(e(tell(b1,auctioneer,bid([art1],19),auction1),100)),
fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
fulf(e(tell(b2,auctioneer,bid([art1],29),auction1),200)),
fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
fulf(e(tell(b3,auctioneer,bid([art1],39),auction1),300)),
fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
fulf(e(tell(b4,auctioneer,bid([art1],49),auction1),400)),
fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
fulf(e(tell(b4,auctioneer,bid([art1],49),auction1),400)),
fulf(e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500)),
fulf(e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500)),
fulf(e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500)),
fulf(e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500)),
fulf(e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0)),
```

```
fulf(e(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),500
)),
fulf(e(tell(b1,auctioneer,bid([art1],19),auction1),100)),
fulf(e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),550)),
fulf(e(tell(b2,auctioneer,bid([art1],29),auction1),200)),
fulf(e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),550)),
fulf(e(tell(b3,auctioneer,bid([art1],39),auction1),300)),
fulf(e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),550)),
fulf(e(tell(b4,auctioneer,bid([art1],49),auction1),400)),
fulf(e(tell(auctioneer,b4,answer(win,[art1],49),auction1),550)),
fulf(en(tell(auctioneer,b4,answer(lose,[art1],_D),auction1),_C)),
fulf(en(tell(auctioneer,b3,answer(win,[art1],_F),auction1),_E)),
fulf(en(tell(auctioneer,b2,answer(win,[art1],_H),auction1),_G)),
fulf(en(tell(auctioneer,b1,answer(win,[art1],_J),auction1),_I)),
fulf(en(tell(_P,_Q,openauction(_N,_O,_M,_L,_K),auction1),_A)),



ic([h(tell(_X,_Y,closeauction(_V,_W,_U,_T,_S),_R),_A1)],[[_W>0,_U>0,_T>0,_S>0,
e(tell(_X,_Y,openauction(_V,_W,_U,_T,_S),_R),_Z)]]),
psic([[h(tell(_X,_Y,closeauction(_V,_W,_U,_T,_S),_R),_A1)],[],[],[],[],[],[]],
[[st(_W#>0),st(_U#>0),st(_T#>0),st(_S#>0),e(tell(_X,_Y,openauction(_V,_W,_U,_T
,_S),_R),_Z)]]),
ic([h(tell(_H1,_I1,openauction(_F1,_G1,_E1,_D1,_C1),_B1),_R1)],[[<>(_R1,_Q1),e
n(tell(_O1,_P1,openauction(_J1,_K1,_L1,_M1,_N1),_B1),_Q1)]]),
psic([[h(tell(_H1,_I1,openauction(_F1,_G1,_E1,_D1,_C1),_B1),_R1)],[],[],[],[],
[],[]],[[st(_R1#\=_Q1),en(tell(_O1,_P1,openauction(_J1,_K1,_L1,_M1,_N1),_B1),_
Q1)]]),
ic([h(tell(_Y1,_Z1,openauction(_W1,_X1,_V1,_U1,_T1),_S1),_E2)],[[_E2<_D2,_D2=<
_E2+_U1,_C2>=_X1,e(tell(_B2,_Y1,bid(_W1,_C2),_S1),_D2)],[e(tell(_Y1,_Z1,closea
uction(_W1,_X1,_V1,_U1,_T1),_S1),_A2),_A2=_E2+_U1]]),
psic([[h(tell(_Y1,_Z1,openauction(_W1,_X1,_V1,_U1,_T1),_S1),_E2)],[],[],[],[],
[],[]],[[_E2#<_D2,_D2#=<_E2+_U1,_C2#>=_X1,e(tell(_B2,_Y1,bid(_W1,_C2),_S1),_D2
)],[_A2#=_E2+_U1,e(tell(_Y1,_Z1,closeauction(_W1,_X1,_V1,_U1,_T1),_S1),_A2)]])
,
ic([h(tell(_I2,_J2,bid(_G2,_H2),_F2),_Q2)],[[e(tell(_J2,_O2,openauction(_G2,_K
2,_L2,_M2,_N2),_F2),_P2)]]),
psic([[h(tell(_I2,_J2,bid(_G2,_H2),_F2),_Q2)],[],[],[],[],[],[]],[[e(tell(_J2,
_O2,openauction(_G2,_K2,_L2,_M2,_N2),_F2),_P2)]]),
ic([h(tell(_X2,_Y2,openauction(_V2,_W2,_U2,_T2,_S2),_R2),_G3),h(tell(_E3,_X2,b
id(_V2,_D3),_R2),_F3)],[[_F3<_C3,_C3=<_F3+_T2,_B3>=_D3+_U2,e(tell(_A3,_X2,bid(
_V2,_B3),_R2),_C3)],[e(tell(_X2,_Y2,closeauction(_V2,_W2,_U2,_T2,_S2),_R2),_Z2
),_Z2=_F3+_T2]]),
psic([[h(tell(_X2,_Y2,openauction(_V2,_W2,_U2,_T2,_S2),_R2),_G3),h(tell(_E3,_X
2,bid(_V2,_D3),_R2),_F3)],[],[],[],[],[],[]],[[_F3#<_C3,_C3#=<_F3+_T2,_B3#>=_D
3+_U2,e(tell(_A3,_X2,bid(_V2,_B3),_R2),_C3)],[_Z2#=_F3+_T2,e(tell(_X2,_Y2,clos
eauction(_V2,_W2,_U2,_T2,_S2),_R2),_Z2)]]),
```

```
ic([h(tell(_N3,_O3,openauction(_L3,_M3,_K3,_J3,_I3),_H3),_V3),h(tell(_T3,_N3,b
id(_L3,_S3),_H3),_U3),h(tell(_N3,_O3,closeauction(_L3,_M3,_K3,_J3,_I3),_H3),_R
3)],[[_Q3>=_M3,_V3<_P3,_P3=<_R3,e(tell(_N3,_N3,reservedinfo(_L3,_M3,_Q3),_H3),
_P3)]]),
psic([[h(tell(_N3,_O3,openauction(_L3,_M3,_K3,_J3,_I3),_H3),_V3),h(tell(_T3,_N
3,bid(_L3,_S3),_H3),_U3),h(tell(_N3,_O3,closeauction(_L3,_M3,_K3,_J3,_I3),_H3)
,_R3)],[],[],[],[],[],[]],[[_Q3#>=_M3,_V3#<_P3,_P3#=<_R3,e(tell(_N3,_N3,reserv
edinfo(_L3,_M3,_Q3),_H3),_P3)]]),
ic([h(tell(_A4,_A4,reservedinfo(_Y3,_Z3,_X3),_W3),_J4)],[[e(tell(_A4,_H4,opena
uction(_Y3,_Z3,_E4,_F4,_G4),_W3),_I4),e(tell(_C4,_A4,bid(_Y3,_B4),_W3),_D4)]])
,
psic([[h(tell(_A4,_A4,reservedinfo(_Y3,_Z3,_X3),_W3),_J4)],[],[],[],[],[],[]],
[[e(tell(_A4,_H4,openauction(_Y3,_Z3,_E4,_F4,_G4),_W3),_I4),e(tell(_C4,_A4,bid
(_Y3,_B4),_W3),_D4)]]),
ic([h(tell(_N4,_O4,bid(_L4,_M4),_K4),_Y4),h(tell(_O4,_O4,reservedinfo(_L4,_V4,
_W4),_K4),_X4),h(tell(_O4,_T4,closeauction(_L4,_V4,_Q4,_R4,_S4),_K4),_U4),_U4=
_Y4+_R4,_M4>=_W4],[[_U4<_P4,_P4=<_U4+_S4,e(tell(_O4,_N4,answer(win,_L4,_M4),_K
4),_P4)]]),
psic([[h(tell(_N4,_O4,bid(_L4,_M4),_K4),_Y4),h(tell(_O4,_O4,reservedinfo(_L4,_
V4,_W4),_K4),_X4),h(tell(_O4,_T4,closeauction(_L4,_V4,_Q4,_R4,_S4),_K4),_U4)],
[],[],[],[],[],[st(_U4#=_Y4+_R4),st(_M4#>=_W4)]],[[_U4#<_P4,_P4#=<_U4+_S4,e(te
ll(_O4,_N4,answer(win,_L4,_M4),_K4),_P4)]]),
ic([h(tell(_C5,_D5,bid(_A5,_B5),_Z4),_N5),h(tell(_D5,_D5,reservedinfo(_A5,_K5,
_L5),_Z4),_M5),h(tell(_D5,_I5,closeauction(_A5,_K5,_F5,_G5,_H5),_Z4),_J5),_J5=
_N5+_G5,_B5<_L5],[[_J5<_E5,_E5=<_J5+_H5,e(tell(_D5,_C5,answer(lose,_A5,_B5),_Z
4),_E5)]]),
psic([[h(tell(_C5,_D5,bid(_A5,_B5),_Z4),_N5),h(tell(_D5,_D5,reservedinfo(_A5,_
K5,_L5),_Z4),_M5),h(tell(_D5,_I5,closeauction(_A5,_K5,_F5,_G5,_H5),_Z4),_J5)],
[],[],[],[],[],[st(_J5#=_N5+_G5),st(_B5#<_L5)]],[[_J5#<_E5,_E5#=<_J5+_H5,e(tel
l(_D5,_C5,answer(lose,_A5,_B5),_Z4),_E5)]]),
ic([h(tell(_R5,_S5,bid(_P5,_Q5),_O5),_D6),h(tell(_B6,_S5,bid(_P5,_A6),_O5),_C6
),h(tell(_S5,_Y5,closeauction(_P5,_U5,_V5,_W5,_X5),_O5),_Z5),_Z5=_C6+_W5,not_u
nif(_R5,_B6)],[[_Z5<_T5,_T5=<_Z5+_X5,e(tell(_S5,_R5,answer(lose,_P5,_Q5),_O5),
_T5)]]),
psic([[h(tell(_R5,_S5,bid(_P5,_Q5),_O5),_D6),h(tell(_B6,_S5,bid(_P5,_A6),_O5),
_C6),h(tell(_S5,_Y5,closeauction(_P5,_U5,_V5,_W5,_X5),_O5),_Z5)],[],[],[],[],[
],[st(_Z5#=_C6+_W5),st(reif_unify(...))]],[[_Z5#<_T5,_T5#=<_Z5+_X5,e(tell(_S5,
_R5,answer(lose,_P5,_Q5),_O5),_T5)]]),
ic([h(tell(_H6,_I6,answer(win,_G6,_F6),_E6),_L6)],[[en(tell(_H6,_I6,answer(los
e,_G6,_J6),_E6),_K6)]]),
psic([[h(tell(_H6,_I6,answer(win,_G6,_F6),_E6),_L6)],[],[],[],[],[],[]],[[en(t
ell(_H6,_I6,answer(lose,_G6,_J6),_E6),_K6)]]),
ic([h(tell(_P6,_Q6,answer(lose,_O6,_N6),_M6),_T6)],[[en(tell(_P6,_Q6,answer(wi
n,_O6,_R6),_M6),_S6)]]),
psic([[h(tell(_P6,_Q6,answer(lose,_O6,_N6),_M6),_T6)],[],[],[],[],[],[]],[[en(
tell(_P6,_Q6,answer(win,_O6,_R6),_M6),_S6)]]),
ic([h(tell(_Y6,_Z6,answer(_W6,_X6,_V6),_U6),_B7)],[[e(tell(_Z6,_Y6,bid(_X6,_V6
```

```
),_U6),_A7)]]),
psic([[h(tell(_Y6,_Z6,answer(_W6,_X6,_V6),_U6),_B7)],[],[],[],[],[],[]],[[e(te
ll(_Z6,_Y6,bid(_X6,_V6),_U6),_A7)]]),



h(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),



psic([[h(tell(_D7,auctioneer,bid([art1],_C7),auction1),_H7),h(tell(auctioneer,
bidders,closeauction([art1],19,10,100,50),auction1),_G7)],[],[],[],[],[],[]],[
[_F7#>=19,0#<_E7,_E7#=<_G7,e(tell(auctioneer,auctioneer,reservedinfo([art1],19
,_F7),auction1),_E7)]]),
psic([[h(tell(_J7,auctioneer,bid([art1],_I7),auction1),_L7)],[],[],[],[],[],[]
],[[_L7#<_M7,_M7#=<_L7+100,_N7#>=_I7+10,e(tell(_O7,auctioneer,bid([art1],_N7),
auction1),_M7)],[_K7#=_L7+100,e(tell(auctioneer,bidders,closeauction([art1],19
,10,100,50),auction1),_K7)]]),
e(tell(b1,auctioneer,bid([art1],19),auction1),100),
en(tell(_P,_Q,openauction(_N,_O,_M,_L,_K),auction1),_A),



h(tell(b1,auctioneer,bid([art1],19),auction1),100),



e(tell(b2,auctioneer,bid([art1],29),auction1),200),
psic([[h(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),_
R7)],[],[],[],[],[],[]],[[_Q7#>=19,0#<_P7,_P7#=<_R7,e(tell(auctioneer,auctione
er,reservedinfo([art1],19,_Q7),auction1),_P7)]]),
psic([[h(tell(_T7,auctioneer,bid([art1],_S7),auction1),_Y7),h(tell(auctioneer,
_B8,closeauction([art1],_A8,_Z7,_X7,_V7),auction1),_W7)],[],[],[],[],[],[st(_W
7#=_Y7+_X7),st(reif_unify(...))]],[[_W7#<_U7,_U7#=<_W7+_V7,e(tell(auctioneer,b
1,answer(lose,[art1],19),auction1),_U7)]]),
psic([[h(tell(auctioneer,_G8,closeauction([art1],_F8,_E8,_D8,_C8),auction1),_I
8)],[],[],[],[],[],[st(_I8#=100+_D8),st(reif_unify(...))]],[[_I8#<_H8,_H8#=<_I
8+_C8,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_H8)]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_K8,_J8),auction1),_R8
),h(tell(auctioneer,_Q8,closeauction([art1],_K8,_P8,_O8,_M8),auction1),_N8)],[
],[],[],[],[],[st(_N8#=100+_O8),st(19#<_J8)]],[[_N8#<_L8,_L8#=<_N8+_M8,e(tell(
auctioneer,b1,answer(lose,[art1],19),auction1),_L8)]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_T8,_S8),auction1),_A9
),h(tell(auctioneer,_Z8,closeauction([art1],_T8,_Y8,_X8,_V8),auction1),_W8)],[
],[],[],[],[],[st(_W8#=100+_X8),st(19#>=_S8)]],[[_W8#<_U8,_U8#=<_W8+_V8,e(tell
(auctioneer,b1,answer(win,[art1],19),auction1),_U8)]]),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),
```

```
h(tell(b2,auctioneer,bid([art1],29),auction1),200),



psic([[h(tell(auctioneer,_F9,closeauction([art1],_E9,_D9,_C9,_B9),auction1),_H
9)],[],[],[],[],[],[st(_H9#=200+_C9),st(reif_unify(...))]],[[_H9#<_G9,_G9#=<_H
9+_B9,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_G9)]]),
e(tell(b3,auctioneer,bid([art1],39),auction1),300),
psic([[h(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),_
K9)],[],[],[],[],[],[]],[[_J9#>=19,0#<_I9,_I9#=<_K9,e(tell(auctioneer,auctione
er,reservedinfo([art1],19,_J9),auction1),_I9)]]),
psic([[h(tell(_M9,auctioneer,bid([art1],_L9),auction1),_R9),h(tell(auctioneer,
_U9,closeauction([art1],_T9,_S9,_Q9,_O9),auction1),_P9)],[],[],[],[],[],[st(_P
9#=_R9+_Q9),st(reif_unify(...))]],[[_P9#<_N9,_N9#=<_P9+_O9,e(tell(auctioneer,b
2,answer(lose,[art1],29),auction1),_N9)]]),
psic([[h(tell(auctioneer,_Z9,closeauction([art1],_Y9,_X9,_W9,_V9),auction1),_B
10)],[],[],[],[],[],[st(_B10#=200+_W9),st(reif_unify(...))]],[[_B10#<_A10,_A10
#=<_B10+_V9,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_A10)]]),
psic([[h(tell(auctioneer,_G10,closeauction([art1],_F10,_E10,_D10,_C10),auction
1),_I10)],[],[],[],[],[],[st(_I10#=100+_D10),st(reif_unify(...))]],[[_I10#<_H1
0,_H10#=<_I10+_C10,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_H10)
]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_K10,_J10),auction1),_
R10),h(tell(auctioneer,_Q10,closeauction([art1],_K10,_P10,_O10,_M10),auction1)
,_N10)],[],[],[],[],[],[st(_N10#=200+_O10),st(29#<_J10)]],[[_N10#<_L10,_L10#=<
_N10+_M10,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_L10)]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_T10,_S10),auction1),_
A11),h(tell(auctioneer,_Z10,closeauction([art1],_T10,_Y10,_X10,_V10),auction1)
,_W10)],[],[],[],[],[],[st(_W10#=200+_X10),st(29#>=_S10)]],[[_W10#<_U10,_U10#=
<_W10+_V10,e(tell(auctioneer,b2,answer(win,[art1],29),auction1),_U10)]]),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),



h(tell(b3,auctioneer,bid([art1],39),auction1),300),



psic([[h(tell(auctioneer,_F11,closeauction([art1],_E11,_D11,_C11,_B11),auction
1),_H11)],[],[],[],[],[],[st(_H11#=300+_C11),st(reif_unify(...))]],[[_H11#<_G1
1,_G11#=<_H11+_B11,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_G11)
]]),
psic([[h(tell(auctioneer,_M11,closeauction([art1],_L11,_K11,_J11,_I11),auction
1),_O11)],[],[],[],[],[],[st(_O11#=300+_J11),st(reif_unify(...))]],[[_O11#<_N1
```

```
1,_N11#=<_O11+_I11,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_N11)
]]),
e(tell(b4,auctioneer,bid([art1],49),auction1),400),
psic([[h(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),_
R11)],[],[],[],[],[],[]],[[_Q11#>=19,0#<_P11,_P11#=<_R11,e(tell(auctioneer,auc
tioneer,reservedinfo([art1],19,_Q11),auction1),_P11)]]),
psic([[h(tell(_T11,auctioneer,bid([art1],_S11),auction1),_Y11),h(tell(auctione
er,_B12,closeauction([art1],_A12,_Z11,_X11,_V11),auction1),_W11)],[],[],[],[],
[],[st(_W11#=_Y11+_X11),st(reif_unify(...))]],[[_W11#<_U11,_U11#=<_W11+_V11,e(
tell(auctioneer,b3,answer(lose,[art1],39),auction1),_U11)]]),
psic([[h(tell(auctioneer,_G12,closeauction([art1],_F12,_E12,_D12,_C12),auction
1),_I12)],[],[],[],[],[],[st(_I12#=300+_D12),st(reif_unify(...))]],[[_I12#<_H1
2,_H12#=<_I12+_C12,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_H12)
]]),
psic([[h(tell(auctioneer,_N12,closeauction([art1],_M12,_L12,_K12,_J12),auction
1),_P12)],[],[],[],[],[],[st(_P12#=200+_K12),st(reif_unify(...))]],[[_P12#<_O1
2,_O12#=<_P12+_J12,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_O12)
]]),
psic([[h(tell(auctioneer,_U12,closeauction([art1],_T12,_S12,_R12,_Q12),auction
1),_W12)],[],[],[],[],[],[st(_W12#=100+_R12),st(reif_unify(...))]],[[_W12#<_V1
2,_V12#=<_W12+_Q12,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_V12)
]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_Y12,_X12),auction1),_
F13),h(tell(auctioneer,_E13,closeauction([art1],_Y12,_D13,_C13,_A13),auction1)
,_B13)],[],[],[],[],[],[st(_B13#=300+_C13),st(39#<_X12)]],[[_B13#<_Z12,_Z12#=<
_B13+_A13,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_Z12)]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_H13,_G13),auction1),_
O13),h(tell(auctioneer,_N13,closeauction([art1],_H13,_M13,_L13,_J13),auction1)
,_K13)],[],[],[],[],[],[st(_K13#=300+_L13),st(39#>=_G13)]],[[_K13#<_I13,_I13#=
<_K13+_J13,e(tell(auctioneer,b3,answer(win,[art1],39),auction1),_I13)]]),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),



h(tell(b4,auctioneer,bid([art1],49),auction1),400),



psic([[h(tell(auctioneer,_T13,closeauction([art1],_S13,_R13,_Q13,_P13),auction
1),_V13)],[],[],[],[],[],[st(_V13#=400+_Q13),st(reif_unify(...))]],[[_V13#<_U1
3,_U13#=<_V13+_P13,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_U13)
]]),
psic([[h(tell(auctioneer,_A14,closeauction([art1],_Z13,_Y13,_X13,_W13),auction
1),_C14)],[],[],[],[],[],[st(_C14#=400+_X13),st(reif_unify(...))]],[[_C14#<_B1
4,_B14#=<_C14+_W13,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_B14)
]]),
psic([[h(tell(auctioneer,_H14,closeauction([art1],_G14,_F14,_E14,_D14),auction
```

```
1),_J14)],[],[],[],[],[],[st(_J14#=400+_E14),st(reif_unify(...))]],[[_J14#<_I1
4,_I14#=<_J14+_D14,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_I14)
]]),
e(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),500),
psic([[h(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),_
M14)],[],[],[],[],[],[]],[[_L14#>=19,0#<_K14,_K14#=<_M14,e(tell(auctioneer,auc
tioneer,reservedinfo([art1],19,_L14),auction1),_K14)]]),
psic([[h(tell(_O14,auctioneer,bid([art1],_N14),auction1),_T14),h(tell(auctione
er,_W14,closeauction([art1],_V14,_U14,_S14,_Q14),auction1),_R14)],[],[],[],[],
[],[st(_R14#=_T14+_S14),st(reif_unify(...))]],[[_R14#<_P14,_P14#=<_R14+_Q14,e(
tell(auctioneer,b4,answer(lose,[art1],49),auction1),_P14)]]),
psic([[h(tell(auctioneer,_B15,closeauction([art1],_A15,_Z14,_Y14,_X14),auction
1),_D15)],[],[],[],[],[],[st(_D15#=400+_Y14),st(reif_unify(...))]],[[_D15#<_C1
5,_C15#=<_D15+_X14,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_C15)
]]),
psic([[h(tell(auctioneer,_I15,closeauction([art1],_H15,_G15,_F15,_E15),auction
1),_K15)],[],[],[],[],[],[st(_K15#=300+_F15),st(reif_unify(...))]],[[_K15#<_J1
5,_J15#=<_K15+_E15,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_J15)
]]),
psic([[h(tell(auctioneer,_P15,closeauction([art1],_O15,_N15,_M15,_L15),auction
1),_R15)],[],[],[],[],[],[st(_R15#=200+_M15),st(reif_unify(...))]],[[_R15#<_Q1
5,_Q15#=<_R15+_L15,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_Q15)
]]),
psic([[h(tell(auctioneer,_W15,closeauction([art1],_V15,_U15,_T15,_S15),auction
1),_Y15)],[],[],[],[],[],[st(_Y15#=100+_T15),st(reif_unify(...))]],[[_Y15#<_X1
5,_X15#=<_Y15+_S15,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_X15)
]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_A16,_Z15),auction1),_
H16),h(tell(auctioneer,_G16,closeauction([art1],_A16,_F16,_E16,_C16),auction1)
,_D16)],[],[],[],[],[],[st(_D16#=400+_E16),st(49#<_Z15)]],[[_D16#<_B16,_B16#=<
_D16+_C16,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_B16)]]),
psic([[h(tell(auctioneer,auctioneer,reservedinfo([art1],_J16,_I16),auction1),_
Q16),h(tell(auctioneer,_P16,closeauction([art1],_J16,_O16,_N16,_L16),auction1)
,_M16)],[],[],[],[],[],[st(_M16#=400+_N16),st(49#>=_I16)]],[[_M16#<_K16,_K16#=
<_M16+_L16,e(tell(auctioneer,b4,answer(win,[art1],49),auction1),_K16)]]),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),



h(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500),



psic([[h(tell(auctioneer,_U16,closeauction([art1],19,_T16,_S16,_R16),auction1)
,_W16)],[],[],[],[],[],[st(_W16#=400+_S16),st(49#>=39)]],[[_W16#<_V16,_V16#=<_
W16+_R16,e(tell(auctioneer,b4,answer(win,[art1],49),auction1),_V16)]]),
psic([[h(tell(auctioneer,_A17,closeauction([art1],19,_Z16,_Y16,_X16),auction1)
```

```
,_C17)],[],[],[],[],[],[st(_C17#=400+_Y16),st(49#<39)]],[[_C17#<_B17,_B17#=<_C
17+_X16,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_B17)]]),
psic([[h(tell(auctioneer,_G17,closeauction([art1],19,_F17,_E17,_D17),auction1)
,_I17)],[],[],[],[],[],[st(_I17#=300+_E17),st(39#>=39)]],[[_I17#<_H17,_H17#=<_
I17+_D17,e(tell(auctioneer,b3,answer(win,[art1],39),auction1),_H17)]]),
psic([[h(tell(auctioneer,_M17,closeauction([art1],19,_L17,_K17,_J17),auction1)
,_O17)],[],[],[],[],[],[st(_O17#=300+_K17),st(39#<39)]],[[_O17#<_N17,_N17#=<_O
17+_J17,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_N17)]]),
psic([[h(tell(auctioneer,_S17,closeauction([art1],19,_R17,_Q17,_P17),auction1)
,_U17)],[],[],[],[],[],[st(_U17#=200+_Q17),st(29#>=39)]],[[_U17#<_T17,_T17#=<_
U17+_P17,e(tell(auctioneer,b2,answer(win,[art1],29),auction1),_T17)]]),
psic([[h(tell(auctioneer,_Y17,closeauction([art1],19,_X17,_W17,_V17),auction1)
,_A18)],[],[],[],[],[],[st(_A18#=200+_W17),st(29#<39)]],[[_A18#<_Z17,_Z17#=<_A
18+_V17,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_Z17)]]),
psic([[h(tell(auctioneer,_E18,closeauction([art1],19,_D18,_C18,_B18),auction1)
,_G18)],[],[],[],[],[],[st(_G18#=100+_C18),st(19#>=39)]],[[_G18#<_F18,_F18#=<_
G18+_B18,e(tell(auctioneer,b1,answer(win,[art1],19),auction1),_F18)]]),
psic([[h(tell(auctioneer,_K18,closeauction([art1],19,_J18,_I18,_H18),auction1)
,_M18)],[],[],[],[],[],[st(_M18#=100+_I18),st(19#<39)]],[[_M18#<_L18,_L18#=<_M
18+_H18,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_L18)]]),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),
e(tell(b4,auctioneer,bid([art1],49),auction1),400),



h(tell(auctioneer,bidders,closeauction([art1],19,10,100,50),auction1),500),



psic([[],[],[],[],[],[],[st(500#=100+100),st(19#<39)]],[[500#<_N18,_N18#=<500+
50,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_N18)]]),
psic([[],[],[],[],[],[],[st(500#=100+100),st(19#>=39)]],[[500#<_O18,_O18#=<500
+50,e(tell(auctioneer,b1,answer(win,[art1],19),auction1),_O18)]]),
psic([[],[],[],[],[],[],[st(500#=200+100),st(29#<39)]],[[500#<_P18,_P18#=<500+
50,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_P18)]]),
psic([[],[],[],[],[],[],[st(500#=200+100),st(29#>=39)]],[[500#<_Q18,_Q18#=<500
+50,e(tell(auctioneer,b2,answer(win,[art1],29),auction1),_Q18)]]),
psic([[],[],[],[],[],[],[st(500#=300+100),st(39#<39)]],[[500#<_R18,_R18#=<500+
50,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_R18)]]),
psic([[],[],[],[],[],[],[st(500#=300+100),st(39#>=39)]],[[500#<_S18,_S18#=<500
+50,e(tell(auctioneer,b3,answer(win,[art1],39),auction1),_S18)]]),
psic([[],[],[],[],[],[],[st(500#=400+100),st(49#<39)]],[[500#<_T18,_T18#=<500+
50,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_T18)]]),
e(tell(auctioneer,b4,answer(win,[art1],49),auction1),550),
psic([[],[],[],[],[],[],[st(500#=100+100),st(reif_unify(...))]],[[500#<_U18,_U
18#=<500+50,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_U18)]]),
psic([[],[],[],[],[],[],[st(500#=200+100),st(reif_unify(...))]],[[500#<_V18,_V
```

```
18#=<500+50,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_V18)]]),
psic([[],[],[],[],[],[],[st(500#=300+100),st(reif_unify(...))]],[[500#<_W18,_W
18#=<500+50,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_W18)]]),
psic([[],[],[],[],[],[],[st(500#=400+100),st(reif_unify(...))]],[[500#<_X18,_X
18#=<500+50,e(tell(auctioneer,b4,answer(lose,[art1],49),auction1),_X18)]]),
e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500),
e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),550),
e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),550),
e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),550),
psic([[],[],[],[],[],[],[st(500#=100+100),st(reif_unify(...))]],[[500#<_Y18,_Y
18#=<500+50,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_Y18)]]),
psic([[],[],[],[],[],[],[st(500#=200+100),st(reif_unify(...))]],[[500#<_Z18,_Z
18#=<500+50,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_Z18)]]),
psic([[],[],[],[],[],[],[st(500#=300+100),st(reif_unify(...))]],[[500#<_A19,_A
19#=<500+50,e(tell(auctioneer,b3,answer(lose,[art1],39),auction1),_A19)]]),
e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500),
psic([[],[],[],[],[],[],[st(500#=300+100),st(reif_unify(...))]],[[500#<_B19,_B
19#=<500+50,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_B19)]]),
psic([[],[],[],[],[],[],[st(500#=300+100),st(reif_unify(...))]],[[500#<_C19,_C
19#=<500+50,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_C19)]]),
psic([[],[],[],[],[],[],[st(500#=100+100),st(reif_unify(...))]],[[500#<_D19,_D
19#=<500+50,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_D19)]]),
psic([[],[],[],[],[],[],[st(500#=200+100),st(reif_unify(...))]],[[500#<_E19,_E
19#=<500+50,e(tell(auctioneer,b2,answer(lose,[art1],29),auction1),_E19)]]),
e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500),
psic([[],[],[],[],[],[],[st(500#=200+100),st(reif_unify(...))]],[[500#<_F19,_F
19#=<500+50,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_F19)]]),
psic([[],[],[],[],[],[],[st(500#=100+100),st(reif_unify(...))]],[[500#<_G19,_G
19#=<500+50,e(tell(auctioneer,b1,answer(lose,[art1],19),auction1),_G19)]]),
e(tell(auctioneer,auctioneer,reservedinfo([art1],19,39),auction1),500),
e(tell(auctioneer,bidders,openauction([art1],19,10,100,50),auction1),0),



h(tell(auctioneer,b1,answer(lose,[art1],19),auction1),550),



e(tell(b1,auctioneer,bid([art1],19),auction1),100),
en(tell(auctioneer,b1,answer(win,[art1],_J),auction1),_I),



h(tell(auctioneer,b2,answer(lose,[art1],29),auction1),550),
```

```
e(tell(b2,auctioneer,bid([art1],29),auction1),200),
en(tell(auctioneer,b2,answer(win,[art1],_H),auction1),_G),



h(tell(auctioneer,b3,answer(lose,[art1],39),auction1),550),



e(tell(b3,auctioneer,bid([art1],39),auction1),300),
en(tell(auctioneer,b3,answer(win,[art1],_F),auction1),_E),



h(tell(auctioneer,b4,answer(win,[art1],49),auction1),550),



e(tell(b4,auctioneer,bid([art1],49),auction1),400),
en(tell(auctioneer,b4,answer(lose,[art1],_D),auction1),_C),
st(_A,0#\=_A),
st(_B,0#\=_B) ? ;
no
| ?-
```

Appendice C

Verifiche di proprietà

C.1 History di controesempio vuota

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 4.6.1 relativa alla generazione di una history di controesempio nella verifica di una proprietà goduta dell'asta Inglese.

```
| ?- run.
close_history,
current_time(0),
fulf(en(tell(_A,_B,answer(win,_C,_D),_E),_F)),
ic([h(tell(_M,_N,closeauction(_H,_I,_J,_K,_L),_O),_P)],[[_I>0,_J>0,_K>0,_L>0,
e(tell(_M,_N,openauction(_H,_I,_J,_K,_L),_O),_G)]]),
psic([[h(tell(_M,_N,closeauction(_H,_I,_J,_K,_L),_O),_P)],[],[],[],[],[],[]],
[[st(_I#>0),st(_J#>0),st(_K#>0),st(_L#>0),e(tell(_M,_N,openauction(_H,_I,_J,_
K,
_L),_O),_G)]]),
ic([h(tell(_D1,_E1,openauction(_Y,_Z,_A1,_B1,_C1),_F1),_G1)],[[<>(_G1,_X),en(
tell(_V,_W,openauction(_Q,_R,_S,_T,_U),_F1),_X)]]),
psic([[h(tell(_D1,_E1,openauction(_Y,_Z,_A1,_B1,_C1),_F1),_G1)],[],[],[],[],[
],[]],[[st(_G1#\=_X),en(tell(_V,_W,openauction(_Q,_R,_S,_T,_U),_F1),_X)]]),
ic([h(tell(_Q1,_R1,openauction(_L1,_M1,_N1,_O1,_P1),_S1),_T1)],[[_T1<_K1,_K1=
<_T1+_O1,_J1>=_M1,e(tell(_I1,_Q1,bid(_L1,_J1),_S1),_K1)],[e(tell(_Q1,_R1,clos
ea
uction(_L1,_M1,_N1,_O1,_P1),_S1),_H1),_H1=_T1+_O1]]),
psic([[h(tell(_Q1,_R1,openauction(_L1,_M1,_N1,_O1,_P1),_S1),_T1)],[],[],[],[]
,[],[]],[[_T1#<_K1,_K1#=<_T1+_O1,_J1#>=_M1,e(tell(_I1,_Q1,bid(_L1,_J1),_S1),_
K1
)],[_H1#=_T1+_O1,e(tell(_Q1,_R1,closeauction(_L1,_M1,_N1,_O1,_P1),_S1),_H1)]]
),
```

```
ic([h(tell(_C2,_D2,bid(_A2,_B2),_E2),_F2)],[[e(tell(_D2,_Y1,openauction(_A2,_
U1,_V1,_W1,_X1),_E2),_Z1)]]),
psic([[h(tell(_C2,_D2,bid(_A2,_B2),_E2),_F2)],[],[],[],[],[],[]],[[e(tell(_D2
,_Y1,openauction(_A2,_U1,_V1,_W1,_X1),_E2),_Z1)]]),
ic([h(tell(_S2,_T2,openauction(_N2,_O2,_P2,_Q2,_R2),_U2),_V2),h(tell(_L2,_S2,
bid(_N2,_K2),_U2),_M2)],[[_M2<_J2,_J2=<_M2+_Q2,_I2>=_K2+_P2,e(tell(_H2,_S2,bi
d(
_N2,_I2),_U2),_J2)],[e(tell(_S2,_T2,closeauction(_N2,_O2,_P2,_Q2,_R2),_U2),_G
2),_G2=_M2+_Q2]]),
psic([[h(tell(_S2,_T2,openauction(_N2,_O2,_P2,_Q2,_R2),_U2),_V2),h(tell(_L2,_
S2,bid(_N2,_K2),_U2),_M2)],[],[],[],[],[],[]],[[_M2#<_J2,_J2#=<_M2+_Q2,_I2#>=
_K
2+_P2,e(tell(_H2,_S2,bid(_N2,_I2),_U2),_J2)],[_G2#=_M2+_Q2,e(tell(_S2,_T2,clo
seauction(_N2,_O2,_P2,_Q2,_R2),_U2),_G2)]]),
ic([h(tell(_H3,_I3,openauction(_C3,_D3,_E3,_F3,_G3),_J3),_K3),h(tell(_A3,_H3,
bid(_C3,_Z2),_J3),_B3),h(tell(_H3,_I3,closeauction(_C3,_D3,_E3,_F3,_G3),_J3),
_Y
2)],[[_X2>=_D3,_K3<_W2,_W2=<_Y2,e(tell(_H3,_H3,reservedinfo(_C3,_D3,_X2),_J3)
,_W2)]]),
psic([[h(tell(_H3,_I3,openauction(_C3,_D3,_E3,_F3,_G3),_J3),_K3),h(tell(_A3,_
H3,bid(_C3,_Z2),_J3),_B3),h(tell(_H3,_I3,closeauction(_C3,_D3,_E3,_F3,_G3),_J
3)
,_Y2)],[],[],[],[],[],[]],[[_X2#>=_D3,_K3#<_W2,_W2#=<_Y2,e(tell(_H3,_H3,reser
vedinfo(_C3,_D3,_X2),_J3),_W2)]]),
ic([h(tell(_W3,_W3,reservedinfo(_T3,_U3,_V3),_X3),_Y3)],[[e(tell(_W3,_R3,open
auction(_T3,_U3,_O3,_P3,_Q3),_X3),_S3),e(tell(_M3,_W3,bid(_T3,_L3),_X3),_N3)]
])
,
psic([[h(tell(_W3,_W3,reservedinfo(_T3,_U3,_V3),_X3),_Y3)],[],[],[],[],[],[]]
,[[e(tell(_W3,_R3,openauction(_T3,_U3,_O3,_P3,_Q3),_X3),_S3),e(tell(_M3,_W3,b
id
(_T3,_L3),_X3),_N3)]]),
ic([h(tell(_K4,_L4,bid(_I4,_J4),_M4),_N4),h(tell(_L4,_L4,reservedinfo(_I4,_F4
,_G4),_M4),_H4),h(tell(_L4,_D4,closeauction(_I4,_F4,_A4,_B4,_C4),_M4),_E4),_E
4=
_N4+_B4,_J4>=_G4],[[_E4<_Z3,_Z3=<_E4+_C4,e(tell(_L4,_K4,answer(win,_I4,_J4),_
M4),_Z3)]]),
psic([[h(tell(_K4,_L4,bid(_I4,_J4),_M4),_N4),h(tell(_L4,_L4,reservedinfo(_I4,
_F4,_G4),_M4),_H4),h(tell(_L4,_D4,closeauction(_I4,_F4,_A4,_B4,_C4),_M4),_E4)
],
[],[],[],[],[],[st(_E4#=_N4+_B4),st(_J4#>=_G4)]],[[_E4#<_Z3,_Z3#=<_E4+_C4,e(t
ell(_L4,_K4,answer(win,_I4,_J4),_M4),_Z3)]]),
ic([h(tell(_Z4,_A5,bid(_X4,_Y4),_B5),_C5),h(tell(_A5,_A5,reservedinfo(_X4,_U4
,_V4),_B5),_W4),h(tell(_A5,_S4,closeauction(_X4,_U4,_P4,_Q4,_R4),_B5),_T4),_T
4=
_C5+_Q4,_Y4<_V4],[[_T4<_O4,_O4=<_T4+_R4,e(tell(_A5,_Z4,answer(lose,_X4,_Y4),_
B5),_O4)]]),
```

```
psic([[h(tell(_Z4,_A5,bid(_X4,_Y4),_B5),_C5),h(tell(_A5,_A5,reservedinfo(_X4,
_U4,_V4),_B5),_W4),h(tell(_A5,_S4,closeauction(_X4,_U4,_P4,_Q4,_R4),_B5),_T4)
],
[],[],[],[],[],[st(_T4#=_C5+_Q4),st(_Y4#<_V4)]],[[_T4#<_O4,_O4#=<_T4+_R4,e(te
ll(_A5,_Z4,answer(lose,_X4,_Y4),_B5),_O4)]]),
ic([h(tell(_P5,_Q5,bid(_N5,_O5),_R5),_S5),h(tell(_L5,_Q5,bid(_N5,_K5),_R5),_M
5),h(tell(_Q5,_I5,closeauction(_N5,_E5,_F5,_G5,_H5),_R5),_J5),_J5=_M5+_G5,not
_u
nif(_P5,_L5)],[[_J5<_D5,_D5=<_J5+_H5,e(tell(_Q5,_P5,answer(lose,_N5,_O5),_R5)
,_D5)]]),
psic([[h(tell(_P5,_Q5,bid(_N5,_O5),_R5),_S5),h(tell(_L5,_Q5,bid(_N5,_K5),_R5)
,_M5),h(tell(_Q5,_I5,closeauction(_N5,_E5,_F5,_G5,_H5),_R5),_J5)],[],[],[],[]
,[
],[st(_J5#=_M5+_G5),st(reif_unify(...))]],[[_J5#<_D5,_D5#=<_J5+_H5,e(tell(_Q5
,_P5,answer(lose,_N5,_O5),_R5),_D5)]]),
ic([h(tell(_X5,_Y5,answer(win,_V5,_W5),_Z5),_A6)],[[en(tell(_X5,_Y5,answer(lo
se,_V5,_T5),_Z5),_U5)]]),
psic([[h(tell(_X5,_Y5,answer(win,_V5,_W5),_Z5),_A6)],[],[],[],[],[],[]],[[en(
tell(_X5,_Y5,answer(lose,_V5,_T5),_Z5),_U5)]]),
ic([h(tell(_F6,_G6,answer(lose,_D6,_E6),_H6),_I6)],[[en(tell(_F6,_G6,answer(w
in,_D6,_B6),_H6),_C6)]]),
psic([[h(tell(_F6,_G6,answer(lose,_D6,_E6),_H6),_I6)],[],[],[],[],[],[]],[[en
(tell(_F6,_G6,answer(win,_D6,_B6),_H6),_C6)]]),
ic([h(tell(_N6,_O6,answer(_K6,_L6,_M6),_P6),_Q6)],[[e(tell(_O6,_N6,bid(_L6,_M
6),_P6),_J6)]]),
psic([[h(tell(_N6,_O6,answer(_K6,_L6,_M6),_P6),_Q6)],[],[],[],[],[],[]],[[e(t
ell(_O6,_N6,bid(_L6,_M6),_P6),_J6)]]),
en(tell(_A,_B,answer(win,_C,_D),_E),_F) ? ;
no
| ?-
```

C.2 History di controesempio

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 5.6.1 relativa alla generazione di una history di controesempio nella verifica di una proprietà goduta dell'asta Combinatoria.

```
| ?- run.
==ChesioPrint==: Inserted
h(tell(_149192,_149457,closeauction(_148096,_148373,_148650),_149722),_148927
)
==ChesioPrint==: Inserted
h(tell(_211782,_211536,openauction(_211290,_211044,_210798),_210552),_212014)
existsf(_A),
forallf(_B),
```

```
forallf(_C),
close_history,
current_time(0),
fulf(e(tell(_H,_G,closeauction(_J,_F,_E),_I),_D)),
fulf(e(tell(_H,_G,openauction(_J,_F,_E),_I),_A)),
fulf(e(tell(_H,_G,closeauction(_J,_F,_E),_I),_D)),
fulf(en(tell(_N,_O,openauction(_L,_M,_K),_I),_B)),
fulf(en(tell(_H,_G,answer(_P,_Q,_R),_I),_S)),
ic([h(tell(_X,_Y,openauction(_V,_W,_U),_T),_A1)],[[_W>0,_U>0,_A1>0,_Z=_A1+_W,
e(tell(_X,_Y,closeauction(_V,_W,_U),_T),_Z)]]),
psic([[h(tell(_X,_Y,openauction(_V,_W,_U),_T),_A1)],[],[],[],[],[],[]],[[st(_
W#>0),st(_U#>0),st(_A1#>0),_Z#=_A1+_W,e(tell(_X,_Y,closeauction(_V,_W,_U),_T)
,_
Z)]]),
ic([h(tell(_F1,_G1,openauction(_D1,_E1,_C1),_B1),_N1)],[[<>(_N1,_M1),en(tell(
_K1,_L1,openauction(_H1,_I1,_J1),_B1),_M1)]]),
psic([[h(tell(_F1,_G1,openauction(_D1,_E1,_C1),_B1),_N1)],[],[],[],[],[],[]],
[[st(_N1#\=_M1),en(tell(_K1,_L1,openauction(_H1,_I1,_J1),_B1),_M1)]]),
ic([h(tell(_S1,_T1,closeauction(_Q1,_R1,_P1),_O1),_V1)],[[e(tell(_S1,_T1,open
auction(_Q1,_R1,_P1),_O1),_U1)]]),
psic([[h(tell(_S1,_T1,closeauction(_Q1,_R1,_P1),_O1),_V1)],[],[],[],[],[],[]]
,[[e(tell(_S1,_T1,openauction(_Q1,_R1,_P1),_O1),_U1)]]),
ic([h(tell(_C2,_D2,bid(_A2,_B2),_E2),_F2)],[[_B2>0,_Z1<_F2,_F2=<_Z1+_Y1,e(tel
l(_D2,_C2,openauction(_W1,_Y1,_X1),_E2),_Z1),included(_A2,_W1)]]),
psic([[h(tell(_C2,_D2,bid(_A2,_B2),_E2),_F2)],[],[],[],[],[],[]],[[st(_B2#>0)
,_Z1#<_F2,_F2#=<_Z1+_Y1,e(tell(_D2,_C2,openauction(_W1,_Y1,_X1),_E2),_Z1),inc
lu
ded(_A2,_W1)]]),
ic([h(tell(_L2,_M2,bid(_J2,_K2),_N2),_O2)],[[<>(_O2,_I2),en(tell(_L2,_M2,bid(
_G2,_H2),_N2),_I2)]]),
psic([[h(tell(_L2,_M2,bid(_J2,_K2),_N2),_O2)],[],[],[],[],[],[]],[[st(_O2#\=_
I2),en(tell(_L2,_M2,bid(_G2,_H2),_N2),_I2)]]),
ic([h(tell(_T2,_U2,closeauction(_R2,_S2,_Q2),_P2),_Z2),h(tell(_U2,_T2,bid(_W2
,_X2),_P2),_Y2)],[[_Z2<_V2,_V2=<_Z2+_Q2,e(tell(_T2,_U2,answer(win,_W2,_X2),_P
2)
,_V2)],[e(tell(_T2,_U2,answer(lose,_W2,_X2),_P2),_V2),_Z2<_V2,_V2=<_Z2+_Q2]])
,
psic([[h(tell(_T2,_U2,closeauction(_R2,_S2,_Q2),_P2),_Z2),h(tell(_U2,_T2,bid(
_W2,_X2),_P2),_Y2)],[],[],[],[],[],[]],[[_Z2#<_V2,_V2#=<_Z2+_Q2,e(tell(_T2,_U
2,
answer(win,_W2,_X2),_P2),_V2)],[_Z2#<_A3,st(_A3#=<_Z2+_Q2),e(tell(_T2,_U2,ans
wer(lose,_W2,_X2),_P2),_A3)]]),
ic([h(tell(_F3,_G3,answer(_C3,_D3,_E3),_H3),_I3)],[[e(tell(_G3,_F3,bid(_D3,_E
3),_H3),_B3)]]),
psic([[h(tell(_F3,_G3,answer(_C3,_D3,_E3),_H3),_I3)],[],[],[],[],[],[]],[[e(t
ell(_G3,_F3,bid(_D3,_E3),_H3),_B3)]]),
ic([h(tell(_N3,_O3,answer(win,_L3,_M3),_P3),_Q3)],[[en(tell(_N3,_O3,answer(lo
```

```
se,_L3,_J3),_P3),_K3)]]),
psic([[h(tell(_N3,_O3,answer(win,_L3,_M3),_P3),_Q3)],[],[],[],[],[],[]],[[en(
tell(_N3,_O3,answer(lose,_L3,_J3),_P3),_K3)]]),
ic([h(tell(_V3,_W3,answer(lose,_T3,_U3),_X3),_Y3)],[[en(tell(_V3,_W3,answer(w
in,_T3,_R3),_X3),_S3)]]),
psic([[h(tell(_V3,_W3,answer(lose,_T3,_U3),_X3),_Y3)],[],[],[],[],[],[]],[[en
(tell(_V3,_W3,answer(win,_T3,_R3),_X3),_S3)]]),
e(tell(_H,_G,closeauction(_J,_F,_E),_I),_D),
en(tell(_H,_G,answer(_P,_Q,_R),_I),_S),



h(tell(_H,_G,closeauction(_J,_F,_E),_I),_D),



psic([[h(tell(_G,_H,bid(_B4,_A4),_I),_D4)],[],[],[],[],[],[]],[[_D#<_C4,_C4#=
<_D+_E,e(tell(_H,_G,answer(win,_B4,_A4),_I),_C4)],[_D#<_Z3,st(_Z3#=<_D+_E),e(
te
ll(_H,_G,answer(lose,_B4,_A4),_I),_Z3)]]),
e(tell(_H,_G,openauction(_J,_F,_E),_I),_A),



h(tell(_H,_G,openauction(_J,_F,_E),_I),_A),



en(tell(_N,_O,openauction(_L,_M,_K),_I),_B),
e(tell(_H,_G,closeauction(_J,_F,_E),_I),_D),
st(_A,_A#\=_B),
st(_A,_A#\=_C),
st(_B,_A#\=_B),
st(_C,_A#\=_C) ? ;



==ChesioPrint==: Inserted
h(tell(_292642,_292660,closeauction(_292606,_292624,_292588),_292568),_297441
)
==ChesioPrint==: Inserted
h(tell(_392570,_392833,openauction(_392044,_392307,_391781),_391518),_393096)
==ChesioFail==:
h(tell(_392570,_392833,openauction(_392044,_392307,_391781),_391518),_393096)
==ChesioPrint==: Inserted
h(tell(_303111,_303374,closeauction(_302571,_302848,_302294),_302017),_303651
)
```

```
==ChesioPrint==: Inserted
h(tell(_379392,_379655,openauction(_378852,_379129,_378575),_378298),_379918)
==ChesioFail==:
h(tell(_379392,_379655,openauction(_378852,_379129,_378575),_378298),_379918)
existsf(_A),
forallf(_B),
forallf(_C),
close_history,
current_time(0),
fulf(e(tell(_G,_F,closeauction(_I,_E,_D),_H),_J)),
fulf(e(tell(_G,_F,openauction(_I,_E,_D),_H),_A)),
fulf(e(tell(_G,_F,closeauction(_I,_E,_D),_H),_K)),
fulf(e(tell(_G,_F,openauction(_I,_E,_D),_H),_A)),
fulf(en(tell(_O,_P,openauction(_M,_N,_L),_H),_B)),
fulf(en(tell(_G,_F,answer(_Q,_R,_S),_H),_T)),
ic([h(tell(_Y,_Z,openauction(_W,_X,_V),_U),_B1)],[[_X>0,_V>0,_B1>0,_A1=_B1+_X
,e(tell(_Y,_Z,closeauction(_W,_X,_V),_U),_A1)]]),
psic([[h(tell(_Y,_Z,openauction(_W,_X,_V),_U),_B1)],[],[],[],[],[],[]],[[st(_
X#>0),st(_V#>0),st(_B1#>0),_A1#=_B1+_X,e(tell(_Y,_Z,closeauction(_W,_X,_V),_U
),
_A1)]]),
ic([h(tell(_G1,_H1,openauction(_E1,_F1,_D1),_C1),_O1)],[[<>(_O1,_N1),en(tell(
_L1,_M1,openauction(_I1,_J1,_K1),_C1),_N1)]]),
psic([[h(tell(_G1,_H1,openauction(_E1,_F1,_D1),_C1),_O1)],[],[],[],[],[],[]],
[[st(_O1#\=_N1),en(tell(_L1,_M1,openauction(_I1,_J1,_K1),_C1),_N1)]]),
ic([h(tell(_T1,_U1,closeauction(_R1,_S1,_Q1),_P1),_W1)],[[e(tell(_T1,_U1,open
auction(_R1,_S1,_Q1),_P1),_V1)]]),
psic([[h(tell(_T1,_U1,closeauction(_R1,_S1,_Q1),_P1),_W1)],[],[],[],[],[],[]]
,[[e(tell(_T1,_U1,openauction(_R1,_S1,_Q1),_P1),_V1)]]),
ic([h(tell(_D2,_E2,bid(_B2,_C2),_F2),_G2)],[[_C2>0,_A2<_G2,_G2=<_A2+_Z1,e(tel
l(_E2,_D2,openauction(_X1,_Z1,_Y1),_F2),_A2),included(_B2,_X1)]]),
psic([[h(tell(_D2,_E2,bid(_B2,_C2),_F2),_G2)],[],[],[],[],[],[]],[[st(_C2#>0)
,_A2#<_G2,_G2#=<_A2+_Z1,e(tell(_E2,_D2,openauction(_X1,_Z1,_Y1),_F2),_A2),inc
lu
ded(_B2,_X1)]]),
ic([h(tell(_M2,_N2,bid(_K2,_L2),_O2),_P2)],[[<>(_P2,_J2),en(tell(_M2,_N2,bid(
_H2,_I2),_O2),_J2)]]),
psic([[h(tell(_M2,_N2,bid(_K2,_L2),_O2),_P2)],[],[],[],[],[],[]],[[st(_P2#\=_
J2),en(tell(_M2,_N2,bid(_H2,_I2),_O2),_J2)]]),
ic([h(tell(_U2,_V2,closeauction(_S2,_T2,_R2),_Q2),_A3),h(tell(_V2,_U2,bid(_X2
,_Y2),_Q2),_Z2)],[[_A3<_W2,_W2=<_A3+_R2,e(tell(_U2,_V2,answer(win,_X2,_Y2),_Q
2)
,_W2)],[e(tell(_U2,_V2,answer(lose,_X2,_Y2),_Q2),_W2),_A3<_W2,_W2=<_A3+_R2]])
,
psic([[h(tell(_U2,_V2,closeauction(_S2,_T2,_R2),_Q2),_A3),h(tell(_V2,_U2,bid(
_X2,_Y2),_Q2),_Z2)],[],[],[],[],[],[]],[[_A3#<_W2,_W2#=<_A3+_R2,e(tell(_U2,_V
2,
```

```
answer(win,_X2,_Y2),_Q2),_W2)],[_A3#<_B3,st(_B3#=<_A3+_R2),e(tell(_U2,_V2,ans
wer(lose,_X2,_Y2),_Q2),_B3)]]),
ic([h(tell(_G3,_H3,answer(_D3,_E3,_F3),_I3),_J3)],[[e(tell(_H3,_G3,bid(_E3,_F
3),_I3),_C3)]]),
psic([[h(tell(_G3,_H3,answer(_D3,_E3,_F3),_I3),_J3)],[],[],[],[],[],[]],[[e(t
ell(_H3,_G3,bid(_E3,_F3),_I3),_C3)]]),
ic([h(tell(_O3,_P3,answer(win,_M3,_N3),_Q3),_R3)],[[en(tell(_O3,_P3,answer(lo
se,_M3,_K3),_Q3),_L3)]]),
psic([[h(tell(_O3,_P3,answer(win,_M3,_N3),_Q3),_R3)],[],[],[],[],[],[]],[[en(
tell(_O3,_P3,answer(lose,_M3,_K3),_Q3),_L3)]]),
ic([h(tell(_W3,_X3,answer(lose,_U3,_V3),_Y3),_Z3)],[[en(tell(_W3,_X3,answer(w
in,_U3,_S3),_Y3),_T3)]]),
psic([[h(tell(_W3,_X3,answer(lose,_U3,_V3),_Y3),_Z3)],[],[],[],[],[],[]],[[en
(tell(_W3,_X3,answer(win,_U3,_S3),_Y3),_T3)]]),
e(tell(_G,_F,closeauction(_I,_E,_D),_H),_J),
en(tell(_G,_F,answer(_Q,_R,_S),_H),_T),



h(tell(_G,_F,closeauction(_I,_E,_D),_H),_J),



psic([[h(tell(_F,_G,bid(_C4,_B4),_H),_E4)],[],[],[],[],[],[]],[[_J#<_D4,_D4#=
<_J+_D,e(tell(_G,_F,answer(win,_C4,_B4),_H),_D4)],[_J#<_A4,st(_A4#=<_J+_D),e(
te
ll(_G,_F,answer(lose,_C4,_B4),_H),_A4)]]),
e(tell(_G,_F,openauction(_I,_E,_D),_H),_A),



h(tell(_G,_F,openauction(_I,_E,_D),_H),_A),



en(tell(_O,_P,openauction(_M,_N,_L),_H),_B),
e(tell(_G,_F,closeauction(_I,_E,_D),_H),_K),
reified_unif:not_unify_constr(_J,_K),
h(tell(_G,_F,closeauction(_I,_E,_D),_H),_K),
psic([[h(tell(_F,_G,bid(_H4,_G4),_H),_J4)],[],[],[],[],[],[]],[[_K#<_I4,_I4#=
<_K+_D,e(tell(_G,_F,answer(win,_H4,_G4),_H),_I4)],[_K#<_F4,st(_F4#=<_K+_D),e(
te
ll(_G,_F,answer(lose,_H4,_G4),_H),_F4)]]),
e(tell(_G,_F,openauction(_I,_E,_D),_H),_A),
st(_A,_A#\=_B),
st(_A,_A#\=_C),
st(_B,_A#\=_B),
```

```
st(_C,_A#\=_C) ? ;



==ChesioPrint==: Inserted
h(tell(_369690,_369708,openauction(_369654,_369672,_369636),_369616),_374403)
==ChesioPrint==: Inserted
h(tell(_380475,_380738,openauction(_379935,_380212,_379658),_379381),_381015)
==ChesioFail==:
h(tell(_380475,_380738,openauction(_379935,_380212,_379658),_379381),_381015)
==ChesioFail==:
h(tell(_303111,_303374,closeauction(_302571,_302848,_302294),_302017),_303651
)
==ChesioFail==:
h(tell(_211782,_211536,openauction(_211290,_211044,_210798),_210552),_212014)
==ChesioFail==:
h(tell(_149192,_149457,closeauction(_148096,_148373,_148650),_149722),_148927
)
no
| ?-
```

C.3 Probabile successo

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 5.6.2 relativa ad un probabile successo nella verifica di una proprietà goduta dell'asta Combinatoria.

```
| ?- run.
==ChesioPrint==: Inserted
h(tell(_147556,_147821,bid(_146737,_147014),_148086),_147291)
==ChesioPrint==: Inserted
h(tell(_229230,_229462,openauction(_228070,_228766,_228534),_228998),_230859)
==ChesioPrint==: Inserted
h(tell(_308570,_308338,closeauction(_307642,_307396,_307150),_308106),_307874)
==ChesioPrint==: Inserted
h(tell(_93957,_94189,answer(win,_93725,_93493),_93261),_93029)
==ChesioFail==: h(tell(_93957,_94189,answer(win,_93725,_93493),_93261),_93029)
==ChesioPrint==: Inserted
h(tell(_94142,_94374,answer(lose,_93910,_93678),_93446),_93214)
==ChesioFail==:
h(tell(_94142,_94374,answer(lose,_93910,_93678),_93446),_93214)
==ChesioFail==:
h(tell(_34080,_34030,closeauction(_33880,_33831,_33782),_33980),_33930)
==ChesioFail==:
h(tell(_24770,_24820,openauction(_24520,_24670,_24620),_24720),_25154)
```

```
==ChesioPrint==: Inserted
h(tell(_24770,_24820,openauction(_24520,_24670,_24620),_24720),_24570)
==ChesioPrint==: Inserted
h(tell(_100696,_100464,closeauction(_99768,_99522,_99276),_100232),_100000)
==ChesioPrint==: Inserted
h(tell(_166853,_167085,answer(win,_166621,_166389),_166157),_165925)
==ChesioFail==: h(tell(_41822,_41872,answer(win,_41772,_41722),_41672),_41622)
==ChesioPrint==: Inserted
h(tell(_44887,_45119,answer(lose,_44655,_44423),_44191),_43959)
==ChesioFail==:
h(tell(_44887,_45119,answer(lose,_44655,_44423),_44191),_43959)
==ChesioFail==:
h(tell(_33500,_33450,closeauction(_33300,_33251,_33202),_33400),_33350)
==ChesioFail==:
h(tell(_24770,_24820,openauction(_24520,_24670,_24620),_24720),_24570)
==ChesioPrint==: Inserted
h(tell(_60707,_60939,openauction(_59547,_60243,_60011),_60475),_62336)
==ChesioPrint==: Inserted
h(tell(_140047,_139815,closeauction(_139119,_138873,_138627),_139583),_139351)
==ChesioPrint==: Inserted
h(tell(_206204,_206436,answer(win,_205972,_205740),_205508),_205276)
==ChesioFail==: h(tell(_45043,_45093,answer(win,_44993,_44943),_44893),_44843)
==ChesioPrint==: Inserted
h(tell(_48108,_48340,answer(lose,_47876,_47644),_47412),_47180)
==ChesioFail==:
h(tell(_48108,_48340,answer(lose,_47876,_47644),_47412),_47180)
==ChesioFail==:
h(tell(_36721,_36671,closeauction(_36521,_36472,_36423),_36621),_36571)
==ChesioFail==:
h(tell(_27411,_27461,openauction(_27161,_27311,_27261),_27361),_27795)
==ChesioPrint==: Inserted
h(tell(_27411,_27461,openauction(_27161,_27311,_27261),_27361),_27211)
==ChesioPrint==: Inserted
h(tell(_103337,_103105,closeauction(_102409,_102163,_101917),_102873),_102641)
==ChesioPrint==: Inserted
h(tell(_169494,_169726,answer(win,_169262,_169030),_168798),_168566)
==ChesioFail==: h(tell(_44463,_44513,answer(win,_44413,_44363),_44313),_44263)
==ChesioPrint==: Inserted
h(tell(_47528,_47760,answer(lose,_47296,_47064),_46832),_46600)
==ChesioFail==:
h(tell(_47528,_47760,answer(lose,_47296,_47064),_46832),_46600)
==ChesioFail==:
h(tell(_36141,_36091,closeauction(_35941,_35892,_35843),_36041),_35991)
==ChesioFail==:
h(tell(_27411,_27461,openauction(_27161,_27311,_27261),_27361),_27211)
==ChesioFail==: h(tell(_14854,_14901,bid(_14719,_14763),_14948),_14807)
no
```

```
| ?-
```

C.4 Probabile loop infinito

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 4.6.2 relativa all'entrata in loop infinito nella verifica di una proprietà goduta dell'asta Inglese.

```
==ChesioPrint==: Inserted
h(tell(_283451,_283716,answer(_282909,_283981,_284246),_284511),_283186)
==ChesioPrint==: Inserted
h(tell(_311140,_311386,bid(_310894,_310648),_310402),_312286)
==ChesioPrint==: Inserted
h(tell(_475379,_476771,openauction(_475133,_476539,_476307,_476075,_475843),_
474887),_475611)
==ChesioPrint==: Inserted
h(tell(_633457,_632903,bid(_632640,_633180),_632377),_633720)
==ChesioPrint==: Inserted
h(tell(_160021,_159467,bid(_159204,_159744),_158941),_158678)
==ChesioPrint==: Inserted
h(tell(_523855,_420104,openauction(_523819,_420082,_420060,_420042,_420024),_
523745),_531607)
==ChesioPrint==: Inserted
h(tell(_188260,_187706,bid(_187443,_187983),_187180),_186917)
==ChesioPrint==: Inserted
h(tell(_1022508,_1023893,openauction(_1022245,_1023616,_1023339,_1023062,_102
2785),_1021982),_1021719)
==ChesioPrint==: Inserted
h(tell(_278865,_278311,bid(_278048,_278588),_277785),_277522)
==ChesioPrint==: Inserted
h(tell(_374115,_373561,bid(_373298,_373838),_373035),_372772)
==ChesioPrint==: Inserted
h(tell(_468099,_467545,bid(_467282,_467822),_467019),_466756)
==ChesioPrint==: Inserted
h(tell(_550174,_550272,openauction(_550244,_549410,_549396,_549382,_549368),_
550090),_548172)
==ChesioPrint==: Inserted
h(tell(_594956,_594402,bid(_594139,_594679),_593876),_593613)
==ChesioPrint==: Inserted
h(tell(_740367,_739813,bid(_739550,_740090),_739287),_739024)
==ChesioPrint==: Inserted
h(tell(_847415,_848800,openauction(_847152,_848523,_848246,_847969,_847692),_
846889),_846626)
==ChesioPrint==: Inserted
h(tell(_896379,_895825,bid(_895562,_896102),_895299),_895036)
```

```
==ChesioPrint==: Inserted
h(tell(_875719,_875165,bid(_874902,_875442),_874639),_874376)
Prolog interruption (h for help)? a
\% Execution aborted
```

C.5 History parziale e probabile loop infinito

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 4.6.3 relativa ad un primo tentativo di generazione di una history di controesempio, con successiva entrata in loop infinito, nella verifica di una proprietà goduta dell'asta Inglese.

```
| ?- run.
==ChesioPrint==: Inserted
h(tell(_282485,_283304,bid(_283569,_282762),_283834),_283039)
==ChesioPrint==: Inserted
h(tell(_442189,_443581,openauction(_441943,_443349,_443117,_442885,_442653),_
441697),_442421)
existsf(_A),
forallf(_B),
forallf(_C),
close_history,
current_time(0),
fulf(e(tell(_I,_E,bid(_D,_H),_F),_G)),
fulf(e(tell(_E,_N,openauction(_D,_M,_L,_K,_J),_F),_A)),
fulf(e(tell(_I,_E,bid(_D,_H),_F),_G)),
fulf(e(tell(_I,_E,bid(_D,_H),_F),_G)),
fulf(en(tell(_T,_U,openauction(_R,_S,_Q,_P,_O),_F),_B)),
fulf(en(tell(_E,_V,answer(win,_D,_W),_F),_X)),
ic([h(tell(_E1,_F1,closeauction(_Z,_A1,_B1,_C1,_D1),_G1),_H1)],[[_A1>0,_B1>0,
_C1>0,_D1>0,e(tell(_E1,_F1,openauction(_Z,_A1,_B1,_C1,_D1),_G1),_Y)]]),
psic([[h(tell(_E1,_F1,closeauction(_Z,_A1,_B1,_C1,_D1),_G1),_H1)],[],[],[],[]
,[],[]],[[st(_A1#>0),st(_B1#>0),st(_C1#>0),st(_D1#>0),e(tell(_E1,_F1,openauct
io
n(_Z,_A1,_B1,_C1,_D1),_G1),_Y)]]),
ic([h(tell(_O1,_P1,openauction(_M1,_N1,_L1,_K1,_J1),_I1),_Y1)],[[<>(_Y1,_X1),
en(tell(_V1,_W1,openauction(_Q1,_R1,_S1,_T1,_U1),_I1),_X1)]]),
psic([[h(tell(_O1,_P1,openauction(_M1,_N1,_L1,_K1,_J1),_I1),_Y1)],[],[],[],[]
,[],[]],[[st(_Y1#\=_X1),en(tell(_V1,_W1,openauction(_Q1,_R1,_S1,_T1,_U1),_I1)
,_
X1)]]),
ic([h(tell(_F2,_G2,openauction(_D2,_E2,_C2,_B2,_A2),_Z1),_L2)],[[_L2<_K2,_K2=
<_L2+_B2,_J2>=_E2,e(tell(_I2,_F2,bid(_D2,_J2),_Z1),_K2)],[e(tell(_F2,_G2,clos
ea
```

```
uction(_D2,_E2,_C2,_B2,_A2),_Z1),_H2),_H2=_L2+_B2]]),
psic([[h(tell(_F2,_G2,openauction(_D2,_E2,_C2,_B2,_A2),_Z1),_L2)],[],[],[],[]
,[],[]],[[_L2#<_K2,_K2#=<_L2+_B2,_J2#>=_E2,e(tell(_I2,_F2,bid(_D2,_J2),_Z1),_
K2
)],[_H2#=_L2+_B2,e(tell(_F2,_G2,closeauction(_D2,_E2,_C2,_B2,_A2),_Z1),_H2)]]
),
ic([h(tell(_P2,_Q2,bid(_N2,_O2),_M2),_X2)],[[e(tell(_Q2,_V2,openauction(_N2,_
R2,_S2,_T2,_U2),_M2),_W2)]]),
psic([[h(tell(_P2,_Q2,bid(_N2,_O2),_M2),_X2)],[],[],[],[],[],[]],[[e(tell(_Q2
,_V2,openauction(_N2,_R2,_S2,_T2,_U2),_M2),_W2)]]),
ic([h(tell(_E3,_F3,openauction(_C3,_D3,_B3,_A3,_Z2),_Y2),_N3),h(tell(_L3,_E3,
bid(_C3,_K3),_Y2),_M3)],[[_M3<_J3,_J3=<_M3+_A3,_I3>=_K3+_B3,e(tell(_H3,_E3,bi
d(
_C3,_I3),_Y2),_J3)],[e(tell(_E3,_F3,closeauction(_C3,_D3,_B3,_A3,_Z2),_Y2),_G
3),_G3=_M3+_A3]]),
psic([[h(tell(_E3,_F3,openauction(_C3,_D3,_B3,_A3,_Z2),_Y2),_N3),h(tell(_L3,_
E3,bid(_C3,_K3),_Y2),_M3)],[],[],[],[],[],[]],[[_M3#<_J3,_J3#=<_M3+_A3,_I3#>=
_K
3+_B3,e(tell(_H3,_E3,bid(_C3,_I3),_Y2),_J3)],[_G3#=_M3+_A3,e(tell(_E3,_F3,clo
seauction(_C3,_D3,_B3,_A3,_Z2),_Y2),_G3)]]),
ic([h(tell(_U3,_V3,openauction(_S3,_T3,_R3,_Q3,_P3),_O3),_C4),h(tell(_A4,_U3,
bid(_S3,_Z3),_O3),_B4),h(tell(_U3,_V3,closeauction(_S3,_T3,_R3,_Q3,_P3),_O3),
_Y
3)],[[_X3>=_T3,_C4<_W3,_W3=<_Y3,e(tell(_U3,_U3,reservedinfo(_S3,_T3,_X3),_O3)
,_W3)]]),
psic([[h(tell(_U3,_V3,openauction(_S3,_T3,_R3,_Q3,_P3),_O3),_C4),h(tell(_A4,_
U3,bid(_S3,_Z3),_O3),_B4),h(tell(_U3,_V3,closeauction(_S3,_T3,_R3,_Q3,_P3),_O
3)
,_Y3)],[],[],[],[],[],[]],[[_X3#>=_T3,_C4#<_W3,_W3#=<_Y3,e(tell(_U3,_U3,reser
vedinfo(_S3,_T3,_X3),_O3),_W3)]]),
ic([h(tell(_O4,_O4,reservedinfo(_L4,_M4,_N4),_P4),_Q4)],[[e(tell(_O4,_J4,open
auction(_L4,_M4,_G4,_H4,_I4),_P4),_K4),e(tell(_E4,_O4,bid(_L4,_D4),_P4),_F4)]
])
,
psic([[h(tell(_O4,_O4,reservedinfo(_L4,_M4,_N4),_P4),_Q4)],[],[],[],[],[],[]]
,[[e(tell(_O4,_J4,openauction(_L4,_M4,_G4,_H4,_I4),_P4),_K4),e(tell(_E4,_O4,b
id
(_L4,_D4),_P4),_F4)]]),
ic([h(tell(_U4,_V4,bid(_S4,_T4),_R4),_F5),h(tell(_V4,_V4,reservedinfo(_S4,_C5
,_D5),_R4),_E5),h(tell(_V4,_A5,closeauction(_S4,_C5,_X4,_Y4,_Z4),_R4),_B5),_B
5=
_F5+_Y4,_T4>=_D5],[[_B5<_W4,_W4=<_B5+_Z4,e(tell(_V4,_U4,answer(win,_S4,_T4),_
R4),_W4)]]),
psic([[h(tell(_U4,_V4,bid(_S4,_T4),_R4),_F5),h(tell(_V4,_V4,reservedinfo(_S4,
_C5,_D5),_R4),_E5),h(tell(_V4,_A5,closeauction(_S4,_C5,_X4,_Y4,_Z4),_R4),_B5)
],
[],[],[],[],[],[st(_B5#=_F5+_Y4),st(_T4#>=_D5)]],[[_B5#<_W4,_W4#=<_B5+_Z4,e(t
```

```
ell(_V4,_U4,answer(win,_S4,_T4),_R4),_W4)]]),
ic([h(tell(_J5,_K5,bid(_H5,_I5),_G5),_U5),h(tell(_K5,_K5,reservedinfo(_H5,_R5
,_S5),_G5),_T5),h(tell(_K5,_P5,closeauction(_H5,_R5,_M5,_N5,_O5),_G5),_Q5),_Q
5=
_U5+_N5,_I5<_S5],[[_Q5<_L5,_L5=<_Q5+_O5,e(tell(_K5,_J5,answer(lose,_H5,_I5),_
G5),_L5)]]),
psic([[h(tell(_J5,_K5,bid(_H5,_I5),_G5),_U5),h(tell(_K5,_K5,reservedinfo(_H5,
_R5,_S5),_G5),_T5),h(tell(_K5,_P5,closeauction(_H5,_R5,_M5,_N5,_O5),_G5),_Q5)
],
[],[],[],[],[],[st(_Q5#=_U5+_N5),st(_I5#<_S5)]],[[_Q5#<_L5,_L5#=<_Q5+_O5,e(te
ll(_K5,_J5,answer(lose,_H5,_I5),_G5),_L5)]]),
ic([h(tell(_Y5,_Z5,bid(_W5,_X5),_V5),_K6),h(tell(_I6,_Z5,bid(_W5,_H6),_V5),_J
6),h(tell(_Z5,_F6,closeauction(_W5,_B6,_C6,_D6,_E6),_V5),_G6),_G6=_J6+_D6,not
_u
nif(_Y5,_I6)],[[_G6<_A6,_A6=<_G6+_E6,e(tell(_Z5,_Y5,answer(lose,_W5,_X5),_V5)
,_A6)]]),
psic([[h(tell(_Y5,_Z5,bid(_W5,_X5),_V5),_K6),h(tell(_I6,_Z5,bid(_W5,_H6),_V5)
,_J6),h(tell(_Z5,_F6,closeauction(_W5,_B6,_C6,_D6,_E6),_V5),_G6)],[],[],[],[]
,[
],[st(_G6#=_J6+_D6),st(reif_unify(...))]],[[_G6#<_A6,_A6#=<_G6+_E6,e(tell(_Z5
,_Y5,answer(lose,_W5,_X5),_V5),_A6)]]),
ic([h(tell(_P6,_Q6,answer(win,_N6,_O6),_R6),_S6)],[[en(tell(_P6,_Q6,answer(lo
se,_N6,_L6),_R6),_M6)]]),
psic([[h(tell(_P6,_Q6,answer(win,_N6,_O6),_R6),_S6)],[],[],[],[],[],[]],[[en(
tell(_P6,_Q6,answer(lose,_N6,_L6),_R6),_M6)]]),
ic([h(tell(_X6,_Y6,answer(lose,_V6,_W6),_Z6),_A7)],[[en(tell(_X6,_Y6,answer(w
in,_V6,_T6),_Z6),_U6)]]),
psic([[h(tell(_X6,_Y6,answer(lose,_V6,_W6),_Z6),_A7)],[],[],[],[],[],[]],[[en
(tell(_X6,_Y6,answer(win,_V6,_T6),_Z6),_U6)]]),
ic([h(tell(_F7,_G7,answer(_C7,_D7,_E7),_H7),_I7)],[[e(tell(_G7,_F7,bid(_D7,_E
7),_H7),_B7)]]),
psic([[h(tell(_F7,_G7,answer(_C7,_D7,_E7),_H7),_I7)],[],[],[],[],[],[]],[[e(t
ell(_G7,_F7,bid(_D7,_E7),_H7),_B7)]]),
e(tell(_I,_E,bid(_D,_H),_F),_G),
en(tell(_E,_V,answer(win,_D,_W),_F),_X),



h(tell(_I,_E,bid(_D,_H),_F),_G),



psic([[h(tell(_K7,_E,bid(_D,_J7),_F),_P7),h(tell(_E,_S7,closeauction(_D,_R7,_
Q7,_O7,_M7),_F),_N7)],[],[],[],[],[],[st(_N7#=_P7+_O7),st(reif_unify(...))]],
[[
_N7#<_L7,_L7#=<_N7+_M7,e(tell(_E,_I,answer(lose,_D,_H),_F),_L7)]]),
psic([[h(tell(_E,_Z7,closeauction(_D,_Y7,_X7,_W7,_U7),_F),_V7)],[],[],[],[],[
```

```
],[st(_V7#=_G+_W7),st(reif_unify(...))]],[[_V7#<_T7,_T7#=<_V7+_U7,e(tell(_E,_
I,
answer(lose,_D,_H),_F),_T7)]]),
psic([[h(tell(_E,_E,reservedinfo(_D,_G8,_D8),_F),_I8),h(tell(_E,_H8,closeauct
ion(_D,_G8,_F8,_E8,_B8),_F),_C8)],[],[],[],[],[],[st(_C8#=_G+_E8),st(_H#<_D8)
]]
,[[_C8#<_A8,_A8#=<_C8+_B8,e(tell(_E,_I,answer(lose,_D,_H),_F),_A8)]]),
psic([[h(tell(_E,_E,reservedinfo(_D,_P8,_M8),_F),_R8),h(tell(_E,_Q8,closeauct
ion(_D,_P8,_O8,_N8,_K8),_F),_L8)],[],[],[],[],[],[st(_L8#=_G+_N8),st(_H#>=_M8
)]
],[[_L8#<_J8,_J8#=<_L8+_K8,e(tell(_E,_I,answer(win,_D,_H),_F),_J8)]]),
e(tell(_E,_N,openauction(_D,_M,_L,_K,_J),_F),_A),



h(tell(_E,_N,openauction(_D,_M,_L,_K,_J),_F),_A),



psic([[h(tell(_T8,_E,bid(_D,_S8),_F),_X8),h(tell(_E,_N,closeauction(_D,_M,_L,
_K,_J),_F),_W8)],[],[],[],[],[],[]],[[_V8#>=_M,_A#<_U8,_U8#=<_W8,e(tell(_E,_E
,r
eservedinfo(_D,_M,_V8),_F),_U8)]]),
psic([[h(tell(_E,_N,closeauction(_D,_M,_L,_K,_J),_F),_A9)],[],[],[],[],[],[]]
,[[_Z8#>=_M,_A#<_Y8,_Y8#=<_A9,e(tell(_E,_E,reservedinfo(_D,_M,_Z8),_F),_Y8)]]
),

psic([[h(tell(_C9,_E,bid(_D,_B9),_F),_E9)],[],[],[],[],[],[]],[[_E9#<_F9,_F9#
=<_E9+_K,_G9#>=_B9+_L,e(tell(_H9,_E,bid(_D,_G9),_F),_F9)],[_D9#=_E9+_K,e(tell
(_
E,_N,closeauction(_D,_M,_L,_K,_J),_F),_D9)]]),
e(tell(_I,_E,bid(_D,_H),_F),_G),
e(tell(_I,_E,bid(_D,_H),_F),_G),
en(tell(_T,_U,openauction(_R,_S,_Q,_P,_O),_F),_B),



clpfd:'x=<y IND'(_M,_H),
clpfd:'t=<u+c'(_A,_G,-1),
clpfd:'t=<u+c'(_G,_G,-1),
clpfd:scalar_product([1,-1,-1],[_G,_A,_K],#=<,0),
clpfd:scalar_product([1,-1,1],[_K,_G,_G],#>=,0),
clpfd:scalar_product([1,-1,1],[_L,_H,_H],#=<,0),



_A in inf..sup,
```

```
_K in inf..sup,
_L in inf..sup,
_M in inf..sup,
_G in inf..sup,
_H in inf..sup,
st(_B,_A#\=_B),
st(_A,_A#\=_B),
st(_A,_A#\=_C),
st(_C,_A#\=_C) ? ;



==ChesioPrint==: Inserted
h(tell(_735715,_735663,bid(_735609,_735697),_735589),_739990)
==ChesioPrint==: Inserted
h(tell(_68423,_66156,bid(_66102,_66210),_66082),_67137)
==ChesioPrint==: Inserted
h(tell(_68288,_68236,bid(_68182,_70119),_68162),_67137)
==ChesioPrint==: Inserted
h(tell(_76245,_75428,bid(_75691,_75968),_75165),_74902)
existsf(_A),
forallf(_B),
forallf(_C),
clpfd:'t=u IND'(_D,_E)#<=>_F,
clpfd:'t=u IND'(_G,_H)#<=>_I,
close_history,
current_time(0),
fulf(e(tell(_M,_K,bid(_J,_H),_L),_E)),
fulf(e(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A)),
fulf(e(tell(_M,_K,bid(_J,_H),_L),_E)),
fulf(e(tell(_S,_K,bid(_J,_G),_L),_D)),
fulf(e(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A)),
fulf(e(tell(_S,_K,bid(_J,_G),_L),_D)),
fulf(en(tell(_Y,_Z,openauction(_W,_X,_V,_U,_T),_L),_B)),
fulf(en(tell(_K,_A1,answer(win,_J,_B1),_L),_C1)),
ic([h(tell(_J1,_K1,closeauction(_E1,_F1,_G1,_H1,_I1),_L1),_M1)],[[_F1>0,_G1>0
,_H1>0,_I1>0,e(tell(_J1,_K1,openauction(_E1,_F1,_G1,_H1,_I1),_L1),_D1)]]),
psic([[h(tell(_J1,_K1,closeauction(_E1,_F1,_G1,_H1,_I1),_L1),_M1)],[],[],[],[
],[],[]],[[st(_F1#>0),st(_G1#>0),st(_H1#>0),st(_I1#>0),e(tell(_J1,_K1,openauc
ti
on(_E1,_F1,_G1,_H1,_I1),_L1),_D1)]]),
ic([h(tell(_T1,_U1,openauction(_R1,_S1,_Q1,_P1,_O1),_N1),_D2)],[[<>(_D2,_C2),
en(tell(_A2,_B2,openauction(_V1,_W1,_X1,_Y1,_Z1),_N1),_C2)]]),
psic([[h(tell(_T1,_U1,openauction(_R1,_S1,_Q1,_P1,_O1),_N1),_D2)],[],[],[],[]
,[],[]],[[st(_D2#\=_C2),en(tell(_A2,_B2,openauction(_V1,_W1,_X1,_Y1,_Z1),_N1)
,_
C2)]]),
```

```
ic([h(tell(_K2,_L2,openauction(_I2,_J2,_H2,_G2,_F2),_E2),_Q2)],[[_Q2<_P2,_P2=
<_Q2+_G2,_O2>=_J2,e(tell(_N2,_K2,bid(_I2,_O2),_E2),_P2)],[e(tell(_K2,_L2,clos
ea
uction(_I2,_J2,_H2,_G2,_F2),_E2),_M2),_M2=_Q2+_G2]]),
psic([[h(tell(_K2,_L2,openauction(_I2,_J2,_H2,_G2,_F2),_E2),_Q2)],[],[],[],[]
,[],[]],[[_Q2#<_P2,_P2#=<_Q2+_G2,_O2#>=_J2,e(tell(_N2,_K2,bid(_I2,_O2),_E2),_
P2
)],[_M2#=_Q2+_G2,e(tell(_K2,_L2,closeauction(_I2,_J2,_H2,_G2,_F2),_E2),_M2)]]
),
ic([h(tell(_U2,_V2,bid(_S2,_T2),_R2),_C3)],[[e(tell(_V2,_A3,openauction(_S2,_
W2,_X2,_Y2,_Z2),_R2),_B3)]]),
psic([[h(tell(_U2,_V2,bid(_S2,_T2),_R2),_C3)],[],[],[],[],[],[]],[[e(tell(_V2
,_A3,openauction(_S2,_W2,_X2,_Y2,_Z2),_R2),_B3)]]),
ic([h(tell(_J3,_K3,openauction(_H3,_I3,_G3,_F3,_E3),_D3),_S3),h(tell(_Q3,_J3,
bid(_H3,_P3),_D3),_R3)],[[_R3<_O3,_O3=<_R3+_F3,_N3>=_P3+_G3,e(tell(_M3,_J3,bi
d(
_H3,_N3),_D3),_O3)],[e(tell(_J3,_K3,closeauction(_H3,_I3,_G3,_F3,_E3),_D3),_L
3),_L3=_R3+_F3]]),
psic([[h(tell(_J3,_K3,openauction(_H3,_I3,_G3,_F3,_E3),_D3),_S3),h(tell(_Q3,_
J3,bid(_H3,_P3),_D3),_R3)],[],[],[],[],[],[]],[[_R3#<_O3,_O3#=<_R3+_F3,_N3#>=
_P
3+_G3,e(tell(_M3,_J3,bid(_H3,_N3),_D3),_O3)],[_L3#=_R3+_F3,e(tell(_J3,_K3,clo
seauction(_H3,_I3,_G3,_F3,_E3),_D3),_L3)]]),
ic([h(tell(_Z3,_A4,openauction(_X3,_Y3,_W3,_V3,_U3),_T3),_H4),h(tell(_F4,_Z3,
bid(_X3,_E4),_T3),_G4),h(tell(_Z3,_A4,closeauction(_X3,_Y3,_W3,_V3,_U3),_T3),
_D
4)],[[_C4>=_Y3,_H4<_B4,_B4=<_D4,e(tell(_Z3,_Z3,reservedinfo(_X3,_Y3,_C4),_T3)
,_B4)]]),
psic([[h(tell(_Z3,_A4,openauction(_X3,_Y3,_W3,_V3,_U3),_T3),_H4),h(tell(_F4,_
Z3,bid(_X3,_E4),_T3),_G4),h(tell(_Z3,_A4,closeauction(_X3,_Y3,_W3,_V3,_U3),_T
3)
,_D4)],[],[],[],[],[],[]],[[_C4#>=_Y3,_H4#<_B4,_B4#=<_D4,e(tell(_Z3,_Z3,reser
vedinfo(_X3,_Y3,_C4),_T3),_B4)]]),
ic([h(tell(_T4,_T4,reservedinfo(_Q4,_R4,_S4),_U4),_V4)],[[e(tell(_T4,_O4,open
auction(_Q4,_R4,_L4,_M4,_N4),_U4),_P4),e(tell(_J4,_T4,bid(_Q4,_I4),_U4),_K4)]
])
,
psic([[h(tell(_T4,_T4,reservedinfo(_Q4,_R4,_S4),_U4),_V4)],[],[],[],[],[],[]]
,[[e(tell(_T4,_O4,openauction(_Q4,_R4,_L4,_M4,_N4),_U4),_P4),e(tell(_J4,_T4,b
id
(_Q4,_I4),_U4),_K4)]]),
ic([h(tell(_Z4,_A5,bid(_X4,_Y4),_W4),_K5),h(tell(_A5,_A5,reservedinfo(_X4,_H5
,_I5),_W4),_J5),h(tell(_A5,_F5,closeauction(_X4,_H5,_C5,_D5,_E5),_W4),_G5),_G
5=
_K5+_D5,_Y4>=_I5],[[_G5<_B5,_B5=<_G5+_E5,e(tell(_A5,_Z4,answer(win,_X4,_Y4),_
W4),_B5)]]),
psic([[h(tell(_Z4,_A5,bid(_X4,_Y4),_W4),_K5),h(tell(_A5,_A5,reservedinfo(_X4,
```

```
_H5,_I5),_W4),_J5),h(tell(_A5,_F5,closeauction(_X4,_H5,_C5,_D5,_E5),_W4),_G5)
],
[],[],[],[],[],[st(_G5#=_K5+_D5),st(_Y4#>=_I5)]],[[_G5#<_B5,_B5#=<_G5+_E5,e(t
ell(_A5,_Z4,answer(win,_X4,_Y4),_W4),_B5)]]),
ic([h(tell(_O5,_P5,bid(_M5,_N5),_L5),_Z5),h(tell(_P5,_P5,reservedinfo(_M5,_W5
,_X5),_L5),_Y5),h(tell(_P5,_U5,closeauction(_M5,_W5,_R5,_S5,_T5),_L5),_V5),_V
5=
_Z5+_S5,_N5<_X5],[[_V5<_Q5,_Q5=<_V5+_T5,e(tell(_P5,_O5,answer(lose,_M5,_N5),_
L5),_Q5)]]),
psic([[h(tell(_O5,_P5,bid(_M5,_N5),_L5),_Z5),h(tell(_P5,_P5,reservedinfo(_M5,
_W5,_X5),_L5),_Y5),h(tell(_P5,_U5,closeauction(_M5,_W5,_R5,_S5,_T5),_L5),_V5)
],
[],[],[],[],[],[st(_V5#=_Z5+_S5),st(_N5#<_X5)]],[[_V5#<_Q5,_Q5#=<_V5+_T5,e(te
ll(_P5,_O5,answer(lose,_M5,_N5),_L5),_Q5)]]),
ic([h(tell(_D6,_E6,bid(_B6,_C6),_A6),_P6),h(tell(_N6,_E6,bid(_B6,_M6),_A6),_O
6),h(tell(_E6,_K6,closeauction(_B6,_G6,_H6,_I6,_J6),_A6),_L6),_L6=_O6+_I6,not
_u
nif(_D6,_N6)],[[_L6<_F6,_F6=<_L6+_J6,e(tell(_E6,_D6,answer(lose,_B6,_C6),_A6)
,_F6)]]),
psic([[h(tell(_D6,_E6,bid(_B6,_C6),_A6),_P6),h(tell(_N6,_E6,bid(_B6,_M6),_A6)
,_O6),h(tell(_E6,_K6,closeauction(_B6,_G6,_H6,_I6,_J6),_A6),_L6)],[],[],[],[]
,[
],[st(_L6#=_O6+_I6),st(reif_unify(...))]],[[_L6#<_F6,_F6#=<_L6+_J6,e(tell(_E6
,_D6,answer(lose,_B6,_C6),_A6),_F6)]]),
ic([h(tell(_U6,_V6,answer(win,_S6,_T6),_W6),_X6)],[[en(tell(_U6,_V6,answer(lo
se,_S6,_Q6),_W6),_R6)]]),
psic([[h(tell(_U6,_V6,answer(win,_S6,_T6),_W6),_X6)],[],[],[],[],[],[]],[[en(
tell(_U6,_V6,answer(lose,_S6,_Q6),_W6),_R6)]]),
ic([h(tell(_C7,_D7,answer(lose,_A7,_B7),_E7),_F7)],[[en(tell(_C7,_D7,answer(w
in,_A7,_Y6),_E7),_Z6)]]),
psic([[h(tell(_C7,_D7,answer(lose,_A7,_B7),_E7),_F7)],[],[],[],[],[],[]],[[en
(tell(_C7,_D7,answer(win,_A7,_Y6),_E7),_Z6)]]),
ic([h(tell(_K7,_L7,answer(_H7,_I7,_J7),_M7),_N7)],[[e(tell(_L7,_K7,bid(_I7,_J
7),_M7),_G7)]]),
psic([[h(tell(_K7,_L7,answer(_H7,_I7,_J7),_M7),_N7)],[],[],[],[],[],[]],[[e(t
ell(_L7,_K7,bid(_I7,_J7),_M7),_G7)]]),
e(tell(_M,_K,bid(_J,_H),_L),_E),
en(tell(_K,_A1,answer(win,_J,_B1),_L),_C1),



h(tell(_M,_K,bid(_J,_H),_L),_E),



psic([[h(tell(_P7,_K,bid(_J,_O7),_L),_U7),h(tell(_K,_X7,closeauction(_J,_W7,_
V7,_T7,_R7),_L),_S7)],[],[],[],[],[],[st(_S7#=_U7+_T7),st(reif_unify(...))]],
```

```
[[
_S7#<_Q7,_Q7#=<_S7+_R7,e(tell(_K,_M,answer(lose,_J,_H),_L),_Q7)]]),
psic([[h(tell(_K,_E8,closeauction(_J,_D8,_C8,_B8,_Z7),_L),_A8)],[],[],[],[],[
],[st(_A8#=_E+_B8),st(reif_unify(...))]],[[_A8#<_Y7,_Y7#=<_A8+_Z7,e(tell(_K,_
M,
answer(lose,_J,_H),_L),_Y7)]]),
psic([[h(tell(_K,_K,reservedinfo(_J,_L8,_I8),_L),_N8),h(tell(_K,_M8,closeauct
ion(_J,_L8,_K8,_J8,_G8),_L),_H8)],[],[],[],[],[],[st(_H8#=_E+_J8),st(_H#<_I8)
]]
,[[_H8#<_F8,_F8#=<_H8+_G8,e(tell(_K,_M,answer(lose,_J,_H),_L),_F8)]]),
psic([[h(tell(_K,_K,reservedinfo(_J,_U8,_R8),_L),_W8),h(tell(_K,_V8,closeauct
ion(_J,_U8,_T8,_S8,_P8),_L),_Q8)],[],[],[],[],[],[st(_Q8#=_E+_S8),st(_H#>=_R8
)]
],[[_Q8#<_O8,_O8#=<_Q8+_P8,e(tell(_K,_M,answer(win,_J,_H),_L),_O8)]]),
e(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A),



h(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A),



psic([[h(tell(_Y8,_K,bid(_J,_X8),_L),_C9),h(tell(_K,_R,closeauction(_J,_Q,_P,
_O,_N),_L),_B9)],[],[],[],[],[],[]],[[_A9#>=_Q,_A#<_Z8,_Z8#=<_B9,e(tell(_K,_K
,r
eservedinfo(_J,_Q,_A9),_L),_Z8)]]),
psic([[h(tell(_K,_R,closeauction(_J,_Q,_P,_O,_N),_L),_F9)],[],[],[],[],[],[]]
,[[_E9#>=_Q,_A#<_D9,_D9#=<_F9,e(tell(_K,_K,reservedinfo(_J,_Q,_E9),_L),_D9)]]
),

psic([[h(tell(_H9,_K,bid(_J,_G9),_L),_J9)],[],[],[],[],[],[]],[[_J9#<_K9,_K9#
=<_J9+_O,_L9#>=_G9+_P,e(tell(_M9,_K,bid(_J,_L9),_L),_K9)],[_I9#=_J9+_O,e(tell
(_
K,_R,closeauction(_J,_Q,_P,_O,_N),_L),_I9)]]),
e(tell(_M,_K,bid(_J,_H),_L),_E),
e(tell(_S,_K,bid(_J,_G),_L),_D),



h(tell(_S,_K,bid(_J,_G),_L),_D),



e(tell(_S,_K,bid(_J,_G),_L),_D),
psic([[h(tell(_K,_R,closeauction(_J,_Q,_P,_O,_N),_L),_P9)],[],[],[],[],[],[]]
,[[_O9#>=_Q,_A#<_N9,_N9#=<_P9,e(tell(_K,_K,reservedinfo(_J,_Q,_O9),_L),_N9)]]
),
```

```
psic([[h(tell(_K,_W9,closeauction(_J,_V9,_U9,_T9,_R9),_L),_S9)],[],[],[],[],[
],[st(_S9#=_D+_T9),st(reif_unify(...))]],[[_S9#<_Q9,_Q9#=<_S9+_R9,e(tell(_K,_
M,
answer(lose,_J,_H),_L),_Q9)]]),
psic([[h(tell(_Y9,_K,bid(_J,_X9),_L),_D10),h(tell(_K,_G10,closeauction(_J,_F1
0,_E10,_C10,_A10),_L),_B10)],[],[],[],[],[],[st(_B10#=_D10+_C10),st(reif_unif
y(
...))]],[[_B10#<_Z9,_Z9#=<_B10+_A10,e(tell(_K,_S,answer(lose,_J,_G),_L),_Z9)]
]),
psic([[h(tell(_K,_N10,closeauction(_J,_M10,_L10,_K10,_I10),_L),_J10)],[],[],[
],[],[],[st(_J10#=_D+_K10),st(reif_unify(...))]],[[_J10#<_H10,_H10#=<_J10+_I1
0,
e(tell(_K,_S,answer(lose,_J,_G),_L),_H10)]]),
psic([[h(tell(_K,_U10,closeauction(_J,_T10,_S10,_R10,_P10),_L),_Q10)],[],[],[
],[],[],[st(_Q10#=_E+_R10),st(reif_unify(...))]],[[_Q10#<_O10,_O10#=<_Q10+_P1
0,
e(tell(_K,_S,answer(lose,_J,_G),_L),_O10)]]),
psic([[h(tell(_K,_K,reservedinfo(_J,_B11,_Y10),_L),_D11),h(tell(_K,_C11,close
auction(_J,_B11,_A11,_Z10,_W10),_L),_X10)],[],[],[],[],[],[st(_X10#=_D+_Z10),
st
(_G#<_Y10)]],[[_X10#<_V10,_V10#=<_X10+_W10,e(tell(_K,_S,answer(lose,_J,_G),_L
),_V10)]]),
psic([[h(tell(_K,_K,reservedinfo(_J,_K11,_H11),_L),_M11),h(tell(_K,_L11,close
auction(_J,_K11,_J11,_I11,_F11),_L),_G11)],[],[],[],[],[],[st(_G11#=_D+_I11),
st
(_G#>=_H11)]],[[_G11#<_E11,_E11#=<_G11+_F11,e(tell(_K,_S,answer(win,_J,_G),_L
),_E11)]]),
e(tell(_K,_R,openauction(_J,_Q,_P,_O,_N),_L),_A),
en(tell(_Y,_Z,openauction(_W,_X,_V,_U,_T),_L),_B),
reif_unify(_M,_S,_N11),



clpfd:'x=<y IND'(_Q,_G),
clpfd:'t=<u+c'(_D,_D,-1),
clpfd:'t=<u+c'(_A,_D,-1),
clpfd:'t=<u+c'(_E,_E,-1),
clpfd:bool(2,_N11,1,_O11),
clpfd:bool(2,_I,1,_P11),
clpfd:bool(2,_Q11,_R11,_S11),
clpfd:bool(2,_T11,1,_U11),
clpfd:bool(2,_F,_V11,1),
clpfd:bool(2,1,_W11,_X11),
clpfd:bool(2,1,_Y11,_Z11),
clpfd:bool(6,_W11,_O11,1),
clpfd:bool(6,_Q11,_P11,1),
```

```
clpfd:bool(6,_R11,_X11,1),
clpfd:bool(6,_Y11,_S11,1),
clpfd:bool(6,_T11,_Z11,1),
clpfd:bool(6,_V11,_U11,1),
clpfd:scalar_product([1,-1,-1],[_D,_A,_O],#=<,0),
clpfd:scalar_product([1,-1,1],[_O,_D,_D],#>=,0),
clpfd:scalar_product([1,-1,1],[_O,_E,_E],#>=,0),
clpfd:scalar_product([1,-1,1],[_P,_G,_G],#=<,0),
clpfd:scalar_product([1,-1,1],[_P,_H,_H],#=<,0),



_D in inf..sup,
_G in inf..sup,
_A in inf..sup,
_O in inf..sup,
_P in inf..sup,
_Q in inf..sup,
_E in inf..sup,
_H in inf..sup,
_N11 in 0..1,
_O11 in 0..1,
_W11 in 0..1,
_X11 in 0..1,
_I in 0..1,
_P11 in 0..1,
_Q11 in 0..1,
_R11 in 0..1,
_S11 in 0..1,
_Y11 in 0..1,
_Z11 in 0..1,
_T11 in 0..1,
_U11 in 0..1,
_F in 0..1,
_V11 in 0..1,
st(_B,_A#\=_B),
st(_A,_A#\=_B),
st(_A,_A#\=_C),
st(_C,_A#\=_C) ? ;



==ChesioPrint==: Inserted
h(tell(_451979,_374934,openauction(_451943,_374912,_374890,_374872,_374854),_
451869),_459731)
==ChesioPrint==: Inserted
h(tell(_189633,_189651,openauction(_189597,_102825,_102803,_102785,_102767),_
```

```
189523),_191003)
==ChesioPrint==: Inserted
h(tell(_149440,_149388,bid(_149334,_149422),_149314),_152073)
==ChesioPrint==: Inserted
h(tell(_766560,_767945,openauction(_766297,_767668,_767391,_767114,_766837),_
766034),_765771)
==ChesioPrint==: Inserted
h(tell(_254437,_254385,bid(_254331,_254419),_254311),_257170)
==ChesioPrint==: Inserted
h(tell(_608104,_609489,openauction(_607841,_609212,_608935,_608658,_608381),_
607578),_607315)
==ChesioPrint==: Inserted
h(tell(_367522,_367494,bid(_367452,_367508),_367438),_368580)
==ChesioPrint==: Inserted
h(tell(_451546,_452931,openauction(_451283,_452654,_452377,_452100,_451823),_
451020),_450757)
==ChesioPrint==: Inserted
h(tell(_522634,_522606,bid(_522564,_522620),_522550),_523201)
==ChesioPrint==: Inserted
h(tell(_622204,_623589,openauction(_621941,_623312,_623035,_622758,_622481),_
621678),_621415)
==ChesioPrint==: Inserted
h(tell(_708482,_708454,bid(_708412,_708468),_708398),_709147)
==ChesioPrint==: Inserted
h(tell(_823481,_824866,openauction(_823218,_824589,_824312,_824035,_823758),_
822955),_822692)
Prolog interruption (h for help)? a
% Execution aborted
| ?-
```

C.6 Constraint overflow

Riportiamo l'esito, in versione integrale, della simulazione condotta nella sezione 4.6.4 relativa al verificarsi di un errore di overflow nella verifica di una proprietà goduta dell'asta Inglese.

```
| ?- run.
==ChesioPrint==: Inserted
h(tell(_285763,_286028,closeauction(_286293,_286558,_286823,_287088,_287353),
_287618),_285498)
==ChesioPrint==: Inserted
h(tell(_47888,_47642,openauction(_47396,_47150,_46904,_46658,_46412),_46166),
_48788)
==ChesioPrint==: Inserted
h(tell(_172417,_171721,bid(_171475,_172185),_171229),_171953)
```

```
==ChesioPrint==: Inserted
h(tell(_272056,_272056,reservedinfo(_271810,_271564,_272520),_271318),_272288
)
==ChesioPrint==: Inserted
h(tell(_72194,_73302,openauction(_71668,_71931,_73025,_72748,_72471),_71405),
_73565)
==ChesioPrint==: Inserted
h(tell(_353271,_352533,answer(lose,_353025,_352287),_352779),_353503)
! Representation error in user:'t>=u+c'/3
! CLPFD integer overflow
! goal:  't>=u+c'(_14400,_14051,1)
| ?-
```

Appendice D

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly

available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires

Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.